CENTRO DE CIENCIAS BÁSICAS

DEPARTAMENTO DE SISTEMAS DE INFORMACIÓN

TRABAJO PRÁCTICO

Security Testing Techniques and Tools: A Study for Web Based Applications

QUE PRESENTA

Alan Tonatiuh González Peña

PARA OPTAR POR EL GRADO DE MAESTRÍA EN INFORMÁTICA Y TECNOLOGÍAS COMPUTACIONALES

COMITÉ TUTORAL

Tutor: Dra. Lizeth Itziguery Solano Romo

Cotutor: M. en ITC Edgar Oswaldo Díaz

Asesor: Dr. José Manuel Andrade

Asesor: M. en ITC Jorge Eduardo Macías Luévano

Aguascalientes, Aguascalientes Mayo 2018

UNIVERSIDAD AUTONOMA
DE AGUASCALIENTES

FORMATO DE CARTA DE VOTO APROBATORIO

M. EN C. JOSE DE JESUS RUIZ GALLEGOS
DECANO DEL CENTRO DE CIENCIAS BASICAS
P R E S E N T E

Por medio del presente como Tutor designado del estudiante **ALAN TONATIUH GONZALEZ PEÑA** con ID 91852 quien realizó el trabajo práctico titulado: **SECURITY TESTING TECHNIQUES AND TOOLS: A STUDY FOR WEB BASED APPLICATIONS**, y con fundamento en el Artículo 175, Apartado II del Reglamento General de Docencia, me permito emitir el **VOTO APROBATORIO**, para que él pueda proceder a imprimirlo, y así como continuar con el procedimiento administrativo para la obtención del grado.

Pongo lo anterior a su digna consideración y sin otro particular por el momento, me permito enviarle un cordial saludo.

A T E N T A M E N T E
"Se Lumen Proferre"
Aguascalientes, Ags., a 21 de mayo de 2018.

DRA. LIZETH ITZIGUERY SOLANO ROMO
Tutor de trabajo práctico

c.c.p.- Interesado
c.c.p.- Secretaría Técnica del Programa de Posgrado

INSTITUTO NACIONAL DE ESTADÍSTICA Y GEOGRAFÍA

ASUNTO: CARTA VOTO APROBATORIO

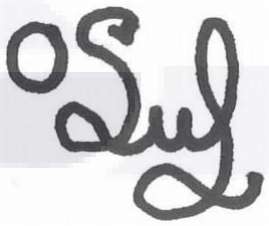
Aguascalientes, Ags., a 21 de mayo de 2018.

M. EN C. JOSE DE JESUS RUIZ GALLEGOS. DECANO DEL CENTRO DE CIENCIAS BASICAS P R E S E N T E

Por medio del presente como Co-Tutor designado del estudiante **ALAN TONATIUH GONZALEZ PEÑA** con [ID91852], quien realizó el trabajo práctico titulado: **SECURITY TESTING TECHNIQUES AND TOOLS: A STUDY FOR WEB BASED APPLICATIONS**, y con fundamento en el Artículo 175, Apartado II del Reglamento General de Docencia, me permito emitir el **VOTO APROBATORIO**, para que él pueda proceder a imprimirlo, y así como continuar con el procedimiento administrativo para la obtención del grado.

Pongo lo anterior a su digna consideración y sin otro particular por el momento, me permito enviarle un cordial saludo.

ATENTAMENTE



MSc. Edgar Oswaldo Díaz
Dirección de Cómputo y Comunicaciones Grupo de Ingeniería en Sistemas

Avenida Héroe de Nacozari Sur 2301
Edificio de Informática, Nivel 1, Fraccionamiento Jardines del Parque
20276, Aguascalientes, Aguascalientes, Aguascalientes
Entre calle INEGI, Avenida del Lago y Avenida Paseo de las Garzas
910 53 00 ext. 4953 oswaldo.diaz@inegi.org.mx

UNIVERSIDAD AUTONOMA
DE AGUASCALIENTES

FORMATO DE CARTA DE VOTO APROBATORIO

M. EN C. JOSE DE JESUS RUIZ GALLEGOS
DECANO DEL CENTRO DE CIENCIAS BASICAS
P R E S E N T E

Por medio del presente como Tutor designado del estudiante **ALAN TONATIUH GONZALEZ PEÑA** con ID 91852 quien realizó el trabajo práctico titulado: **SECURITY TESTING TECHNIQUES AND TOOLS: A STUDY FOR WEB BASED APPLICATIONS**, y con fundamento en el Artículo 175, Apartado II del Reglamento General de Docencia, me permito emitir el **VOTO APROBATORIO**, para que él pueda proceder a imprimirlo, y así como continuar con el procedimiento administrativo para la obtención del grado.

Pongo lo anterior a su digna consideración y sin otro particular por el momento, me permito enviarle un cordial saludo.

A T E N T A M E N T E
"Se Lumen Proferre"
Aguascalientes, Ags., a 21 de mayo de 2018.

DR. JOSE MANUEL ANDRADE
Tutor de trabajo práctico

c.c.p.- Interesado
c.c.p.- Secretaría Técnica del Programa de Posgrado

UNIVERSIDAD AUTONOMA
DE AGUASCALIENTES

FORMATO DE CARTA DE VOTO APROBATORIO

M. EN C. JOSE DE JESUS RUIZ GALLEGOS
DECANO DEL CENTRO DE CIENCIAS BASICAS
P R E S E N T E

Por medio del presente como Tutor designado del estudiante **ALAN TONATIUH GONZALEZ PEÑA** con ID 91852 quien realizó el trabajo práctico titulado: **SECURITY TESTING TECHNIQUES AND TOOLS: A STUDY FOR WEB BASED APPLICATIONS**, y con fundamento en el Artículo 175, Apartado II del Reglamento General de Docencia, me permito emitir el **VOTO APROBATORIO**, para que él pueda proceder a imprimirlo, y así como continuar con el procedimiento administrativo para la obtención del grado.

Pongo lo anterior a su digna consideración y sin otro particular por el momento, me permito enviarle un cordial saludo.

A T E N T A M E N T E
"Se Lumen Proferre"
Aguascalientes, Ags., a 21 de mayo de 2018.

M.I.T.C. JORGE EDUARDO MACIAS LUEVANO
Tutor de trabajo práctico

c.c.p.- Interesado
c.c.p.- Secretaría Técnica del Programa de Posgrado

UNIVERSIDAD AUTONOMA
DE AGUASCALIENTES

ALAN TONATIUH GONZALEZ PEÑA
MAESTRÍA EN INFORMÁTICA Y TECNOLOGÍAS COMPUTACIONALES
P R E S E N T E.

Estimado alumno:

Por medio de este conducto me permito comunicar a Usted que habiendo recibido los votos aprobatorios de los revisores de su trabajo de tesis y/o caso práctico titulado: **"SECURITY TESTING TECHNIQUES AND TOOLS: A STUDY FOR WEB BASED APPLICATIONS"**, hago de su conocimiento que puede imprimir dicho documento y continuar con los trámites para la presentación de su examen de grado.

Sin otro particular me permito saludarle muy afectuosamente.

A T E N T A M E N T E
Aguascalientes, Ags., a 25 de mayo de 2018
"Se lumen proferre"
EL DECANO

M. en C. JOSÉ DE JESÚS RUÍZ GALLEGOS

c.c.p.- Archivo.

Agradecimientos

Me gustaría tomar en esta oportunidad para expresar mi gratitud a mi familia por su apoyo constante y animo sin el cual esto hubiese sido imposible.

También me gustaría dar las gracias a mis tutores, la Dra. Lizeth Itziguery Solano Romo, el Maestro Edgar Oswaldo Díaz, Dr. José Manuel Andrade y al Maestro Jorge Macías Luévano quienes me brindaron su apoyo y estuvieron al tanto durante todo el proceso de este trabajo.

A la Universidad Autónoma de Aguascalientes por el plan académico del que fuimos parte mis compañeros y yo y los recursos brindados para llevar a cabo este trabajo.

Y por último a mis compañeros de trabajo y clases que hicieron especial estos dos años además de su disposición a ayudar, consejos y sugerencias.

Índice

Índice de tablas

Índice de imágenes

Resumen

En este trabajo practico se hará una evaluación en materia de seguridad en un organismo gubernamental con proyectos e información real sobre las amenazas o vulnerabilidades que se puedan presentar, en un ambiente controlado, para esto, en primera parte se usará el marco de trabajo creado por OWASP, denominado SAMM para evaluar la organización y tener información sobre donde se encuentra actualmente en temas de seguridad, debido a la gran extensión del mismo, nos enfocaremos en el apartado de “Verificación” que incluye tres sub módulos, “Revisión del Diseño, Revisión de la Implementación y Pruebas de Seguridad”.

Después del análisis teórico, pasaremos a la práctica buscando amenazas o vulnerabilidades para las aplicaciones con tecnología Web 2.0 tomando en cuenta el proyecto de OWASP Top 10 y obteniendo del mismo, tres que consideramos son las más críticas, con mayor cantidad de ataques y que pueden llegar a tener gran impacto en la organización, estas vulnerabilidades que se revisarán más a fondo son, A1 Inyección (SQL Injection), A3 Secuencia de Comandos en Sitios Cruzados (Cross-Site Scripting o XSS) y A10 APIs no Protegidas (Underprotected APIs) (OWASP, 2017a).

Tomando en cuenta la información que presenta OWASP se montó un laboratorio de pruebas de seguridad en el Instituto Nacional de Estadística y Geografía (INEGI) donde se analizaron aplicaciones con tecnología Web 2.0 del año 2015 y 2016 construidas usando tecnología Oracle Java y Microsoft .Net para su análisis a fondo y confirmación usando una herramienta de código abierto del mismo organismo OWASP, esta herramienta llamada Zed Attack Proxy por sus siglas en inglés “ZAP”, ayuda a correr un escaneo completo de la aplicación con tecnología Web 2.0, detectar vulnerabilidades y simular como si fuera el atacante real, al final se genera un reporte que puede ser utilizado para corregir estas vulnerabilidades.

Abstract

This thesis presents an assessment for security in an existing government organization with real projects and real information about threats and/or vulnerabilities that may present in a controlled test environment, for this, we will use OWASP's SAMM Framework to evaluate the organization and gather information on where the organization stands in security matter, due the large extension of the framework, we will focus on the "Verification" module that is sub divided into three smaller modules, "Design Review, Implementation Review and Security Testing".

After the theoretical analysis, we will put it to practice searching for threats or vulnerabilities for Applications with Web 2.0 Technology using OWASP Top 10 project and getting from it, three vulnerabilities that we consider are the most critical, most attacks and/or more destructive or impact to the organization, these vulnerabilities that we will be thoroughly reviewed are, A1 Injection, A3 Cross-Site Scripting or XSS and A10 Under protected APIs (OWASP, 2017a).

Taking into consideration the information presented by OWASP, a test laboratory was built in INEGI where we analyzed their Applications with Web 2.0 Technology from past years 2015 and 2016 built using either Oracle Java or Microsoft .Net Technologies, this laboratory was set to further analyze and try to confirm the vulnerabilities found using OWASP open source tool called Zed Attack Proxy or "ZAP", this tool helped us run a full scan in all Applications with Web 2.0 Technology, detect vulnerabilities and simulate as if we were a real life attacker, at the end a report is generated that may be used later on to correct these vulnerabilities.

1. Introducción

1.1. Presentación del trabajo práctico

La innovación del software ha liberado una revolución de los datos sin precedentes en la historia, impulsada gracias a la abundancia de datos pero además por las tecnologías que cambian la manera en que reunimos, almacenamos, analizamos y transformamos la información. Hoy en día los datos son otro recurso clave que se pone a funcionar y del que se obtiene muchísimo conocimiento, dicho conocimiento puesto en práctica permite diseñar estrategias de alto valor que distingan al negocio.

El triángulo de Integridad, Disponibilidad y Confidencialidad, mejor conocida como la tríada CIA (López, 2008), expone las características que deben poseer los datos e información. La falta de alguna de estas puede causar altos impactos en cuestiones monetarias, legales y de credibilidad. Por lo que, siendo un activo tan importante, debe protegerse de su destrucción, uso incorrecto y/o acceso no autorizado.

A la par de esta revolución de los datos, dado en parte por el interés de los usuarios de utilizar nuevas y mejores aplicaciones que pongan a su disposición los recursos y bondades del uso de las nuevas tecnologías y por otra parte por la utilización de metodologías agiles que se adaptan a los cambiantes requisitos de los proyectos y reducen los tiempos de desarrollo, obteniendo así productos rápidamente y con un alto nivel de calidad.

El autor (Betancur Cartagena, 2016) expone que no existe una solución concreta en cuanto a la seguridad de la información, según la norma ISO 27002, por lo que en ocasiones la calidad requerida no es la calidad conseguida. Cada vulnerabilidad de las aplicaciones web, puede ser explotada por atacantes, los que utilizan diferentes rutas para dañar el negocio y la organización. Dado los diferentes métodos de explotar las vulnerabilidades, la eliminación del riesgo causado por el ataque presenta en ocasiones alta complejidad e impacto.

En el artículo Fortaleciendo un enfoque DevOps con Seguridad y Gestión de Riesgos: una experiencia de su implementación en un Centro de Datos en una organización Mexicana el autor expone una evolución del tradicional DevOps para incluir Seguridad y manejo de riesgos en un enfoque mejorado que cubra las necesidades actuales y demandantes de los proyectos que impacto critico que en forma operativa necesitan estar activos 24 horas y los 365 días del año, de esta manera se incluye en el nuevo enfoque una nueva fase que es Seguridad Informática, sumando cuatro fases en total (Ingeniería de Software, Control de Calidad, Infraestructura Tecnológica para la Operación y Seguridad Informática), todo esto dentro del marco de Gestión de Riesgos. (Muñoz & Díaz, 2017)

Cabe mencionar que en la fase de Control de Calidad se analizan los cuatro grandes procesos que toma el marco de trabajo SAMM de OWASP y el módulo de Verificación que será analizado más a fondo en este trabajo. (Muñoz & Díaz, 2017)

En la fase de Seguridad Informática se toma en cuenta todas las actividades de diseño, procedimientos, normas y técnicas que ayudan a generar un producto de calidad más seguro y confiable y que tenga en sus pilares la triada de la información (Confidencialidad, Integridad y Disponibilidad). (Muñoz & Díaz, 2017)

Sin embargo, la fundación con nombre Proyecto Abierto de Seguridad en Aplicaciones Web, (OWASP por sus siglas en inglés), ha creado un marco de trabajo que ayude a las organizaciones a formular e implementar estrategias que les permita eliminar el riesgo causado por vulnerabilidades en sus aplicaciones. Dentro de este marco de trabajo, se crea el estándar de seguridad OWASP Top 10 que tiene como objetivo concientizar sobre los problemas de seguridad en el software (OWASP, 2001).

Es importante señalar que existen varias técnicas para realizar un análisis de seguridad en las aplicaciones web, por mencionar algunas: Escáner de Vulnerabilidad, Escaneo de Seguridad, Pruebas de Penetración, Hackeo Ético, Evaluación de Riesgos, Auditorias de Seguridad, Cracking de Contraseñas, Evaluación de la Postura y Pruebas de Seguridad. Estas técnicas pueden ser realizadas en herramientas como: Nessus, Checkmarx, Microfocus Fortify Web Inspect y OWASP ZAP.

En este caso práctico se ha decidido utilizar las técnicas de Evaluación de la Postura y Pruebas de Seguridad llevadas a cabo con la herramienta OWASP ZAP. Con el objetivo de analizar diferentes aplicaciones con tecnología web 2.0 y dar a conocer el estado de la seguridad de las mismas, identificando vulnerabilidades y áreas de mejora.

1.2. Organización del documento

El documento se divide en siete capítulos como se describe a continuación:

1. Introducción: Se presenta un breve resumen del tema práctico, el objetivo principal y la organización del documento.

2. Planteamiento de la problemática a atender a través del trabajo práctico: se habla de los antecedentes y el problema existente, justificación y el sector o población que es afectado y beneficiado por la solución.

3. Fundamentación Teórica: Se brinda una imagen de la actualidad en materia de seguridad, las teorías y marco de trabajo a utilizarse OWASP Top 10, OWASP SAMM y OWASP ZAP para la correcta solución al problema, aunado a la revisión literaria y casos similares para comparación.

4. Diseño de la intervención al caso problema: Se describe el proceso a seguir, las actividades y beneficios del análisis de las aplicaciones web 2.0 del cliente.

5. Resultados de la intervención: Se describen los resultados obtenidos tras la intervención en sus distintas etapas.

6. Evaluación de la intervención: Se realiza una valoración por comparación entre los objetivos propuestos y los alcanzados, los problemas encontrados durante la implementación, los beneficios obtenidos como resultado de la intervención y las recomendaciones finales al cliente.

7. Conclusiones: se comentan los resultados obtenidos tras la intervención, el proceso, mejoras, introspectiva y retrospectiva y argumentos finales.

2. Planteamiento de la problemática a atender a través del trabajo práctico

2.1. Antecedentes

El organismo público autónomo de la República de Mexicana, Instituto Nacional de Estadística y Geografía (INEGI), es una institución que “genera información sobre fenómenos demográficos, sociales, económicos y del medio ambiente y su relación con el territorio nacional”. Los datos recopilados por INEGI y la información que procesa son estrictamente confidenciales, según lo que se establece en la Ley Mexicana (INEGI, 1985a).

La institución tiene una coordinación estatal en los 32 estados federativos de la República Mexicana y cuenta con Oficinas Centrales en el estado de Aguascalientes, en estas oficinas trabaja la Coordinación General de Informática, que tiene en su plantilla al Grupo de Ingeniería en Sistemas con la responsabilidad de detectar, verificar, documentar y dar seguimiento soluciones para las eventualidades de los servicios y productos que están hospedados en los “Centros de Datos”, a nivel nacional.

Para la difusión de información estadística y geografía, se desarrollan diferentes productos impresos y digitales (INEGI, 1985b), sin embargo, para el cumplir con el objetivo de este trabajo estarán contempladas solo los proyectos y servicios que tienen implementados en sus procesos automatizados la tecnología web 2.0 con base en Oracle Java y Microsoft .Net, debido a temas de confidencialidad no es factible presentar reporte a detalle (con datos sensibles) de todas las aplicaciones - proyectos existentes en el instituto, por lo tanto se analizará solamente una población de doscientas aplicaciones de los dos años 2015 y 2016.

Hoy en día en cualquier organización de cualquier ramo como educación, gobierno, finanzas, salud, telecomunicaciones, energía, defensa, entre otros tipos, utiliza en el día a día algún tipo de software, siendo las predominantes

aplicaciones con tecnología Web. De tal manera que la información y datos que INEGI obtiene, manipula, analiza y el conocimiento que genera es de vital importancia para la organización, esta información va de la mano con el crecimiento de la misma, le da su identidad y reputación a, por lo tanto el buen manejo de esta es indispensable.

A pesar de los grandes avances en la tecnología, aún se presentan amenazas de seguridad que dependiendo del grado de criticidad pueden afectar directamente a la organización financieramente. Entre más avanzadas son las aplicaciones en la organización se va volviendo más difícil lograr un determinado nivel de seguridad aunado a la importancia y el riesgo potencial de las vulnerabilidades que no son encontradas.

Es por lo mencionado anteriormente que debemos tomar en cuenta las distintas amenazas que existen y podrían afectar a la organización que se analizará para este trabajo practico el proyecto Top 10 de la Organización OWASP que se encarga de generar conciencia en desarrolladores y el área administrativa de cada proyecto u organización, el cual se ha convertido no solo en una guía sino además en un referente básico en la industria para cualquier organización o empresa.

El objetivo principal del proyecto OWASP Top 10 es aumentar la conciencia acerca de la seguridad en las aplicaciones al identificar algunos de los riesgos más críticos que enfrentan las organizaciones. Este proyecto se ayuda de diversos estándares, libros, herramientas y organizaciones como MITRE, PCI, DSS, DISA, FTC y muchas más. El proyecto OWASP Top 10 fue liberado por primera vez en el año 2003, se hicieron actualizaciones menores en 2004 y 2007. La versión de 2010 fue renovada para priorizar no solo por riesgo sino también prevalencia y han seguido utilizando este patrón desde entonces para la versión de 2013 y 2017.

En la siguiente tabla (Tabla 1), se presenta una comparación sobre distintas herramientas disponibles en el mercado y algunas de sus características, esta tabla fue tomada en cuenta para tomar la decisión sobre que herramienta sería la indicada para este trabajo practico además de ser la indicada para el análisis a la organización.

Tabla 1. Comparación herramientas para Seguridad en el Mercado.

Característica	ZAP	Nessus	NMap	Checkmarx	Microfocus (HP) Web Inspect
Escaneo URL	✓	✓	✓	✓	✓
Código Abierto	✓		✓		
Cumplimiento Auditorías		✓		✓	✓
Línea de Comandos (Consola)	✓	✓	✓		
Documentación	✓	✓	✓	✓	✓
Soporte	Comunidad	Profesional	Comunidad	Profesional	Profesional
Multi-Plataforma	✓	✓	✓	Multi-Lenguaje	
Reportes	✓	✓	✓	✓	✓

Además de esto, podemos observar en la siguiente imagen (Imagen 1) el cuadrante mágico de Gartner que presenta las distintas alternativas para Pruebas de Seguridad en Aplicaciones analizadas hasta febrero del año 2018, con esto nos podemos dar una idea de que tan confiable es la herramienta que utilizaremos, el organismo OWASP no aparece en el cuadrante mágico de Gartner ya que es una organización sin fines de lucro.

Imagen 1. Cuadrante Mágico de Gartner,para Pruebas de Seguridad en Aplicaciones, Marzo 2018.

El sitio Vulnerabilidades y Exposiciones Comunes (CVE, por sus siglas en inglés) que es arbitrado por MITRE, genera una lista de los ataques más comunes públicamente en el área de vulnerabilidades de ciberseguridad.

A cada registro se le asigna un valor por medio de Autoridades Numeradoras de CVE (CNAs, por sus siglas en inglés) alrededor del mundo para dar confianza a todos los involucrados al momento de discutir sobre un software en particular o vulnerabilidad, esto brinda una base para la evaluación de

herramientas, vendedores u organizaciones y hace más fácil compartir información para la implementación de automatización de ciber seguridad.

CVE tiene las siguientes características:

- Es un identificador para cada vulnerabilidad o exposición.
- Contiene una descripción estandarizada para cada vulnerabilidad o exposición.
- Es más un diccionario que una base de datos.
- Es una forma estándar para que las bases de datos y herramientas puedan hablar el mismo idioma.
- Brinda una forma de interoperabilidad y mejor cobertura de seguridad.
- Brinda una base para la evaluación de servicios, herramientas y bases de datos.
- Herramienta gratuita para el uso público.
- Enfocada a la industria a través de CNAs, la comisión CVE y numerosos productos y servicios que incluyen CVE.

¿Por qué CVE?: CVE fue lanzada en 1999 cuando la mayoría de las herramientas para ciberataques usaban su propia base de datos con sus propios nombres, lo cual generaba una variación significante entre los distintos productos y no existía una manera fácil de hacer referencia al mismo problema. Esto generaba riesgos potenciales y grandes brechas al momento de hacer la cobertura en seguridad y poca o nula interoperabilidad entre herramientas y bases de datos, además de que cada organización tenía sus propias métricas lo cual deriva en la falta de estandarización.

El sitio CVE genera una tabla histórica con los ataques a nivel mundial, divida por año y tipo de ataque como podemos ver a continuación (Imagen 2).

Vulnerabilities By Type

Year	# of Vulnerabilities	DoS	Code Execution	Overflow	Memory Corruption	Sql Injection	XSS	Directory Traversal	Http Response Splitting	Bypass something	Gain Information	Gain Privileges	CSRF	File Inclusion	# of exploits
1999	894	177	112	172			2	7		25	16	103			2
2000	1020	257	208	206		2	4	20		48	19	139			
2001	1677	403	403	297		7	34	123		83	36	220		2	2
2002	2156	498	553	435	2	41	200	103		127	74	199	2	14	1
2003	1527	381	477	371	2	49	129	60	1	62	69	144		16	5
2004	2451	580	614	410	3	148	291	110	12	145	96	134	5	38	5
2005	4935	838	1627	657	21	604	786	202	15	289	261	221	11	100	15
2006	6610	893	2719	663	91	967	1302	322	8	267	271	184	18	849	30
2007	6520	1101	2601	953	95	706	884	339	14	267	323	242	69	700	44
2008	5632	894	2310	699	128	1101	807	363	7	288	270	188	83	170	74
2009	5736	1035	2185	700	188	963	851	322	9	337	302	223	115	138	738
2010	4652	1102	1714	680	342	520	605	275	8	234	282	238	86	73	1493
2011	4155	1221	1334	770	351	294	467	108	7	197	409	206	58	17	557
2012	5297	1425	1459	843	423	243	758	122	13	343	389	250	166	14	624
2013	5191	1454	1186	859	366	156	650	110	7	352	511	274	123	1	205
2014	7946	1598	1574	850	420	305	1105	204	12	457	2104	239	264	2	401
2015	6480	1791	1825	1079	749	217	776	149	12	577	748	367	248	5	127
2016	6447	2028	1494	1326	717	94	497	99	15	444	843	600	87	7	1
2017	14712	3154	3004	2805	745	503	1515	274	11	629	1706	459	327	18	6
2018	5647	817	1055	678	171	191	701	115	5	270	478	110	158	9	4
Total	99685	21647	28454	15453	4814	7111	12364	3427	156	5441	9207	4740	1820	2173	4334
% Of All		21.7	28.5	15.5	4.8	7.1	12.4	3.4	0.2	5.5	9.2	4.8	1.8	2.2	

Imagen 2. Total de Ataques/Vulnerabilidades por Tipo y Año por la CVE.

Tabla 2. Total de Vulnerabilidades por Año y Tipo del Sitio CVE Details.

Año	# de Vulnerabilidades	DoS	Ejecución de Código	Desborde	Corrupción de Memoria	Inyección SQL	XSS	Cruce de Directorio	División de Respuesta HTTP	Sobrepases	Obtención de Información	Obtención de Privilegios	CSRF	Inclusión de Archivos	# de explotaciones
2015	8670	1791	1825	1079	749	217	776	149	12	577	748	367	248	5	127
2016	8252	2028	1494	1326	717	94	497	99	15	444	843	600	87	7	1
Total	**16922**	**3819**	**3319**	**2405**	**1466**	**311**	**1273**	**248**	**27**	**1021**	**1591**	**967**	**335**	**12**	**128**
% de Todos		22.6	19.6	14.2	8.7	1.8	7.5	1.5	0.2	6.0	9.4	5.7	2.0	0.1	0.8

Así mismo, la Imagen 1 y la Tabla 2 muestran el resumen del análisis histórico para los años 2015 y 2016 a nivel mundial, dividido por año y por tipo de ataque de todas las vulnerabilidades registradas.

Como podemos observar en la tabla presentada en la Imagen 1, el año 2017 fue el año con mayor número total de ataques a nivel mundial con 14,712 ataques registrados, de los cuales 503 son de Inyección (SQL Injection) y 1,515 de Secuencia de Comandos en Sitios Cruzados (Cross-Site Scripting o XSS) que son los que tomaremos en cuenta para este estudio, en análisis, los ataques totales del año 2017 representan un aumento del 56% en comparación con los 6,447 registrados en 2016, de igual manera, el número de vulnerabilidades de tipo Inyección en 2016 fue de 94 contra 503 en 2017 y de XSS 497 en 2016 contra 1,515 en 2017, lo cual representa un aumento de 81% y 67% respectivamente.

Alcance (Scope)

La evaluación de las aplicaciones web 2.0 creadas en Oracle Java y Microsoft .NET de INEGI se realizará con la herramienta gratuita de OWASP Zed Attack Proxy (ZAP, por sus siglas en inglés), con la cual se analizarán en busca de vulnerabilidades.

Debido a la extensa lista de vulnerabilidades que existen y ponen en riesgo las aplicaciones con tecnología web 2.0, se tomará en cuenta solamente y basados en la versión 2017 del proyecto de OWASP Top 10 las siguientes tomando en cuenta la importancia/impacto de las mismas al ser las dos primeras vulnerabilidades (A1 Inyección y A7 XSS Tomadas de la versión final) las que tienen mayor incidencia en la organización y la A10 APIs no Protegidas (Tomada de la versión Candidata a Liberación 1 de 2017 que posteriormente cambia su nombre a Registro y Monitoreo Insuficientes en la versión final) por ser nueva y tener potencialmente el mismo nivel de peligrosidad que las dos primeras:

A1 Inyección: El tipo de vulnerabilidad o fallas de inyección, como son los distintos tipos SQL, No SQL, OS o LDAP se presentan cuando el atacante envía datos no confiables a un intérprete en la aplicación, como parte de un comando o consulta mediante un campo que no valida el tipo de entrada. Los datos dañinos del atacante pueden engañar al intérprete para que ejecute comandos involuntarios o acceda a los datos sin la debida autorización.

A3 Secuencia de Comandos en Sitios Cruzados (XSS): Los ataques de tipo Cross-Site Scripting (por sus siglas en inglés) ocurren cuando una aplicación toma datos no confiables provenientes de un script o código malicioso inyectado por el atacante y los envía al navegador web sin una validación y codificación apropiada; También se puede presentar otro caso en el que se actualiza una página web existente con la ayuda de datos reales que fueron suministrados por el usuario engañado utilizando una API que ejecuta JavaScript en el navegador. Este tipo de ataques permiten ejecutar comandos en el navegador de la víctima y el atacante puede secuestrar una sesión, modificar los sitios web, o re direccionar al usuario hacia un sitio malicioso con el fin de obtener información valiosa real mientras el usuario no se da cuenta del sitio apócrifo (OWASP, 2017a).

A10 APIs no Protegidas: Este tipo de ataques se presenta cuando al estar desarrollando una aplicación se utilizan APIs no seguras, públicas o sin validar y que contienen vulnerabilidades, el atacante al darse cuenta de esta vulnerabilidad puede aplicar ingeniería inversa para obtener información y acceso completo a la aplicación o en el peor de los casos combinar con el ataque A1 e inyectar código malicioso tal cual fuera una aplicación normal (OWASP, 2017a).

2.2. Diagnóstico

El presente caso práctico realiza un análisis completo, de algunas de las aplicaciones con tecnología web 2.0 con que cuenta actualmente el INEGI. Este

análisis permite conocer el estado en que se encuentra el proyecto de la institución en materia de seguridad tomando en cuenta el marco de trabajo OWASP SAMM. Aunado a esto, se implementarán técnicas de Evaluación de la Postura y Pruebas de Seguridad llevadas a cabo con la herramienta OWASP ZAP para escanear las aplicaciones con tecnología web 2.0 en busca de vulnerabilidades y/o posibles amenazas.

Además de las estadísticas descritas anteriormente por la CVE, existe otro organismo que genera esta herramienta que representa de manera más visual y en tiempo real, la distribución de los ataques a nivel mundial. NORSE – Superior Attack Intelligence, mantiene la red dedicada a amenazas más grande del mundo con más de 8 millones de sensores que emulan más de 6 mil aplicaciones desde laptops y cajeros ATM hasta sistemas críticos de infraestructura y Circuitos Cerrados de Cámaras de Seguridad, la red de inteligencia NORSE reúne información sobre los atacantes y sus objetivos

Mapa de Ataques Norse (Norse Attack Map)

NORSE recibe al instante la telemetría del ataque de los más de 8 millones de sensores desplegados a lo largo del mundo, opera la base de datos comercial de inteligencia de ataques con más de 7 peta bytes de ataques detallados, los sensores y mieleros pueden emular más de 6 mil dispositivos y aplicaciones atacados comúnmente. La siguiente imagen (Imagen 3), muestra los ataques en tiempo real en el mundo.

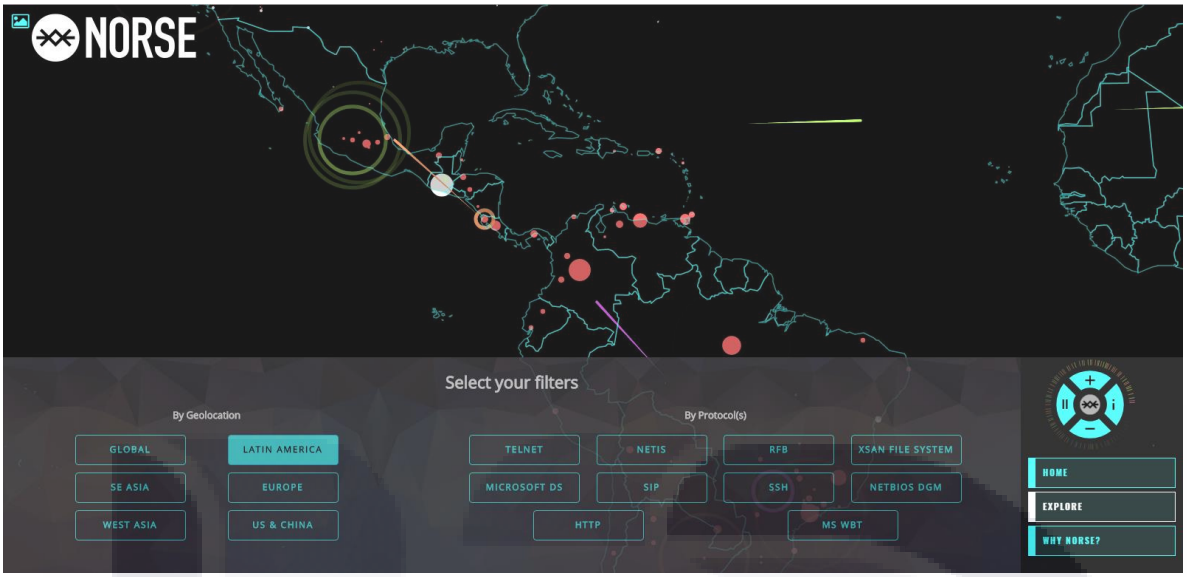


Imagen 3. Mapa NORSE de ataques en tiempo real en Latino América ("Norse Attack Map").

Como ya se mencionó anteriormente, para evitar problemas de confidencialidad de la información, el Instituto solo puede brindar información no actual para realizar el diagnóstico. Es decir, se pueden utilizar solamente datos y proyectos de años pasados, en este caso usaremos 2015 y 2016 para realizar el análisis. Además de persistir la política de información sensible, no se puede dar a conocer nombres reales de aplicaciones, direcciones URL, cantidad de vulnerabilidades o ataques por proyecto etc., se dejará de manera muy general toda esta información que será utilizada.

2.3. Justificación de intervención.

Dada la privacidad de los datos que procesa y difunde INEGI, es indispensable que mantengan la Integridad, Disponibilidad y Confidencialidad como características fundamentales. Al presentar un ataque de cualquier tipo se perdería la credibilidad al usuario final.

Por lo que el Grupo de Ingeniería en Sistemas de INEGI cuente con un análisis y reporte de seguridad para identificar vulnerabilidades en áreas de mejora en las aplicaciones – proyectos con tecnología web 2.0 que desarrollan, o

adquieren por la institución. Obteniendo diferentes ventajas en temas de seguridad como son principalmente:

- Ayuda a prevenir ataques y mitiga riesgos, identificando y/o solucionando vulnerabilidades.
- Reducción de costos al encontrar, en los inicios ciclo de vida de desarrollo del software, comportamientos no deseados y/o errores en las aplicaciones desarrolladas.
- Construcción de software de calidad.
- Aumenta la reputación del negocio.
- Debido a la limitante del tiempo y la gran área que abarca cada vulnerabilidad solo se van a tomar en cuenta tres tipos que son los que más incidencias tienen en la organización (Inyección, XSS y APIs no protegidas). Usaremos la herramienta gratuita OWASP ZAP (Zed Attack Proxy) ya que al ser de código abierto el instituto no debe destinar presupuesto para la obtención de una herramienta además que se adecua a las necesidades del mismo.

2.4. Organización, cliente y usuarios principales del trabajo práctico.

La organización principal para la cual se realiza este trabajo práctico es INEGI. Como cliente principal dentro de la organización está la Coordinación General de Informática y como usuario principal se tiene al Grupo de Ingeniería en Sistemas.

De forma indirecta son clientes del caso práctico, aquellas instituciones que revisan, referencian y utilizan la información que es difundida por INEGI, mediante aplicaciones con tecnología web 2.0.

3. Objetivos, general y específicos

3.1. Objetivo general del trabajo práctico

Evaluar las aplicaciones con tecnología Web 2.0 construidas en Oracle Java y Microsoft .NET por el Instituto Nacional de Estadística Geografía, con la finalidad de identificar vulnerabilidades y áreas de mejora en el sector de seguridad mediante el uso de una herramienta gratuita OWASP ZAP (Zed Attack Proxy).

3.2. Objetivos específicos de la intervención en el caso problema

1. Analizar las aplicaciones Web 2.0 construidas en Java y .NET en búsqueda de las tres vulnerabilidades: A1- Inyección; A7- Secuencia de comandos en sitios cruzados (XSS por sus siglas en inglés) y A10- Registro y Monitoreo Insuficientes (OWASP, 2017b).
2. Presentar el reporte de OWASP ZAP sobre las aplicaciones evaluadas.
3. Presentar el reporte final de la evaluación realizada a la organización.
4. Presentar resultados y sugerencias a la compañía para mejorar la seguridad en aplicaciones Web 2.0.

3.3. Argumentos

1. El uso de las herramientas gratuitas OWASP ZAP (Zed Attack Proxy) permite identificar las vulnerabilidades de las aplicaciones web 2.0
2. El uso de las herramientas gratuitas OWASP ZAP (Zed Attack Proxy) permite disminuir las vulnerabilidades de las aplicaciones web 2.0
3. El uso de las herramientas gratuitas OWASP ZAP (Zed Attack Proxy) permite recomendar estrategias tendientes a disminuir y/ evitar las vulnerabilidades de las aplicaciones web 2.0

4. Fundamentación Teórica

4.1. OWASP TOP 10

OWASP Top 10 es un estándar de seguridad, creado por la fundación OWASP. En sus inicios tuvo como objetivo concientizar a las organizaciones sobre las vulnerabilidades de seguridad en sus procesos de desarrollo de software, sin embargo en la actualidad es una seria de sugerencias sobre temas de seguridad en aplicaciones para desarrolladores, ingenieros de pruebas y gerentes. El estándar se construye gracias al consenso de expertos en seguridad de todo el mundo, acerca de los riesgos de seguridad más críticos para las aplicaciones web (OWASP, 2017).

El proyecto Top 10 es referenciado por muchos estándares, libros, herramientas y organizaciones incluyendo MITRE (Organización americana sin fines de lucro que administra centros de desarrollo e investigación financiados por el gobierno federal y que apoyan a varias agencias del gobierno estadounidense), PCI (Estándar de seguridad de información relacionado con Tarjetas de crédito por sus siglas en inglés Payment Card Industry Data Security Estándar PCI-DSS), DISA (Se puede referir a Data Interchange Standards Association, Asociación de Estándares para Intercambio de Datos, responsable de crear diferentes estándares usados en la industria como EDI – Electronic Data Interchange, DISA dejó de existir en enero de 2016, Defense Information Systems Agency que es básicamente una agencia que forma parte del Departamento de Defensa de Estados Unidos y se encarga de proveer apoyo a las comunicaciones y Tecnologías de Información para el presidente, vicepresidente, secretario de defensa, servicios militares, comandos combatientes y cualquier individuo o sistema que contribuya a la defensa de los Estados Unidos), Comisión Federal de Comercio (por sus siglas en inglés, FTC) y muchos más. La primera liberación del proyecto se realizó en 2003, con actualizaciones menores en 2004 y 2007. Estas versiones del proyecto, eran priorizadas por prevalencia. Sin

embargo en el 2010, se incluye la prioridad por riesgo, este patrón es utilizado en las siguientes versiones.

La lista que se genera a partir del proyecto OWASP Top 10, contiene diferentes tipos de ataques, sin embargo en este caso práctico solamente tomaremos en cuenta los ataques: A1 Inyección, A7 Secuencia de Comandos en Sitios Cruzados y A10 Registro y Monitoreo Insuficientes (OWASP, 2017b).

4.1.1. A1 Inyección.

La organización que lleva como nombre Vulnerabilidades y Exposiciones Comunes (por sus siglas en ingles CVE) (CVE, 2018), en su sitio web expone graficas que muestran las vulnerabilidades por tipo y año. La Imagen 4 muestra específicamente las vulnerabilidades de inyección SQL por año.

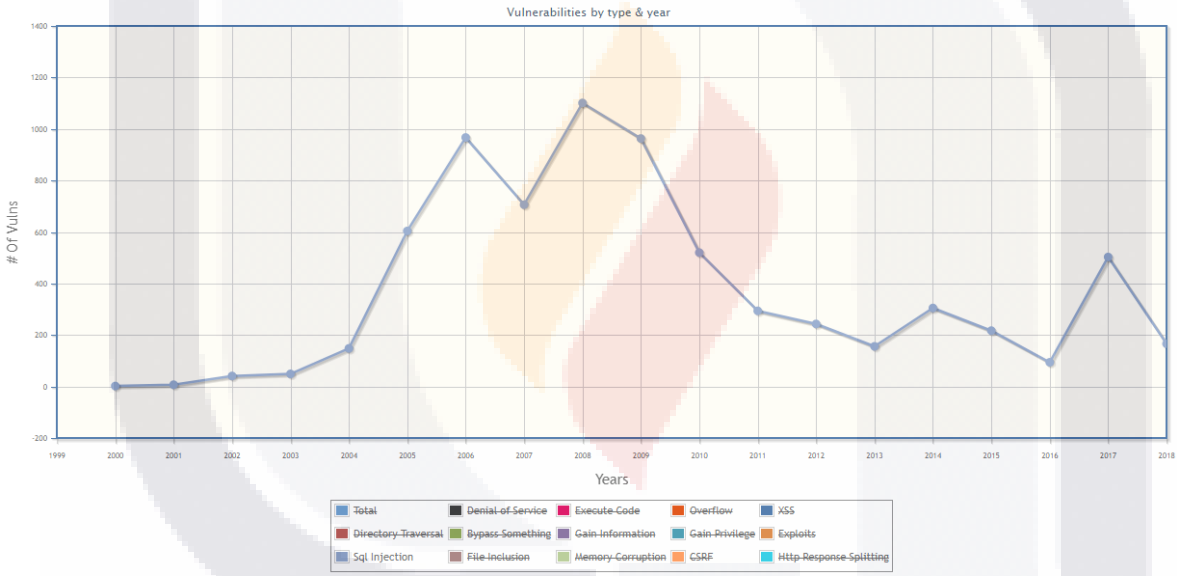


Imagen 4. Gráfico de la CVE de cantidad de ataques por Inyección SQL por año.

Según la CVE la prevalencia de inyección SQL a nivel mundial para el año 2017, fue de 503 ataques registrados mientras que en el año 2016 solo se registraron 94 ataques. Lo cual representa un incremento significativo del 535% en doce meses.

Agentes de Amenaza: Para este tipo de vulnerabilidad tenemos que considerar cualquier persona que sea capaz de enviar datos no confiables al sistema, estas incluyen, usuarios externos, socios de negocio, otros sistemas, usuarios internos y/o administradores, básicamente cualquier persona que tenga acceso a la aplicación y detecte un campo sin validación propia puede explotar esta vulnerabilidad con un poco de conocimiento de comandos de consultas.

Vectores de Ataque: Con una explotabilidad fácil, este tipo de vulnerabilidad se convierte en la más peligrosa, los atacantes pueden enviar comandos sencillos basados en un simple texto, este comando explota la vulnerabilidad en los campos no validados mediante la sintaxis del intérprete objetivo. Prácticamente cualquier fuente de datos puede usarse como un vector de inyección, incluyendo las fuentes internas.

Debilidad de Seguridad: Con una prevalencia común y una defectibilidad promedio, los fallos por inyección se presentan cuando una aplicación por medio de un campo no validado o fuente de datos, envían información no confiable a un intérprete. Los fallos por inyección son muy prevalentes, particularmente en el código de aplicaciones heredadas o proyectos inactivos. Se encuentran frecuentemente en consultas SQL, LDAP, XPath y/o No SQL, comandos de Sistema Operativo, Parsers o Analizadores XML, Encabezados SMTP, lenguajes de expresión, etc. Los fallos por inyección son fáciles de encontrar cuando se examina el código, pero difíciles cuando se hace por medio de pruebas (testing). Los escáneres y fuzzers pueden ayudar a encontrar fallos de este tipo.

Impacto Técnico: Con un impacto severo, este tipo de vulnerabilidad es la peor pesadilla para las aplicaciones Web 2.0, la inyección puede resultar en perdida de datos, corrupción de la información, faltantes de contabilidad o negación del servicio (Denial of Service por sus siglas en inglés DoS) y a veces puede llevar a la perdida completa de control sobre el host o anfitrión.

Impacto al Negocio: Con impacto directo a la aplicación o al negocio, hay que considerar el valor de los datos afectados y la plataforma que ejecuta el intérprete. Toda la información puede ser robada, modificada o borrada. Además la reputación de la organización puede verse comprometida.

¿Cómo saber si soy vulnerable al ataque por Inyección?

La mejor forma de averiguar si una aplicación es vulnerable a inyección es verificar que todos los intérpretes separen claramente los datos no confiables del comando o consulta. En muchos casos se recomienda evadir el intérprete o deshabilitarlo (por ejemplo XXE) si es posible. Para las llamadas a SQL, se necesita hacer uso de variables de enlace en todas las declaraciones preparadas y procedimientos almacenados o stored procedures (SP) y/o evitar el uso de consultas dinámicas.

Validar el código es la forma más rápida y precisa de verificar que la aplicación use los intérpretes de forma segura. El analista de seguridad se puede ayudar de herramientas de análisis de código para el uso de los intérpretes y rastrear el flujo de información a lo largo de la aplicación. Los Ingenieros de pruebas de penetración pueden ayudar a validar estos problemas creando explotables que confirmen la existencia de la vulnerabilidad en la aplicación.

Los escáneres dinámicos automatizados pueden ayudar a dar pistas sobre la existencia de inyecciones explotables en la aplicación. Cabe mencionar que los escáneres no siempre llegan hasta el intérprete y pueden tener dificultades para saber si un ataque fue exitoso. El mal manejo de errores y excepciones puede conducir a los atacantes a descubrir vulnerabilidades del tipo inyección en la aplicación.

¿Cómo puedo prevenir la Inyección?

Primordialmente, para prevenir este tipo de ataques se debe mantener separada la información no confiable de los comandos y/o consultas.

1. La mejor opción es usar una API segura que evite el uso del intérprete completamente o la mayor parte del tiempo mientras sea posible o bien que tenga una interfaz parametrizada. Hay que tener especial cuidado con las APIs como Procedimientos Almacenados que aunque estén parametrizados puedan ejecutar la inyección.

2. Si no tenemos disponible una API parametrizada, se deben crear secuencias de validación de escape de caracteres usando la sintaxis específica para el intérprete usado, existen varias librerías que proveen estas rutinas de escape.

3. Crear una lista positiva o Lista Blanca (White List) con las validaciones de entrada, hay que tener cuidado ya que es recomendable pero no es la defensa completa ya que en muchas situaciones se requiere que se permitan caracteres especiales, si es el caso solo el primer y segundo punto pueden ayudar a su uso seguro.

Escenarios de Ataque (Ejemplos)

Escenario 1: una aplicación usa datos no confiables en la construcción de la siguiente consulta SQL:

*Consulta de Cadena = "SELECT * FROM cuentas WHERE clienteID='" + request.getParameter("id") + "'";*

Escenario 2: de manera similar, si una aplicación confía ciegamente en los marcos de trabajo, podría resultar en consultas que igual son vulnerables:

Consulta HQLQuery = session.createQuery("FROM cuentas

WHERE clienteID='" + request.getParameter("id") + "");

En ambos casos, el atacante modificó el valor del parámetro “id” en el navegador para enviar ‘or ‘1’=’1. Por ejemplo:

Http://example.com/app/accountView?id=' or '1'='1

Esto cambia completamente el significado de ambas consultas para que regresen todos los registros de la tabla de cuentas. Otros ataques más peligrosos pueden modificar datos o incluso invocar procedimientos almacenados (Stored Procedures)

4.1.2. A3 Secuencia de Comandos en Sitios Cruzados.

La Imagen 4 muestra específicamente las vulnerabilidades de Secuencia de Comandos en Sitios Cruzados por año.

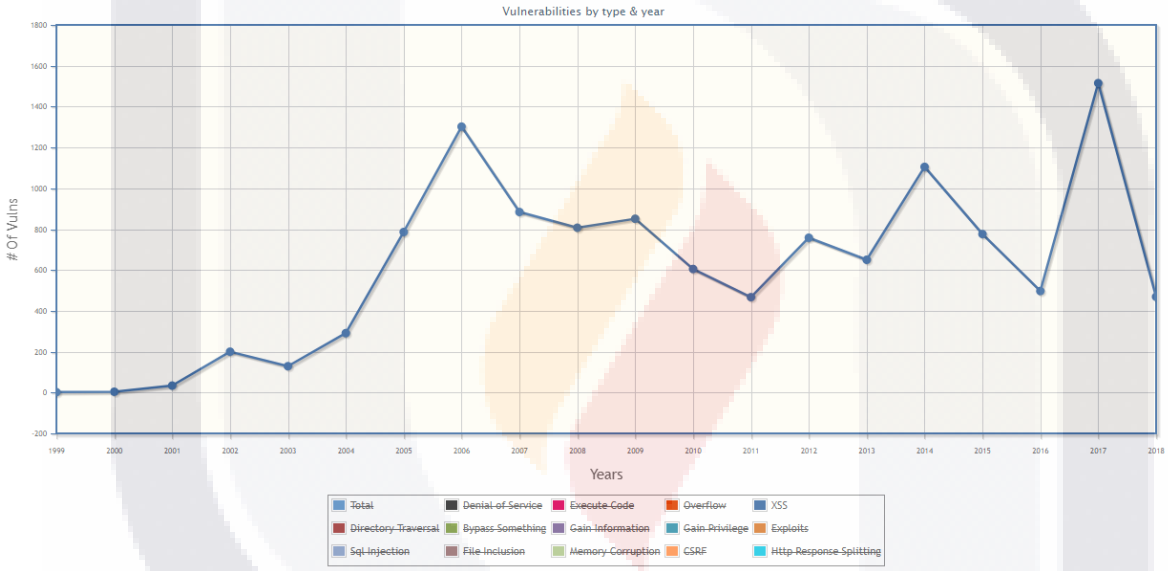


Imagen 5. Gráfico de la CVE de cantidad de ataques por Secuencia de Comandos en Sitios Cruzados por año.

Según la CVE la prevalencia de Secuencia de comandos en sitios cruzados a nivel mundial para el año 2017, fue de 1,515 ataques registrados mientras que en el año 2016 solo se registraron 497 ataques. Lo cual representa un incremento significativo de más del 304% en doce meses.

Agentes de Amenaza: Para este tipo de vulnerabilidad hay que considerar cualquier persona que sea capaz de enviar datos no confiables al sistema, estas incluyen, usuarios externos, socios de negocio, otros sistemas, usuarios internos y/o administradores, cualquier persona que tenga acceso a la aplicación y detecte un fallo en el navegador.

Vectores de Ataque: con una explotabilidad promedio, los atacantes pueden enviar comandos basados en texto o scripts que exploten la vulnerabilidad del intérprete en el navegador. Cualquier fuente de datos puede ser un vector de ataque, incluyendo fuentes internas como los datos de la base de datos.

Debilidad de Seguridad: con una prevalencia muy amplia y una defectibilidad promedio, los fallos XSS se presentan cuando una aplicación actualiza una página web con datos que son controlados por el atacante sin escapar propiamente de ese contenido o usar una API segura de JavaScript. Existen dos categorías de XSS, Almacenados que ocurren en el servidor y son fáciles de detectar por medio de pruebas o análisis de código y Reflejados que se presentan en el cliente y pueden ser muy difíciles de detectar.

Impacto Técnico: con un impacto moderado, los atacantes pueden ejecutar líneas de comando o scripts directamente en el navegador de la víctima, para secuestrar sesiones de usuario, desfasar sitios web, insertar contenido hostil, re direccionar a usuarios o secuestrar el navegador usando malware etc.

Impacto al Negocio: con impactos a la aplicación o específicos al negocio, tenemos que tomar en cuenta el valor para el negocio que representa el sistema afectado y todos los datos que procesa, también hay que considerar el impacto para el negocio si se expone públicamente la vulnerabilidad.

¿Cómo saber si soy vulnerable a la Secuencia de Comandos en Sitios Cruzados?

Podemos ser vulnerables a XSS Almacenados (Tipo Servidor) si el código del lado del servidor usa entradas provistas por el usuario como parte de la salida HTML y no se usan rutinas de escape sensitivas al contexto para asegurar que no se ejecuten. Si una página web usa JavaScript para agregar datos controlables para el atacante de forma dinámica, puede presentarse XSS Reflejado (Tipo Cliente). Generalmente podemos evitar esto usando APIs de JavaScript seguras, pero las validaciones de escape también pueden ser usadas.

Las herramientas automatizadas pueden ayudar a detectar XSS, sin embargo, cada aplicación construye páginas de salida de forma diferente y puede hacer uso de distintos intérpretes como JavaScript, ActiveX, Flash o Silverlight, usualmente usando librerías de terceros construidas arriba de estas tecnologías. Esta diversidad hace que la detección automatizada sea difícil, en especial cuando se usan aplicaciones modernas de una sola página, marcos de trabajo JavaScript y librerías. Por lo tanto para una cobertura completa se requiere una combinación de revisiones manuales de código y pruebas de penetración aunados al enfoque automatizado.

¿Cómo puedo prevenir la Secuencia de Comandos en Sitios Cruzados?

Para prevenir ataques XSS debemos separar los datos no confiables del contenido activo del navegador.

1. Para evitar XSS Almacenado (Tipo Servidor), la mejor opción es tener secuencias de escape para datos no confiables basados en el contexto de HTML (Cuerpo o Body, Atributos, JavaScript, CSS o la URL) en donde se pondrán los datos.

2. Para evitar XSS Reflejado (Tipo Cliente), la mejor opción es evitar el paso de datos no confiables a JavaScript y otras APIs de navegador que puedan generar contenido activo. Cuando esto no se pueda evitar, se

necesitan secuencias de escape similares para aplicarlas en las APIs del navegador.

3. Para contenido enriquecido hay que considerar librerías de auto sanitización como la herramienta de OWASP AntiSamy o el proyecto Sanitizador HTML de Java.

4. Considerar la Política de Contenido Seguro (Content Security Policy o CSP por sus siglas en inglés) para defenderse de XSS en todo el sitio web.

Escenarios de Ataque (Ejemplos)

La aplicación usa datos no confiables para la construcción del siguiente snippet HTML sin validación o secuencias de escape:

(String) page += "<input name='creditcard' type='TEXT' value='" + request.getParameter("TDC") + "'>";

El atacante puede modificar el parámetro Tarjeta de Crédito (TDC) en su navegador a lo siguiente:

'><script>document.location= 'http://www.attacker.com/cgi-bin/cookie.cgi?foo='+document.cookie</script>'.

Este ataque causa que el ID de sesión de la víctima sea enviado al sitio web del atacante, permitiéndole secuestrar la sesión actual.

4.1.3. A10 APIs no Protegidas (OWASP Top 10 RC1)

Hoy en día frecuentemente las aplicaciones hacen uso de APIs o de cliente enriquecido, como ejecutar JavaScript directo en el navegador o aplicaciones móviles que se conectan a una API de algún tipo (SOAP/XML, REST/JSON, RPC, GWT, etc.). La mayoría de las veces estas APIs no son controladas o

verificadas y están desprotegidas y pueden contener numerosas vulnerabilidades o ser usadas como una puerta de entrada.

Para analizar consideraremos lo siguiente:

Agentes de Amenaza: Para esta vulnerabilidad vamos a considerar a cualquiera que tenga la habilidad de enviar peticiones o requests a nuestras APIs. El software cliente puede ser fácilmente invertido y las comunicaciones interceptadas de igual manera, por lo tanto la "oscuridad" no es defensa para las APIs ya que la seguridad por oscuridad u ocultación implica el diseño o implementación en secreto para garantizar la seguridad como por ejemplo mantener el código fuente, algoritmos y protocolos en secreto y no revelar información sensible o vulnerabilidades encontradas.

Vectores de Ataque: Con una explotabilidad media, los atacantes pueden aplicar ingeniería inversa a las APIs al examinar el código del cliente o simplemente monitoreando las comunicaciones. Algunas vulnerabilidades de las APIs pueden ser detectadas automáticamente, otras solo pueden ser descubiertas por expertos.

Debilidad de Seguridad: Con una prevalencia común y detectabilidad difícil las aplicaciones modernas y las APIs se componen cada vez mas de clientes enriquecidos (navegador, móvil, escritorio) que se conectan a APIs traseras o backend (XML, Json, RPC, GWT, personalizadas). Las APIs que pueden estar compuestas por micro servicios, servicios o puntos finales (endpoints), pueden ser vulnerables a un gran rango de ataques. Desafortunadamente, las herramientas dinámicas e incluso las estáticas no funcionan bien con las APIs, por lo tanto se pueden volver difícil de analizar manualmente, debido a esto, este tipo de vulnerabilidades son frecuentemente pasadas por alto.

Impacto Técnico: con un impacto moderado, el rango completo de posibles resultados negativos es muy variado, incluyendo el robo, corrupción y/o

destrucción de datos, acceso no autorizado a la aplicación completa y/o control total del host o anfitrión.

Impacto al Negocio: con impactos a la aplicación o específicos al negocio, tenemos que considerar el impacto de negocio cuando se presenta un ataque a una API y tenemos que hacernos la siguiente pregunta: ¿La API tiene acceso a datos o funciones críticos? Muchas APIs son críticas para los proyectos, por lo tanto también hay que considerar el impacto si se produce un ataque de Negación de Servicio (DoS: Denial of Service por sus siglas en ingles).

¿Cómo saber si soy vulnerable a las APIs No Protegidas?

Se debe considerar incluir pruebas a las APIs de manera similar a como se prueba el resto del sistema. Los distintos tipos de inyección, autenticación, control de acceso, encriptación, configuración y otros tipos de problemas se pueden presentar de igual manera en las APIs así como se presentan en las aplicaciones tradicionales. Sin embargo, debido a que las APIs son construidas para que sean usadas por los mismos programas o aplicaciones y no por humanos, en la mayoría de las veces no hay una interfaz gráfica, usan protocolos y estructuras de datos complejos. Estos factores pueden dificultar las pruebas de seguridad, por lo cual se recomienda el uso de formatos que son ampliamente utilizados y que nos puedan ayudar, como Swagger (API abierta), REST, JSON y XML. Algunos marcos de trabajo como GWT y algunas implementaciones RPC usan formatos personalizados. Algunas aplicaciones y APIs crean sus propios protocolos y formatos de datos como WebSockets. La amplia variedad y complejidad de las APIs hacen que sea complicado aplicar efectivamente las pruebas de seguridad automatizadas, lo cual en muchos de los casos nos puede llevar a tener un falso sentido de seguridad.

Por último, se debe elegir una estrategia adecuada al proyecto u organización para descubrir si nuestras APIs son seguras y tratar de probar los módulos más críticos e importantes.

¿Cómo puedo prevenir APIs No Protegidas?

El factor clave para proteger nuestras APIs es asegurarnos de que entendemos el modelo de amenaza y las defensas con las que contamos.

1. Asegurar que las comunicaciones entre el cliente y la API son seguras.
2. Asegurar que se cuenta con un esquema de autenticación fuerte para las APIs y que todas las formas de acceso como credenciales, llaves y tokens han sido asegurados.
3. Asegurar que sea cual sea el formato que usan las peticiones, la configuración del parser o analizador esté endurecida o hardened contra el ataque.
4. Se debe implementar un esquema de control de accesos que brinde protección cuando las APIs sean llamadas o invocadas indebidamente, incluyendo funciones no autorizadas y referencias de datos.
5. Tomar en cuenta la protección contra todos los tipos de inyección ya que estos ataques son viables de manera normal y por medio de las APIs.
6. Asegurarse que los análisis de seguridad que se corren y las pruebas de seguridad cubran o contemplen todas las APIs y que las herramientas que se utilizan puedan analizarlas efectivamente.

Escenarios de Ataque (Ejemplos)

Escenario 1: imaginemos que una aplicación móvil de un banco se conecta a una API XML en el banco para obtener información de la cuenta y hacer transacciones. El atacante podría aplicar la ingeniería inversa a la aplicación y darse cuenta que el número de cuenta es enviado como parte de la petición o request como parte del método de autenticación junto con el usuario y contraseña hacia el servidor. El atacante podría enviar las credenciales

obtenidas las cuales son verdaderas pero con otro número de cuenta y obtener acceso a la cuenta de otro usuario.

Escenario 2: Tenemos una API pública que obtuvimos a través de una pequeña empresa o startup para enviar mensajes de texto automáticamente. La API acepta mensajes JSON que contienen un campo llamado “id_de_transacción”. La API parsea o analiza este campo como una variable de tipo cadena de texto o string y lo concatena a un comando SQL sin parametrizarlo, lo cual hace a la API vulnerable a inyecciones SQL al igual que si fuera una aplicación normal.

4.2. Marco de Trabajo OWASP SAMM

El modelo de Aseguramiento de la Madurez en el Software (SAMM por sus siglas en inglés) es un marco de trabajo abierto (gratuito y disponible para cualquiera) para ayudar a las organizaciones a formular e implementar una estrategia para la seguridad del software que esté adaptada a la medida de los riesgos específicos que enfrenta la organización(OWASP, 2011). De esta forma, SAMM apoya a:

- Evaluar las prácticas de seguridad de software existentes de una organización.
- Construir un programa balanceado de aseguramiento de la seguridad en el software.
- Demostrar mejorar concretas al programa de aseguramiento de la seguridad.
- Definir y medir las actividades relacionadas con la seguridad a lo largo de la organización.

En la siguiente Imagen (Imagen 6) podemos observar una vista general de los distintos módulos que comprenden el Marco de Trabajo de OWASP, Modelo de Aseguramiento de la Madurez por sus siglas en ingles SAMM, el cual contiene

cuatro grandes áreas o módulos denominados Funciones de Negocio que incluyen Gobernancia, Construcción, Verificación y Operaciones. A su vez estas áreas se dividen en pequeños sub módulos como podemos ver a continuación.

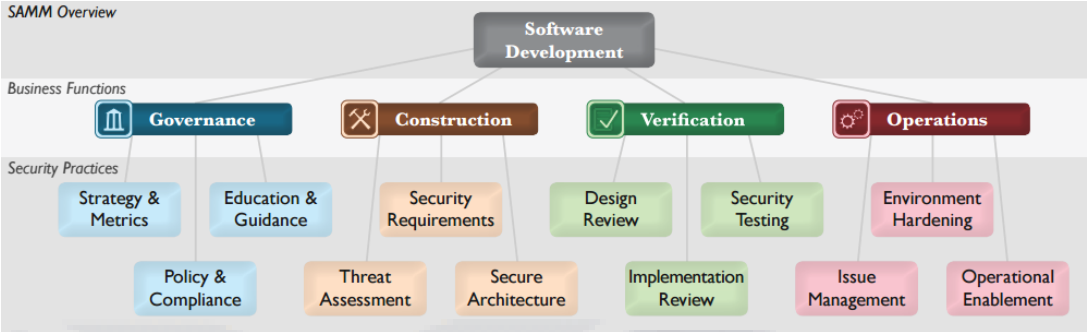


Imagen 6.Visión General del Maro de Trabajo SAMM de OWASP (OWASP, 2011).

El marco de trabajo SAMM es una colección de prácticas de seguridad que están ligadas a las funciones más importantes de la organización, esto es, a sus procesos de negocio sin dejar de considerar a los recursos humanos y tecnológicos involucrados en el proceso de creación de aplicaciones. En este sentido y dentro de éste entorno de trabajo, el desarrollo de software se divide en cuatro grandes áreas (funciones): Gobernanza, Construcción, Verificación y Operaciones. Cada una de estas áreas a su vez se subdivide en pequeños módulos para su mejor entendimiento y aplicación.

Debido a que el objetivo principal de este trabajo practico es analizar las aplicaciones web 2.0 de la organización, nos vamos a enfocar en el módulo de Verificación que es donde justamente se encuentra el sub módulo de Pruebas de Seguridad, además de esto, existen otras limitantes para este caso práctico como son principalmente el tiempo y la extensión de cada módulo del marco de trabajo SAMM, la disponibilidad de los recursos para llevar a cabo el análisis de cada fase en el ciclo de vida de desarrollo y de software por lo anterior mencionado, solo se tomará en cuenta esta etapa junto con sus 3 sub-módulos.

4.2.1. Verificación

El módulo de Verificación se enfoca en las actividades de la organización, que permiten evaluar los productos de software a lo largo del ciclo de desarrollo de software. Se incluyen dentro del módulo las actividades del área de aseguramiento de la calidad (QA por sus siglas en inglés) y pueden incluirse otras revisiones y/o evaluaciones. Dentro del módulo se tienen tres sub-módulos: Revisión del diseño, revisión de la implementación y pruebas de seguridad.

4.2.2. Revisión del diseño. Descripción de la práctica de seguridad.

Es la inspección de los distintos artefactos creados en la etapa de diseño para asegurar que se cumplen las expectativas de la organización en materia de seguridad.

Esta práctica tiene como enfoque principal la evaluación del diseño del software existente, así como la arquitectura de seguridad tomada en cuenta para la construcción del mismo, lo que le permite a una organización encontrar vulnerabilidades temprano en el ciclo de vida de desarrollo de software antes de que se conviertan en problemas y por lo tanto sea más caro arreglarlos evitando altos costos potenciales.

Se empieza con pequeñas actividades que no tienen gran peso pero que ayudan a entender detalles importantes de seguridad en la arquitectura y diseño de la aplicación, en etapas avanzadas esto se convierte en métodos de inspección más formales.

El uso de estas prácticas involucra directamente la revisión más detallada del diseño a nivel de datos y el reforzamiento de las expectativas de seguridad desde la base o núcleo que llevan a evaluaciones de diseño y revisiones de las fallas documentadas encontradas antes de que se aprueben las liberaciones a producción.

4.2.3. Descripción de las Actividades. Revisión del Diseño - Fase 1.

Objetivo: Brindar ayuda a las revisiones de diseño de software para asegurar que los riesgos encontrados sean mitigados desde la línea base.

Actividades: Identificar la superficie del ataque al software y analizar el diseño comparando contra los requerimientos de seguridad conocidos.

Resultados:

- Entendimiento a alto nivel de las implicaciones de seguridad para el perímetro de la arquitectura.
- Alentar el uso de mejores prácticas en los equipos de desarrollo en el área de verificación del diseño.
- Implementar el proceso de revisión de diseño a nivel de proyecto.

4.2.4. Descripción de las Actividades. Revisión del Diseño – Fase 2.

Objetivo: Ofrecer los servicios de evaluación para revisión del diseño del software y comparar contra buenas prácticas de seguridad.

Actividades: Analizar en búsqueda del uso de mecanismos completos de seguridad e implementar el servicio de revisión del diseño en cada equipo de trabajo.

Resultados:

- Ofrecer formalmente el servicio de evaluación para revisar constantemente la seguridad de la arquitectura.
- Determinar con precisión las fallas en seguridad en versiones pasadas y tomarlas en cuenta para mantenimiento de futuras versiones.

- Mayor entendimiento a profundidad entre los interesados sobre como el software proporciona protecciones de seguridad.

4.2.5. Descripción de las Actividades. Revisión del Diseño - Fase 3.

Objetivo: Hacer obligatorio el uso de evaluaciones y validar artefactos para desarrollar y aumentar el mejor entendimiento de los mecanismos de protección.

Actividades: Crear diagramas de flujo de datos para recursos con información sensible y establecer puntos de inspección en cada entrega, para las revisiones de diseño.

Resultados:

- Vista granular de puntos débiles en el diseño de un sistema para fomentar una mejor compartimentación.
- Concientización a nivel organización del estado del proyecto contra las expectativas básicas en seguridad para la arquitectura.
- Comparación entre proyectos de la eficiencia y progreso al mitigar fallos conocidos.

4.2.6. Revisión de la Implementación. Descripción de la práctica de Seguridad.

La revisión de la implementación permite obtener la evaluación del código fuente que ayuda a encontrar vulnerabilidades, antes de que este sea implementado además de brindar una estrategia de mitigación y la base para una codificación segura.

El foco de esta etapa es la inspección y configuración del código fuente del software para detectar vulnerabilidades de seguridad las cuales son en la mayoría de los casos simples de modo conceptual pero incluso para los

desarrolladores más experimentados es fácil dejar pasar o cometer errores que pudieran vulnerar y/o dejar el software expuesto o comprometido a ataques.

Se utilizan simples listas de control en los módulos con mayor criticidad de la aplicación y se pretende que al madurar la organización estos controles se hagan por medio de automatización para aumentar el grado de cobertura y disminuir el tiempo dedicado a las actividades de implementación.

Al usar esta práctica se requiere mayor grado de integración en el proceso de desarrollo para detectar problemas lo más tempranamente posible. Esto ayuda a las organizaciones a tener mejores auditorias y fija como objetivo encontrar fallos antes de las liberaciones o entregas a producción.

4.2.7. Descripción de las Actividades. Revisión de la Implementación – Fase 1.

Objetivo: Encontrar vulnerabilidades básicas a nivel de código y otros problemas de seguridad de gran riesgo.

Actividades: Crear listas de control de revisión a partir de requerimientos de seguridad conocidos y realizar revisiones puntuales en el código de alto riesgo.

Resultados:

- Inspección en búsqueda de vulnerabilidades comunes de configuración o en el código, que conduzcan a su descubrimiento o ataque.
- Revisión ligera en búsqueda de errores de código que lleven a un impacto de seguridad severo.
- Actividad debida a nivel de código básico para el aseguramiento de la seguridad.

4.2.8. Descripción de las Actividades. Revisión de la Implementación - Fase 2.

Objetivo: Hacer las revisiones de implementación, que se llevan a cabo durante el tiempo de desarrollo, más precisas y eficientes por medio del uso de automatización.

Actividades: Utilizar herramientas automatizadas para análisis de código e integrar las actividades de análisis de código en el proceso de desarrollo.

Resultados:

- El equipo de desarrollo es capaz de verificar consistentemente a nivel de código en búsqueda de vulnerabilidades de seguridad.
- Analizar los resultados rutinariamente para compilar datos históricos sobre los hábitos de codificación segura del equipo.
- Las partes interesadas están al tanto de las vulnerabilidades sin mitigar para mejorar el análisis de compensación.

4.2.9. Descripción de las Actividades. Revisión de la Implementación - Fase 3.

Objetivo: Hacer obligatorio el proceso de revisión de implementación para encontrar riesgos a nivel de lenguaje (código) y específicos a la aplicación.

Actividades: Personalizar el análisis de código para las preocupaciones relacionadas a cada aplicación específica y establecer puntos de control en cada entrega para revisiones de código.

Resultados:

- Incremento de la confianza en la precisión y aplicabilidad de los resultados del análisis del código.

- Creación de una línea base a lo largo de la organización para las expectativas de codificación segura.

- Los equipos de cada proyecto tienen una meta objetiva al momento de juzgar la seguridad a nivel de código.

4.2.10. Pruebas de Seguridad. Descripción de la Práctica de Seguridad

Son las pruebas que se realizan en el ambiente y tiempo de ejecución para encontrar vulnerabilidades y que además definen un estándar mínimo para cada reléase dentro del ciclo de vida del software.

En esta práctica nos enfocaremos en las pruebas de seguridad que son llevadas a cabo en tiempo real en el ambiente de ejecución para encontrar defectos. Estas pruebas de seguridad refuerzan y fijan el objetivo de aseguramiento de la calidad al ser ejecutadas en un ambiente de pruebas similar al ambiente real de producción lo cual hace visibles problemas de configuración o errores en la lógica del negocio que de otra forma serian difíciles de encontrar.

La organización comienza con pruebas manuales de penetración y casos de prueba a alto nivel, después de esto se espera que al aumentar el grado de madurez, se cambie a pruebas automatizadas que cubran el amplio espectro de lo que podríamos llamar una vulnerabilidad en el software.

Al llegar a un grado avanzado de madurez se pretende que las pruebas automatizadas sean personalizadas o adaptadas en un conjunto de pruebas de ejecución que cubran las preocupaciones detalladas, específicas a cada aplicación. Con esto se trata de reducir al mínimo los fallos encontrados de seguridad en cada entrega.

4.2.11. Descripción de las Actividades. Pruebas de Seguridad – Fase 1.

Objetivo: Establecer procesos para ejecutar pruebas básicas de seguridad en la implementación y requerimientos de software.

Actividades: Obtener casos de prueba a partir de los requerimientos de seguridad conocidos y llevar a cabo pruebas de penetración en cada liberación a producción.

Resultados:

- Verificación independiente de los mecanismos de seguridad esperados alrededor de las funciones criticas de negocio.
- Actividades debidas de alto nivel enfocadas a las pruebas de seguridad.
- Crecimiento adecuado de un conjunto de pruebas para cada proyecto de software.

4.2.12. Descripción de las Actividades. Pruebas de Seguridad – Fase 2.

Objetivo: Hacer las pruebas de seguridad más completas y eficientes con el uso de automatización.

Actividades: Utilizar herramientas automatizadas para pruebas de seguridad e integrar las pruebas de seguridad desde el proceso de desarrollo.

Resultados:

- Verificación más profunda y más consistente de la funcionalidad del software en materia de seguridad.
- Los equipos de desarrollo son capaces de verificar y corregir problemas antes de cada liberación a producción.
- Las partes interesadas están más conscientes de vulnerabilidades abiertas cuando tomen decisiones de aceptación de riesgos.

4.2.13. Descripción de las Actividades. Pruebas de Seguridad – Fase 3.

Objetivo: Hacer obligatorio el uso de pruebas de seguridad específicas a cada aplicación para asegurar seguridad básica antes de cada despliegue.

Actividades: Emplear herramientas automatizadas de pruebas de seguridad para cada aplicación específica y establecer puntos de control para pruebas de seguridad.

Resultados:

- Creación de una línea base a lo largo de la organización para el desempeño esperado de una aplicación contra ataques.
- Creación de un conjunto de pruebas de seguridad a la medida para mejorar la precisión del análisis automatizado.
- Los equipos de cada proyecto están conscientes de las metas y objetivos para resistir ataques.

4.3. OWASP ZAP

OWASP Zed Attack Proxy (por sus siglas en inglés ZAP), es una herramienta gratuita de la fundación OWASP. Está diseñada para monitorear la seguridad en aplicaciones Web. Tiene como características: Totalmente gratuita y de código abierto; multiplataforma; flexible y extensible y con una gran comunidad en la red.

Mediante sus funciones y análisis específicos la herramienta ayuda a encontrar, durante el ciclo de desarrollo, vulnerabilidades de seguridad en las aplicaciones web. Algunas de las funciones y análisis son:

- Comprobar todas las peticiones y respuestas entre cliente y servidor.
- Localizar recursos en un servidor.

- Análisis automáticos y pasivos.
- Lanzar varios ataques a la vez.
- Utilizar certificados SSL dinámicos.
- Análisis de sistemas de autenticación.

4.4. Revisión de Intervenciones similares

4.4.1. Caso Similar 1.- (Lutz, Boucher, & Roustant, 2013).

Título de la Tesis: A Study of Penetration Testing Tools and Approaches (Chiem, 2014).

Esta tesis es exclusivamente sobre Pruebas de Penetración o Penetration Testing como la técnica más conocida y usada para simular ataques reales pero de manera legal al recrear lo que un hacker hace en la vida real con el objetivo de encontrar o sacar a la luz vulnerabilidades potenciales ya existentes en el sistema y obtener una solución para mitigar esas debilidades existentes mejorando de esta forma la seguridad en toda la aplicación o sistema. Describe además una serie de herramientas automatizadas que ayudan a conducir estas pruebas de penetración, la dificultad para elegir una que se adecuada para la organización o proyecto y brinda una comparación de las distintas opciones en el mercado como son NMap, Nessus, OpenVAS y GFI Languard. Indica además un método para llevar a cabo actividades de análisis después del ataque, el cual llama modelo de árbol de ataques, junto con diagramas que fueron analizados para detectar las superficies de ataque más comunes y neutralizarlos. Por último, comenta que las vulnerabilidades más críticas se pueden encontrar o presentar cuando hacemos uso de Sistemas Operativos que no están actualizados o servicios de seguridad sin parchar, esto permite a los atacantes obtener acceso sin mucho esfuerzo y tiempo además de la debilidad en las contraseñas de los usuarios lo cual es muy común y se puede tomar ventaja de ellos para obtener acceso al sistema y escala en actividades maliciosas.

4.4.2. Caso Similar 2.- (Lutz et al., 2013).

Título de la Tesis: Análisis de la seguridad de sitios web (Rodríguez Argueta, 2011).

En esta tesis se presenta un estudio sobre la calidad de la seguridad en los sitios web con dominio gob.mx y propone un estándar para evaluar los sitios web. Toma una muestra de los sitios web y los divide en diferentes criterios como zonas económicas e índice de violencia y criminalidad ayudándose del INEGI y CIDAC, presenta además una referencia sobre las vulnerabilidades más comunes en los sitios del dominio gob.mx, concluyendo la gran deficiencia que existe en proyectos de carácter gubernamental, encontrando fallas como XSS e Inyección SQL. Al final hace recomendaciones a los encargados de crear las páginas web para gobierno para que reduzcan el número de vulnerabilidades insertadas o pasadas por alto, además de metodologías para el desarrollo seguro de estos sitios usando Procedimientos Operativos Estándar o POE, las cuales comenta, surgieron en la medicina y se han ido expandiendo a diversas áreas de las tecnologías de información. Por último, recomienda la creación de normas nacionales para el desarrollo de sitios web y que un organismo central valide y verifique estas tareas, teniendo el poder para negar la publicación de los sitios si no cumple con los estándares mínimos requeridos o recomendados.

4.4.3. Caso Similar 3.- (Yañez Cedeño, Ericka 2015).

Título de la Tesis: Análisis de las herramientas para el proceso de auditoría de seguridad informática usando Kali Linux (Cedeño, 2015).

En esta tesis se comenta como ha ido ganando terreno el tema de seguridad debido a los ataques presentados a diversas áreas como servidores, redes, aplicaciones etc. Enfocándose primordialmente en auditorias de seguridad, menciona que el área de sistemas y redes en una organización se debe mantener preparado para hacer frente si se presentase un problema.

Propone además la idea de que los recursos de sistemas y/o redes de la organización tomen capacitación con Kali Linux para tomar el rol de Ethical Hackers, de esta forma pensando como un atacante de la vida real y hacer uso de las herramientas que esta distribución brinda para proteger la información de la organización. Pensado como una guía que se pudiera seguir al momento de una auditoria, se habla de las fases y herramientas para cada etapa, destacando herramientas que pudieran ser utilizadas brindando un análisis detallado. Hace uso además, de un laboratorio de pruebas para poder entrenar a los recursos.

4.5. Contribuciones y Limitaciones

En la siguiente tabla (Tabla 3) podemos ver una comparación de este trabajo practico contra los casos similares analizados tomando en cuenta las características de cada uno de ellos. Basados en la tabla alcanzamos a ver que en este documento se toman en cuenta un mayor espectro de los temas analizados como herramientas y técnicas enfocadas a la organización.

Tabla 3. Comparación Trabajo Práctico vs Casos Similares.

Característica	Caso 0	Caso 1	Caso 2	Caso 3
Penetration Testing	✓	✓		
NMap	✓	✓		
Nessus	✓	✓		
OpenVAS		✓		
GFI Languard		✓		
INEGI	✓		✓	
Inyección SQL	✓		✓	
XSS	✓		✓	
Kali Linux				✓
Ethical Hacking	✓			✓
OWASP	✓			

5. Metodología para desarrollar la solución al caso problema

Es importante señalar que por motivos de seguridad y confidencialidad de los resultados obtenidos, no se pueden adicionar al caso práctico los resultados numéricos de la evaluación del módulo de Verificación del marco de trabajo SAMM, así como información sensible sobre los proyectos utilizados, direcciones URL, nombres de campos, nombres de dueños de negocio, documentos o diagramas a menos que sea de forma general o que sea clara y específicamente otorgado un permiso por el INEGI.

5.1. Revisión del Diseño Fase 1

En esta fase se brinda ayuda a las revisiones de diseño de software para de dicha forma lograr asegurarse que los riesgos encontrados sean mitigados desde la línea base.

5.1.1. Actividad 1. Identificar la superficie del ataque al software.

Para cada proyecto de software se creó una vista simplificada de la arquitectura a alto nivel de abstracción, basada en los artefactos del proyecto como los requerimientos a alto nivel y documentos de diseño, entrevistas con el personal técnico o revisiones a nivel modular del código base. Siguiendo las reglas generales de SAMM de que la vista de la arquitectura solo puede abarcar una página.

A partir del diagrama creado, se analizó cada componente en términos de accesibilidad de las interfaces de cada tipo de usuario. Los componentes que proveen las interfaces fueron considerados en la vista sencilla (de una sola página) para encontrar puntos funcionales donde se pasen datos a otros componentes en el diagrama. Se agruparon bajo perfiles similares las interfaces y componentes, con el objetivo de definir las superficies de ataque.

De cada interface se realizó una búsqueda de funcionalidad extra relacionada a la seguridad y se tomaron en cuenta las superficies de ataque identificadas y

se verifico la consistencia del diagrama, según otras interfaces con accesos similares. Si se detectaba una inconsistencia se considerada como un defecto encontrado en la evaluación.

El análisis fue realizado por el equipo experto en seguridad de forma conjunta con los arquitectos, desarrolladores, gerentes y auditores de seguridad de cada proyecto analizado.

5.1.2. Actividad 2. Analizar el diseño comparando contra los requerimientos de seguridad conocidos.

Se identificaron y reunieron los requerimientos tanto formales como conocidos informalmente para cada proyecto analizado y se incluyeron las reglas de seguridad generales con que la aplicación debe contar.

Se revisó y comparo cada elemento de la lista de requerimientos de seguridad conocidos contra el diagrama general inicial de la arquitectura del sistema y se elaboró un diagrama para mostrar las características a nivel de diseño que cubren cada requerimiento de seguridad a la vez que se anotó como un defecto en la evaluación si el requerimiento de seguridad no estaba cubierto por el diseño del sistema.

Este análisis fue llevado a cabo por expertos en seguridad con aportes y/o comentarios de los arquitectos, desarrolladores, gerentes y dueños del negocio.

Luego de obtenidos los análisis se realizaron dos preguntas según como se muestra en la Tabla 4. Que son calificadas con base en una escala de: 0.0 (No), 0.2 (Algo), 0.5 (La mitad) y 1.0 (La mayoría).

Tabla 4. Hoja de Trabajo de Evaluación de Revisión del Diseño - Fase 1.

Hoja de Trabajo de Evaluación	Puntuación
¿Los equipos de cada proyecto documentan la superficie del ataque de diseño del software?	
¿Los equipos de cada proyecto validan el diseño del software contra riesgos conocidos de seguridad?	

Para SAMM las métricas de éxito mínimas para pasar a la siguiente fase son:

- El número de proyectos analizados con actualizaciones en la superficie de ataque en los últimos 12 meses es mayor al 50%.
- El número de proyectos analizados con actualizaciones a los requerimientos de seguridad a nivel de diseño en los últimos 12 meses es mayor al 50%.

5.2. Revisión del Diseño Fase 2

En esta fase se ofrecen servicios de evaluación para revisión del diseño del software y comparar contra buenas prácticas de seguridad.

5.2.1. Actividad 1. Analizar en búsqueda del uso de mecanismos completos de seguridad.

En esta actividad se realizó para cada interface (interna o externa), en cada módulo distinto en el diagrama de arquitectura de alto nivel, iteraciones a lo largo de la lista de mecanismos de seguridad y se analizó el sistema para su suministro.

Se tuvieron en cuenta seis mecanismos de seguridad, autenticación, autorización, validación de entradas, codificación de salidas, manejo de errores,

manejo de registros y el manejo de sesiones Para cada interface, se determinó la parte del sistema que provee cada mecanismo y se hizo una anotación de cualquier característica faltante o que no sea clara para levantar un defecto.

Esta actividad se llevó a cabo en conjunto con los expertos de seguridad y con la ayuda del equipo de cada proyecto que tenga el conocimiento específico de cada aplicación.

5.2.2. Actividad 2. Implementar el servicio de revisión del diseño en cada equipo de trabajo.

Para el desarrollo esta actividad, se realizó la implementación de un proceso donde los interesados del proyecto pudieran hacer las peticiones de revisión de diseño. Dicho servicio se distribuyó entre el personal y todos los miembros revisores tomaron un entrenamiento para desempeñar sus actividades completa y consistentemente.

Las revisiones del proyecto que fueron obtenidas se analizaron y fueron priorizada por el Grupo de Ingeniería en Sistemas de INEGI, lo que hizo posible que la priorización de revisiones de proyecto estuviera alineada correctamente con el riesgo. El equipo logro formular un acuerdo de la superficie de ataque, emparejar los requerimientos específicos de seguridad con los elementos de diseño y verificar los mecanismos de seguridad en las interfaces modulares.

Luego de realizados ambas actividades y obtenidos los análisis se realizaron de igual forma que en la Fase 1, dos preguntas, que son mostradas en la Tabla 5. Dichas preguntas podían tomar una calificación dentro de la escala de: 0.0 (No), 0.2 (Algo), 0.5 (La mitad) y 1.0 (La mayoría).

Tabla 5. Hoja de Trabajo de Evaluación de Revisión del Diseño - Fase 2.

Hoja de Trabajo de Evaluación	Puntuación
¿Los equipos de cada proyecto analizan específicamente los elementos de diseño para mecanismos de seguridad?	
¿Las partes interesadas del proyecto están al tanto de cómo obtener una revisión de diseño de seguridad formal?	

Para SAMM las métricas de éxito mínimas para pasar a la Revisión del Diseño - Fase 3 son:

- El número total de las partes interesadas informadas sobre el estado de las solicitudes de revisión en los últimos 6 meses debe ser mayor al 80%.
- El número total de proyectos en proceso de revisión de diseño en los últimos 12 meses debe ser mayor al 75%.

Por motivos de seguridad y confidencialidad de los resultados obtenidos, no se pueden adicionar al caso práctico los resultados numéricos de la evaluación de la Revisión del Diseño – Fase 2.

5.3. Revisión del Diseño Fase 3

En esta fase se utilizaron y se validaron los artefactos para desarrollar y aumentar el mejor entendimiento de los mecanismos de protección de cada proyecto analizado.

5.3.1. Actividad 1. Crear diagramas de flujo de datos para recursos con información sensible.

Cada proyecto de software, según su función comercial, fue analizado con el objetivo de identificar detalles en el comportamiento del sistema, que en sus

funciones críticas pudieran generar altos riesgos para la institución. Normalmente esta funcionalidad se correlaciona con las características que implementan la creación, acceso, actualización y supresión de datos sensible.

Según SAMM la funcionalidad de alto riesgo incluye la lógica de negocio específica al proyecto que es crítica por naturaleza, ya sea por negación de servicio o perspectiva de compromiso.

Para cada función comercial se hizo la selección y notación, de forma estándar, capturándose así los módulos, fuentes de datos, actores y mensajes, de la aplicación que fueran relevantes. La realización de esta actividad se llevó a cabo con la realización de un diagrama de diseño a alto nivel, en el que se fueron removiendo iterativamente detalles relevantes mientras se iban descartando elementos que no correspondían al recurso sensible. De igual forma el diagrama fue analizado para determinar cuellos de botella en el diseño.

5.3.2. Actividad 2. Establecer puntos de inspección en cada entrega, para las revisiones de diseño.

Llegado a este punto se obtuvo un programa consistente de revisión de diseño, pero para seguirlo reforzando se definió un punto en el ciclo de vida de desarrollo de las aplicaciones web 2.0, donde no se les permitiría avanzar a la siguiente fase, si aún existían defectos se severidad alta abiertos, sin atender o sin ser aceptados por el negocio.

Se detectaron sistemas heredados o proyectos inactivos, los cuales dada su importancia continuaban en operación. Para este tipo de proyectos se generó una excepción, con marcos de tiempo asignados cada proyecto, para ser revisados e identificar vulnerabilidades escondidas en los sistemas existentes. Según SAMM estas excepciones no deben exceder del 20% de todos los proyectos.

De igual forma que en las fases anteriores, luego de realizadas ambas actividades se hicieron, dos preguntas, que son mostradas en la Tabla 6. Dichas preguntas podían tomar una calificación dentro de una escala de entre: 0.0 (No), 0.2 (Algo), 0.5 (La mitad) y 1.0 (La mayoría).

Tabla 6. Hoja de Trabajo de Evaluación de Revisión del Diseño - Fase 3.

Hoja de Trabajo de Evaluación - Revisión de Diseño	Puntuación
¿El proceso de revisión de diseño de seguridad incorpora análisis detallado a nivel de datos?	
¿Existe un nivel mínimo básico para la revisión de los resultados de seguridad de diseño?	

Las métricas de éxito mínimas, según SAMM, para cumplir con esta fase son:

- El número de proyectos con actualizaciones a los diagramas de datos en los últimos seis meses debe ser mayor al 80%.
- El número de proyectos que pase la auditoria de revisión de diseño en los últimos seis meses debe ser mayor al 75%.

5.4. Revisión de la Implementación Fase 1

Para esta fase se tenía como objetivo identificar y encontrar vulnerabilidades básicas a nivel de código y otros problemas de seguridad de gran riesgo en las aplicaciones web 2.0.

5.4.1. Actividad 1. Crear listas de control de revisión a partir de requerimientos de seguridad conocidos.

Debido a que estamos analizando las aplicaciones web 2.0 creadas con tecnología Oracle Java y Microsoft .NET fueron creadas dos listas de control de

revisión de implementación para seguridad, según las preocupaciones alrededor de los requerimientos funcionales para mejores prácticas en codificación segura. Estas listas de control fueron revisadas por las partes técnicas interesadas como gerentes de desarrollo, arquitectos, desarrolladores y auditores de seguridad.

Estas listas de control de revisión de implementación para seguridad se crearon cortas y sencillas, logrando mayor eficiencia, pues se tenía como objetivo localizar problemas de alto riesgo que fueran fáciles de encontrar en el código, ya sea de forma manual o con herramientas de búsqueda

5.4.2. Actividad 2. Realizar revisiones puntuales en el código de alto riesgo.

Se determinó utilizar las listas de control de revisión de implementación como parte normal del ciclo de desarrollo donde a los integrantes del proyecto se les asignaron módulos donde se encontraron vulnerabilidades en el código para que fueran revisados si se realizaba algún cambio, de igual forma se determinó que se podían utilizar herramientas automatizadas de revisión y/o auditores de seguridad según se hiciera necesario. Para esta actividad se definió que los gerentes de desarrollo tenían que discutir los defectos y priorizar la solución apropiada a las vulnerabilidades, teniendo en cuenta los comentarios de las otras partes interesadas.

Para esta fase las preguntas de control, que se muestran en la Tabla 7, se podían calificar dentro de una escala de entre: 0.0 (No), 0.2 (Bus Área o Algo), 0.5 (A lo largo de la organización o la mitad) y 1.0 (A lo largo de la organización y requerido o la mayoría).

Tabla 7. Hoja de Trabajo de Evaluación - Revisión de la Implementación Fase 1.

Hoja de Trabajo de Evaluación - Revisión de la Implementación	Puntuación
¿Los equipos de proyecto tienen revisiones de listas de control (checklist) basadas en problemas de seguridad comunes?	
¿Los equipos de proyecto analizan el código seleccionado como de alto riesgo?	

Las métricas de éxito mínimas de SAMM para pasar a la siguiente fase son:

- El número de equipos de proyecto informados sobre listas de control de revisión de código relevantes en los últimos seis meses debe ser mayor a 80%.
- El número de equipos de proyecto que llevan a cabo tareas de revisión de código de alto riesgo en los últimos seis meses debe ser mayor a 50%.
- La escala Likert sobre la usabilidad de las listas de control de revisiones de código reportadas por el equipo de desarrollo deber ser mayor a 3.0.

5.5. Revisión de la Implementación Fase 2

Para esta fase se tenía como objetivo lograr que las revisiones de implementación durante el ciclo de desarrollo fueran más precisas y eficientes usando herramientas de automatización.

5.5.1. Actividad 1. Utilizar herramientas automatizadas para análisis de código.

La mayoría de las vulnerabilidades de seguridad a nivel de código son complejas de entender y requieren inspección cuidadosa para encontrarlas, por lo cual es necesario utilizar herramientas automatizadas para análisis de código, que aunque pueden producir falsos negativos, nos permiten ahorrar tiempo y energía durante el ciclo de desarrollo, y de esta forma enfocar la atención en

secciones con la mayoría del código sospechoso pudiendo así detectar automáticamente errores y vulnerabilidades.

INEGI dada su trayectoria e importancia tenían definidas previamente las herramientas automatizadas para análisis de código a utilizar que cubren sus necesidades para las plataformas Java y .Net

5.5.2. Actividad 2. Integrar las actividades de análisis de código en el proceso de desarrollo.

Se analizó la integración de las herramientas seleccionadas por INEGI para el análisis de código en los distintos equipos de trabajo de desarrollo. Se pudo obtener la forma en la que realizan los escaneos, los cuales se realizan ya sea automáticamente o de forma manual desde el código en el repositorio del proyecto, según las características de cada proyecto.

La utilización de las herramientas automáticas para el análisis de código les permite obtener que los resultados del análisis estén disponibles antes de la liberación y se logra que el equipo de desarrollo los verifique.

Un problema potencial con los sistemas heredados o grandes proyectos en curso es que los escáneres de código pueden reportar defectos en módulos que no fueron actualizados en esta liberación. Se verificó la configuración del escaneo automático para que se ejecute periódicamente, y se limitó al software para que solo reporte los defectos que han sido agregados, removidos o cambiados desde el último escaneo. Se indicó la importancia de no ignorar el resto de los resultados para que los gerentes de desarrollo puedan recibir comentarios de los auditores de seguridad, partes interesadas y el equipo de proyecto que les permitiera formular un plan concreto para retomar el resto de los defectos.

Los defectos que no son retomados en la revisión de implementación permanecen a la hora de la liberación a producción, estos deben ser revisados, asignarles un valor de riesgo y aceptados por las partes interesadas.

Para esta fase las preguntas de control, que se muestran en la Tabla 8, se podían calificar con una escala de entre: 0.0 (No), 0.2 (Por equipo o Algo), 0.5 (A lo largo de la organización o la mitad) y 1.0 (Proceso Integrado o la mayoría).

Tabla 8. Hoja de Trabajo de Evaluación - Revisión de la Implementación Fase 2.

Hoja de Trabajo de Evaluación - Revisión de la Implementación	Puntuación
¿Los equipos de proyecto tienen acceso a herramientas automatizadas para análisis de código para encontrar problemas de seguridad?	
¿Las Partes Interesadas o Stakeholders revisan constantemente los resultados de las revisiones de código?	

Según SAMM las métricas de éxito de esta fase son:

- El número de proyectos con revisiones de código y visto bueno de las partes interesadas en los últimos seis meses debe ser mayor al 50%.
- El número de proyectos con acceso a resultados automatizados de revisión de código en el último mes debe ser mayor al 80%.

5.6. Revisión de la Implementación - Fase 3

En esta fase se necesita revisar si los equipos de desarrollo hacen uso de forma obligatoria del proceso de revisión de implementación para encontrar riesgos a nivel de lenguaje y/o específicos para las aplicaciones web 2.0.

5.6.1. Actividad 1. Personalizar el análisis de código para las preocupaciones relacionadas a cada aplicación específica.

Las herramientas de escaneo usan de motor reglas de una base de conocimiento para validar el código basándose en APIs de lenguaje y librerías usadas comúnmente, pero tienen habilidad limitada para entender APIs personalizadas y diseños para validaciones análogas. Sin embargo, a través de la personalización, un escáner de código puede ser un poderoso motor de análisis para detectar problemas de seguridad en la organización y específicos al proyecto.

Mientras que los detalles varían entre herramientas en términos de facilidad y poder análisis personalizado, la personalización del escáner de código generalmente involucra controles específicos llevados a cabo con APIs específicas y llamadas de funciones a sitios. Los controles pueden incluir análisis de adherencia a estándares de codificación internos, datos corruptos sin verificar enviados a interfaces personalizadas, rastreo y verificación de manejo de datos sensibles, uso correcto de APIs internas, etc.

Los controles para uso de bases de código compartidas son un lugar efectivo para comenzar los escaneos personalizados debido a que los controles creados pueden ser utilizados entre múltiples proyectos. Para personalizar una herramienta para código base, un auditor de seguridad debe inspeccionar el código y el diseño a alto nivel para identificar los controles candidatos a discutir con el equipo de desarrollo y las partes interesadas para implementación.

5.6.2. Actividad 2. Establecer puntos de control en cada entrega para revisiones de código.

Para establecer una línea base de seguridad para todos los proyectos, se debe establecer un punto particular en el ciclo de vida de desarrollo de software como un punto de control donde un estándar mínimo para que los resultados de

revisión de la implementación sean alcanzados para poder realizar una liberación a producción.

Para empezar, este estándar debe ser fácil de seguir, por ejemplo al elegir uno o dos tipos de vulnerabilidades y dejando en claro que ningún proyecto podrá avanzar si se encuentran defectos de estos dos tipos. Con el tiempo, este estándar básico deberá ser mejorado al añadir criterios adicionales para aprobar el punto de control.

Generalmente, el punto de control en la revisión de implementación debería revisarse hacia el final de la fase de implementación, pero antes de la liberación a producción.

Para los sistemas heredados o proyectos inactivos, se debe crear un proceso de excepción para permitir que estos proyectos continúen operaciones, pero con un plazo de tiempo explícitamente asignado para mitigar riesgos o solucionar defectos. Estas excepciones deben ser limitadas a no más del 20% de todos los proyectos.

Para esta fase las preguntas de control, que se muestran en la Tabla 9, se podían calificar dentro de una escala de entre: 0.0 (No), 0.2 (Bus área o por equipo), 0.5 (A lo largo de la organización) y 1.0 (A lo largo de la organización y requerido o Proceso Integrado).

Tabla 9. Hoja de Trabajo de Evaluación - Revisión de la Implementación Fase 3.

Hoja de Trabajo de Evaluación - Revisión de la Implementación	Puntuación
¿Los equipos de proyecto utilizan automatización para validar el código comparando contra estándares de codificación específicos a cada aplicación?	
¿Existe un nivel mínimo de seguridad base para los resultados de revisiones de código?	

Las métricas de éxito mínimas requeridas por SAMM son:

- La cantidad de proyectos usando análisis personalizados de código debe ser mayor al 50%.
- El número de proyectos que pasen las auditorias de revisión de código en los últimos seis meses debe ser mayor al 75%.

5.7. Pruebas de Seguridad - Fase 1

En esta fase se establecieron los procesos para ejecutar pruebas básicas de seguridad en la implementación y requerimientos de software.

5.7.1. Actividad 1. Obtener casos de prueba a partir de los requerimientos de seguridad conocidos.

Usando los requerimientos de seguridad de cada proyecto, se identificaron un conjunto de casos de prueba para ayudar a la validación correcta de funcionalidad del software, tomando en cuenta las preocupaciones de seguridad en torno a los requerimientos funcionales y la lógica del negocio del sistema. También se incluyeron casos de prueba para las vulnerabilidades comunes de los lenguajes implementados en las aplicaciones Java y .Net.

Se verificó con los equipos de proyecto en qué etapa se construyen los casos de pruebas específicos para cada aplicación además de asentar como buena práctica la creación de un conjunto de casos de prueba generales de seguridad como un tipo de regresión. Así como empezar a usar herramientas de automatización para cubrir este conjunto de casos de pruebas de seguridad.

Como ya se sabe, basados en el ciclo de vida de desarrollo del software entre más temprano se comience con la etapa de pruebas es mejor. El conjunto de casos de pruebas de seguridad candidatos, fueron revisados para lograr aplicabilidad, eficacia y factibilidad por el equipo de desarrollo, seguridad y aseguramiento de la calidad.

5.7.2. Actividad 2. Llevar a cabo pruebas de penetración en cada liberación a producción.

Una vez que se identificó el conjunto de casos de prueba de seguridad, se realizaron pruebas de penetración para evaluar el desempeño del sistema en los proyectos que estaban en la fase de pruebas antes de la liberación a producción, con el objetivo de probar la robustez de la lógica de negocio y pruebas comunes de vulnerabilidad checando el diseño y la implementación. Todo esto fue monitoreado por el auditor de seguridad en su primera fase, para lograr entrenar a los distintos miembros del equipo.

Una vez que se terminó el proceso de pruebas de seguridad, se enviaron los resultados a revisión con las partes interesadas para que aceptaran los riesgos indicados sobre las posibles problemas que pudieran presentarse en producción, a su vez, las partes interesadas definieron un periodo de tiempo concreto para revisar y solucionar las posibles brechas.

Para esta fase las preguntas de control, que se muestran en la Tabla 10, se podían calificar con una escala de entre: 0.0 (No), 0.2 (Algo), 0.5 (La mitad) y 1.0 (La mayoría).

Tabla 10. Hoja de Trabajo de Evaluación - Pruebas de Seguridad Fase 1.

Hoja de Trabajo de Evaluación - Pruebas de Seguridad	Puntuación
¿Los proyectos especifican pruebas de seguridad basados en requerimientos definidos de seguridad?	
¿Se llevan a cabo pruebas de penetración en proyectos de alto riesgo antes de cada liberación a producción?	

De acuerdo a SAMM las métricas de éxito para esta fase son:

- El número de proyectos con casos de prueba de seguridad en los últimos 12 meses debe ser mayor al 50%.

- El número de las partes interesadas informados sobre el estado de las pruebas de seguridad en los últimos seis meses debe ser mayor al 50%.

5.8. Pruebas de Seguridad - Fase 2

En esta fase teníamos como objetivo hacer uso de automatización para lograr que las pruebas de seguridad sean más completas y eficientes.

5.8.1. Actividad 1. Utilizar herramientas automatizadas para pruebas de seguridad.

Para lograr hacer las pruebas en búsqueda de problemas de seguridad, se validaron un gran número de pruebas de introducción contra cada interface del software, lo que hizo que las pruebas de seguridad se convirtieran en una tarea pesada pero a la vez efectiva usando la implementación de pruebas manuales. En consecuencia, se optó por usar herramientas automatizadas para pruebas de seguridad que hagan las validaciones automáticamente, resultando en pruebas de seguridad más eficientes y resultados de calidad más altos.

Existen productos comerciales y de código abierto disponibles que fueron analizados para identificar si son apropiados para la organización, en este caso INEGI. Seleccionar una herramienta adecuada depende de varios factores incluyendo robustez y precisión de los casos de prueba incluidos en ellas, eficacia al probar los tipos de arquitectura importantes para la organización, personalización para modificar o agregar casos de prueba, calidad y usabilidad de los defectos para el equipo de desarrollo, etc.

Se tomaron en cuenta los comentarios del experto en seguridad así como del equipo de desarrollo y de aseguramiento de calidad en el proceso de selección y se revisaron los resultados con las partes interesadas.

5.8.2. Actividad 2. Integrar las pruebas de seguridad desde el proceso de desarrollo.

Con las herramientas para correr pruebas de seguridad automatizadas, los proyectos dentro de la organización ejecutaron pruebas de seguridad rutinariamente y revisaron los resultados durante el desarrollo. Para lograr que se fueran incrementando las pruebas de seguridad automatizadas con baja carga de trabajo, la herramienta fue configurada para que corra automáticamente cada noche o semanalmente dependiendo del proyecto y los defectos encontrados fueron enviados para revisión en cuanto se presentaron.

Llevar a cabo estas pruebas de seguridad tempranamente en la fase de diseño o requerimientos puede ser benéfico. Aunque tradicionalmente se usa para casos de pruebas funcionales, este tipo de enfoque, desarrollo impulsado por pruebas involucra la identificación y ejecución de pruebas de seguridad relevantes temprano en el ciclo de desarrollo, usualmente durante el diseño. Con la ejecución automática de casos de prueba, los proyectos entraron a la fase de implementación con un número de casos de prueba fallados para la funcionalidad aun no existente. La implementación fue completada cuando todos los casos de prueba pasaron. Esto proveyó un objetivo claro y por adelantado para los desarrolladores más tempranamente en el ciclo de desarrollo, de esta manera, disminuyendo el riesgo de retrasos en las liberaciones a producción debido a preocupaciones de seguridad o aceptación forzada de riesgos con tal de llegar a las fechas límites.

Para cada liberación a producción de cada proyecto, los resultados de las pruebas manuales y automatizadas fueron presentados a la gerencia y las partes interesadas del negocio para revisión. Si había defectos que no fueron sido analizados y permanecieran como riesgos aceptados para la liberación a producción, las partes interesadas y los gerentes de desarrollo trabajaron juntos para establecer un marco de tiempo concreto para solucionarlos.

Para esta fase las preguntas de control, que se muestran en la Tabla 11, se podían calificar considerando una escala de entre: 0.0 (No), 0.2 (Algo), 0.5 (La mitad) y 1.0 (La mayoría).

Tabla 11. Hoja de Trabajo de Evaluación - Pruebas de Seguridad Fase 2.

Hoja de Trabajo de Evaluación - Pruebas de Seguridad	Puntuación
¿Los proyectos usan automatización para evaluar los casos de prueba de seguridad?	
¿Los proyectos siguen un proceso consistente para evaluar y reportar pruebas de seguridad a los interesados?	

Las métricas de éxito de acuerdo con SAMM son:

- El número de proyectos con pruebas de seguridad y visto bueno de las partes interesadas en los últimos seis meses debe ser mayor a 50%.
- El número de proyectos con acceso a herramientas automatizadas para pruebas de seguridad en el último mes debe ser mayor al 80%.

5.9. Pruebas de Seguridad - Fase 3

En esta última fase de pruebas de seguridad y del módulo de Verificación de SAMM el objetivo fue, hacer obligatorio el uso de pruebas de seguridad específicas a cada aplicación para asegurar seguridad básica antes de cada despliegue.

5.9.1. Actividad 1. Emplear herramientas automatizadas de pruebas de seguridad para cada aplicación específica.

Ya sea a través de personalización de las herramientas de seguridad, mejoras a las herramientas genéricas de ejecución de casos de prueba o construcción de arneses de prueba, los equipos de proyecto analizaron y fueron iterando formalmente entre los requerimientos de seguridad y construyendo un conjunto

de puntos de control automatizados para probar la seguridad de la lógica de negocio implementada.

Adicionalmente, muchas herramientas de seguridad automatizadas pueden ser mejoradas en precisión y profundidad de cobertura si son personalizadas con más detalle acerca de las interfaces especificas del software del proyecto que se está probando. Más adelante, las preocupaciones especificas a la organización que surgen de cumplimientos o estándares técnicos pueden ser codificados como un conjunto de pruebas centrales reusables para hacer la recolección de datos de auditoria y la visibilidad de la gerencia por proyecto más sencilla.

Los equipos de proyecto se enfocaron en la construcción de casos de prueba granulares basados en la funcionalidad del software y el equipo a nivel de organización liderado por un auditor de seguridad se enfocó en la especificación de pruebas automatizadas para cumplimiento y estándares internos.

5.9.2. Actividad 2. Establecer puntos de control para pruebas de seguridad.

Para prevenir que el software sea liberado a producción con errores de seguridad fáciles de encontrar, se identificó un punto particular en el ciclo de vida de desarrollo de software donde un conjunto establecido de casos de prueba de seguridad debieron pasar para poder hacer la liberación a producción para ese proyecto. Esto estableció una línea base para los tipos de pruebas de seguridad que se espera que todos los proyectos pasen.

Dado que agregar demasiados casos de prueba al inicio puede resultar en una burbuja de gastos sobrecargada, se comenzó por elegir uno o dos problemas de seguridad e incluir una amplia variedad de casos de prueba para cada uno con la expectativa de que ningún proyecto pudiera pasar si alguna prueba fallaba. Con el tiempo, esta línea base fue mejorada al seleccionar problemas de seguridad adicionales y sus casos de prueba correspondientes.

En la mayoría de los proyectos este punto de control de pruebas de seguridad se verificó hacia el final de la fase de implementación o fase de pruebas, ocurriendo siempre antes de la liberación a producción.

Para los sistemas heredados o proyectos inactivos, un proceso de excepción fue creado para permitir que continúen operaciones, pero con un marco de trabajo explícitamente asignado para mitigación de defectos. Las excepciones fueron limitadas a no más del 20% de todos los proyectos como lo indica SAMM.

Para esta fase las preguntas de control, que se muestran en la Tabla 12, se podían calificar considerando una escala de entre: 0.0 (No), 0.2 (Algo o por equipo), 0.5 (La mitad o a lo largo de la organización) y 1.0 (La mayoría o proceso integrado).

Tabla 12. Hoja de Trabajo de Evaluación - Pruebas de Seguridad Fase 3.

Hoja de Trabajo de Evaluación - Pruebas de Seguridad	Puntuación
¿Los casos de prueba de seguridad son generados comprensivamente para la lógica de cada aplicación específica?	
¿Existe un mínimo de seguridad base para pruebas de seguridad?	

SAMM indica las métricas de éxito mínimas para esta fase:

- El número de proyectos usando personalización de pruebas debe ser mayor al 50%.
- El número de proyectos que pasen todas las pruebas de seguridad en los últimos seis meses debe ser mayor al 75%.

5.10. El Uso de ZAP para detección de vulnerabilidades y Pruebas de Penetración.

Como ya se mencionó anteriormente, nos enfocaremos en la etapa de Verificación del marco de trabajo SAMM, más específicamente en el apartado de Pruebas de Seguridad, para lo cual necesitaremos verificar una serie de buenas prácticas, recomendaciones o lista de puntos de control a seguir para evitar que los atacantes se aprovechen de las posibles vulnerabilidades encontradas en las aplicaciones Web 2.0, se aplicará una prueba de penetración de ejemplo, usando distintas herramientas de OWASP disponibles en su sitio web para crear un laboratorio o ambiente de pruebas y simular el ataque, como detectarlo y confirmarlo.

Corriendo la herramienta ZAP

En el siguiente ejemplo analizaremos paso a paso una prueba de penetración para Inyección SQL, comenzaremos por iniciar la aplicación de OWASP ZAP como se muestra en la Imagen 7.

Y usando una aplicación Web hecha en PHP y MySQL exclusivamente para ayudar a profesionales de seguridad a probar sus habilidades y herramientas en un ambiente legal controlado, construido específicamente para este propósito. DVWA ayuda a los desarrolladores Web a entender mejor los procesos para asegurar aplicaciones Web y ayudar a maestros y estudiantes a enseñar o aprender a usar la seguridad en aplicaciones Web en un ambiente de salón de clases.

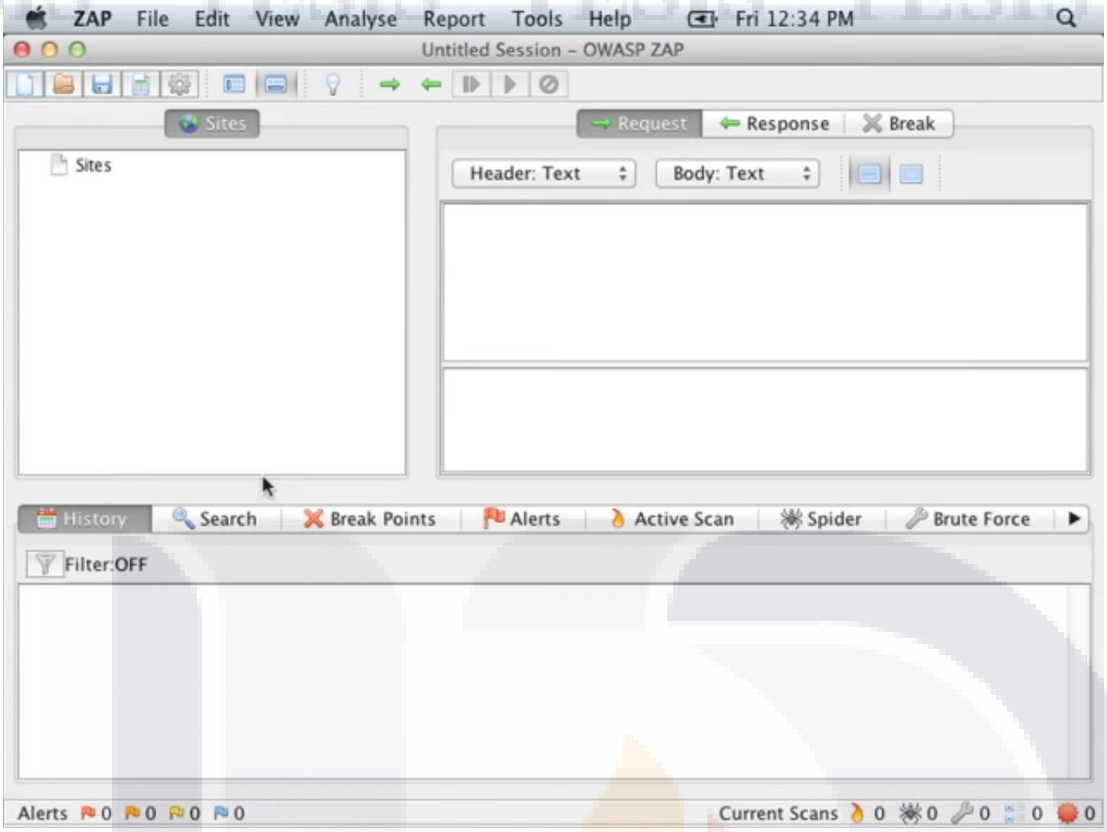


Imagen 7. Herramienta ZAP de OWASP, Pantalla Principal.

La siguiente imagen (Imagen 8) es la pantalla principal de la aplicación de pruebas DVWA que instalamos para poder realizar un ejercicio de prueba de penetración en un ambiente seguro. Para comenzar, debemos configurar el proxy del navegador en este caso Firefox, para que todo el tráfico pase por ZAP y sea analizado automáticamente. Vamos al menú preferencias de Firefox.

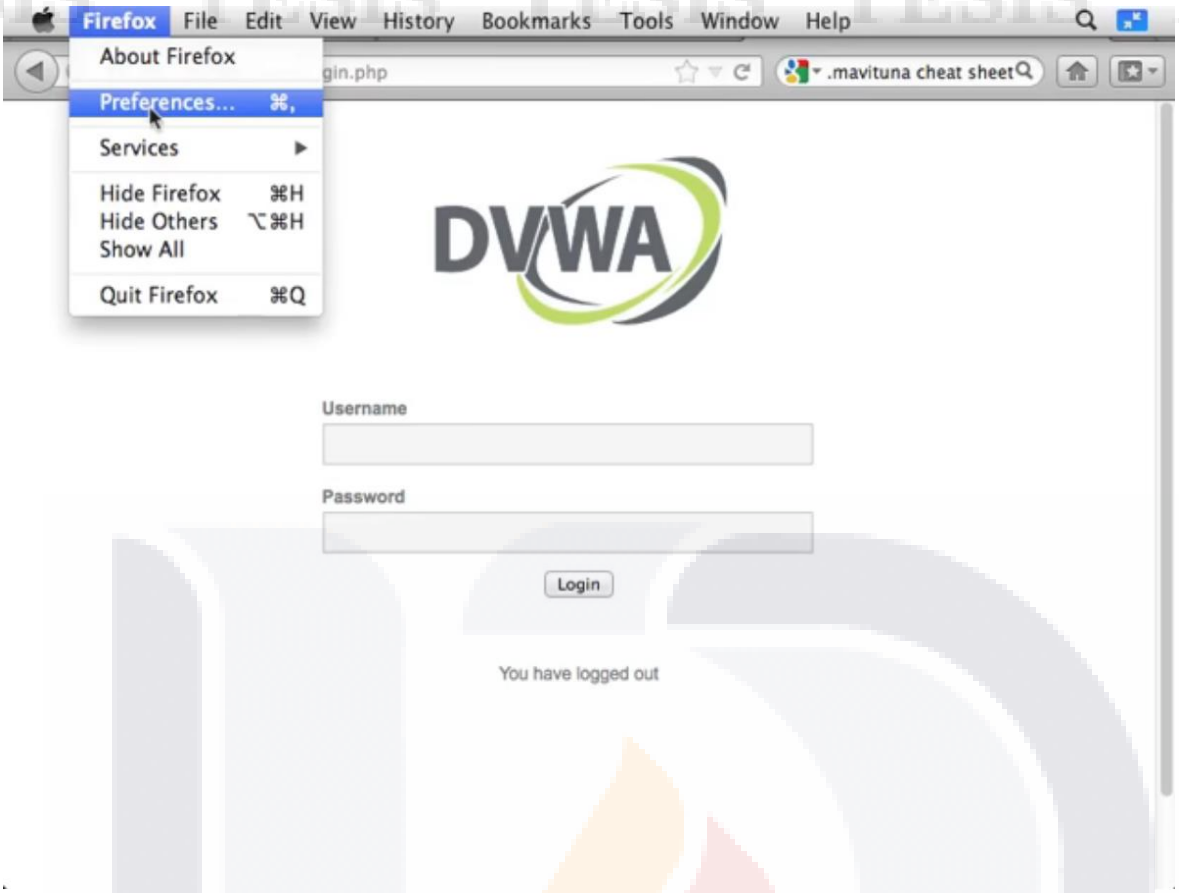


Imagen 8. Configurar Preferencias del Navegador Firefox.

Se debe configurar el proxy del navegador que usamos para que todo el tráfico pase por ZAP, para esto iremos a la Configuración Avanzada (Advanced) y al apartado Redes (Network) de nuestro navegador.

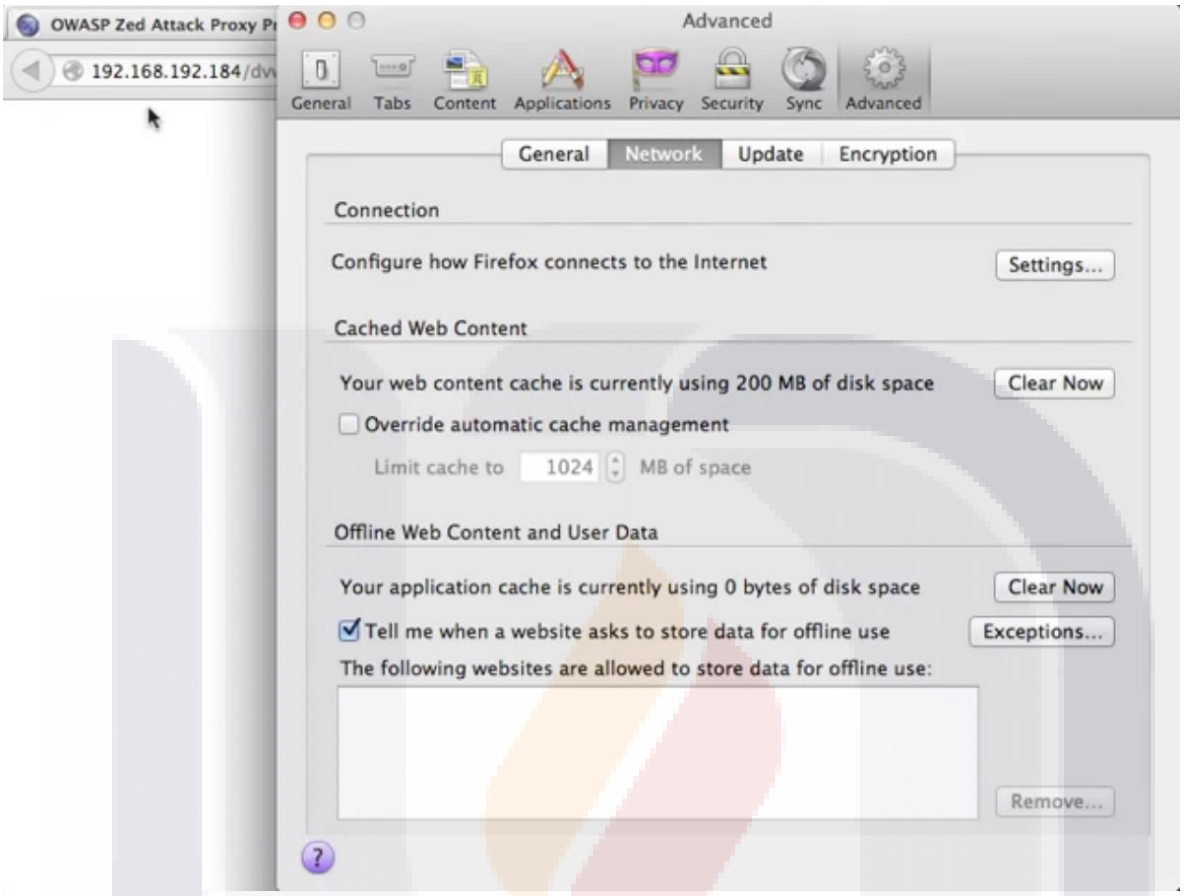


Imagen 9. Apartado de Redes en Preferencias del Navegador Firefox.

A continuación, damos clic en el botón de Ajustes (Settings) y seleccionamos la opción de Configuración Manual de Proxy e introducimos local host, puerto 8080 y damos clic en Ok.

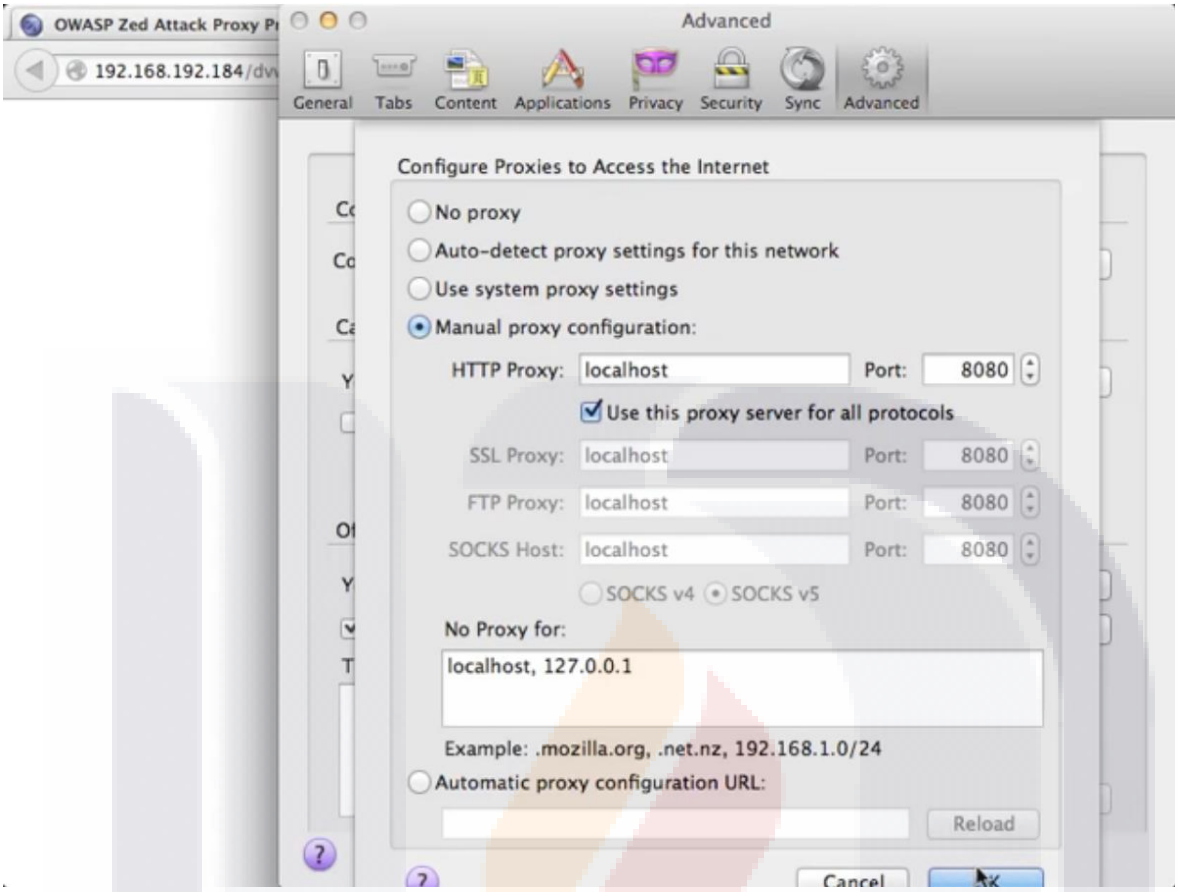


Imagen 10. Ajustes para Configurar el Proxy Manualmente en Navegador Firefox.

Una vez que está configurado el proxy manualmente, todo el tráfico pasará a través de la herramienta ZAP para ser analizado automáticamente por la herramienta, continuamos en la aplicación vulnerable para pruebas, iniciaremos sesión con el usuario admin, contraseña o password: admin. Esta acción nos llevará a la página de bienvenida como se muestra a continuación en la Imagen 13.

Imagen 11. Pantalla Principal de la Aplicación DVWA después de hacer Login.

En esta pantalla debemos verificar en la parte inferior izquierda que el nivel de seguridad sea Bajo (Low).

Imagen 12. Verificar el Usuario y Nivel de Seguridad en la Aplicación DVWA.

Ahora comenzaremos a usar la aplicación e iremos al apartado de SQL Injection localizado en la parte izquierda, introducimos un número “1” en el campo User ID y damos clic en el botón Submit.

Inmediatamente después de presionar la tecla Enter o dar clic en el botón Submit, la aplicación despliega resultados mostrados en rojo en la Imagen 15, al introducir el valor 1, la aplicación que está diseñada para estudiar los ataques, detecta una inyección SQL y trae como resultado información del usuario que tiene su sesión iniciada en la aplicación.

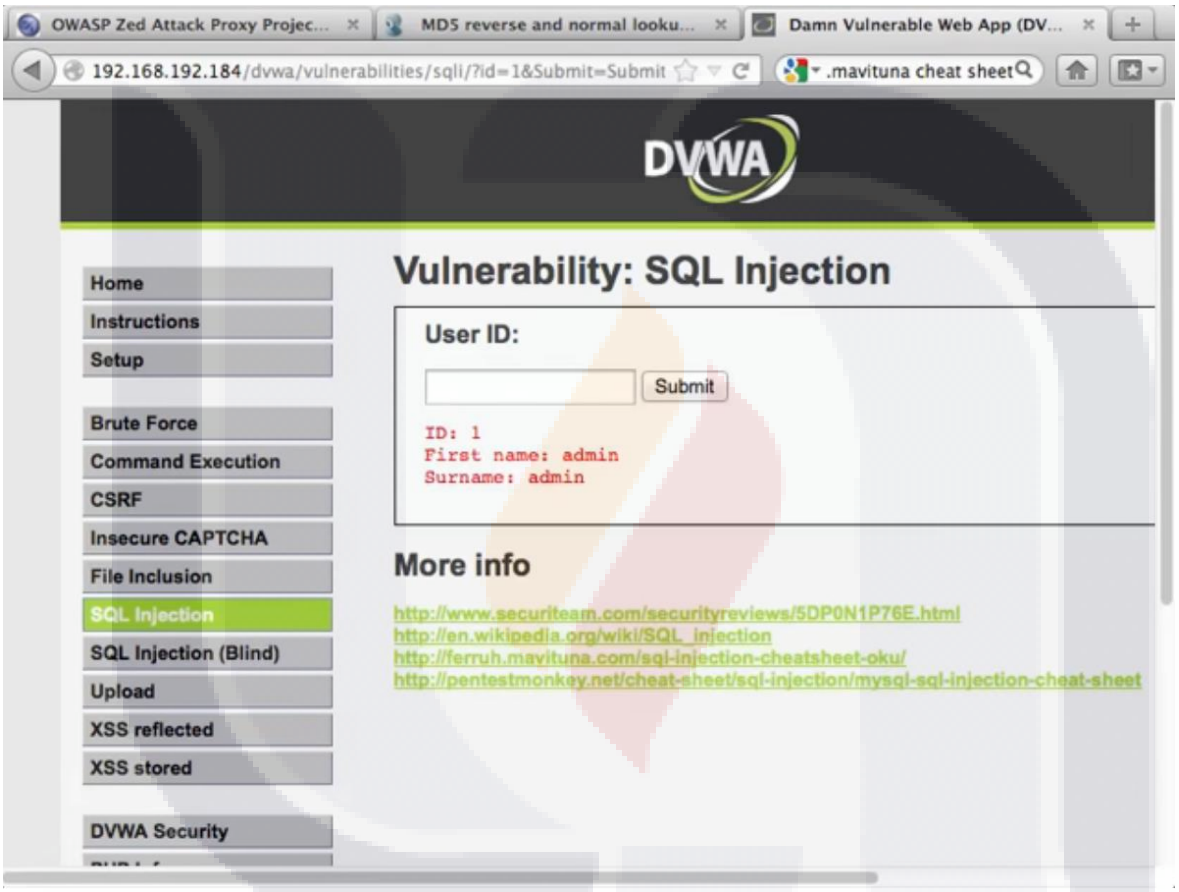


Imagen 13. Apartado SQL Injection en la Aplicación DVWA.

Ahora, iremos a la sección de Command Execution, introducimos la dirección 127.0.0.1 y damos clic en el botón Submit. Al dar clic la aplicación trae el resultado mostrado en rojo en la siguiente imagen (Imagen 17) del ping hecho en el apartado de Ejecución de Comandos.

Imagen 14. Resultados de Command Execution en la Aplicación DVWA.

Mientras tanto, todas las acciones que hemos realizado, se han ido guardado en el fondo en la herramienta ZAP, esto se logró cuando configuramos el proxy para que todo el tráfico, es decir todas las páginas que visitamos y acciones que realizamos en la maquina pasen por las herramienta ZAP para ser analizadas.

En ZAP en el apartado de Sites en la parte superior izquierda, se registra cada página en una carpeta distinta, en la parte inferior podemos observar todas las operaciones post y get que se van registrando con su dirección URL correspondiente, estado, milisegundos que le toma acceder y el tipo de acceso ya sea formulario o script ejecutados.

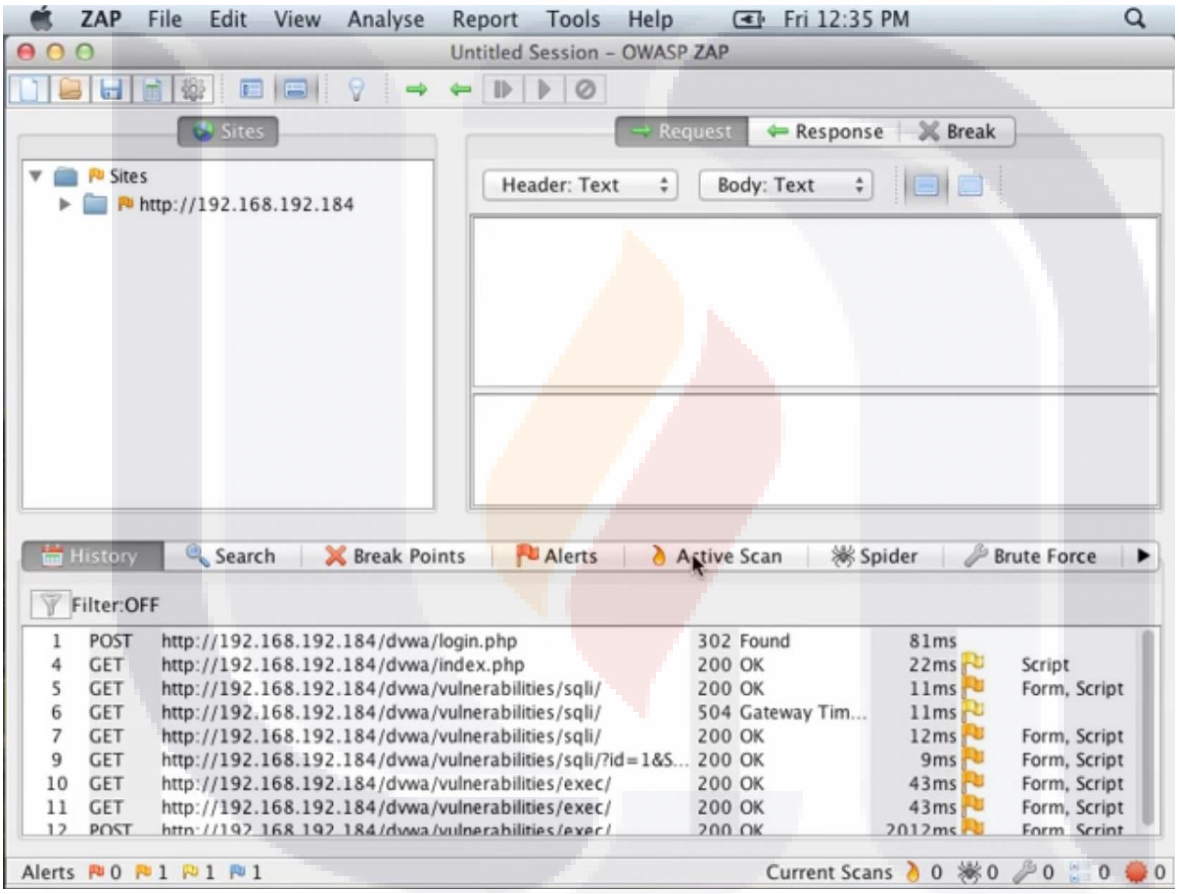


Imagen 15. Registro Automático de ZAP en Segundo Plano.

En la herramienta ZAP, vamos al apartado de Sites en la parte izquierda superior y abrimos el árbol de directorios y seleccionamos la carpeta dvwa como se muestra en la siguiente imagen (Imagen 19) para analizar lo que ha registrado y encontrado hasta el momento, lo cual podemos observar como un pequeño resumen ilustrativo con las banderas localizadas al fondo a la izquierda de la aplicación, organizadas por color, las rojas representan alertas altas, naranjas medias y amarillas bajas.

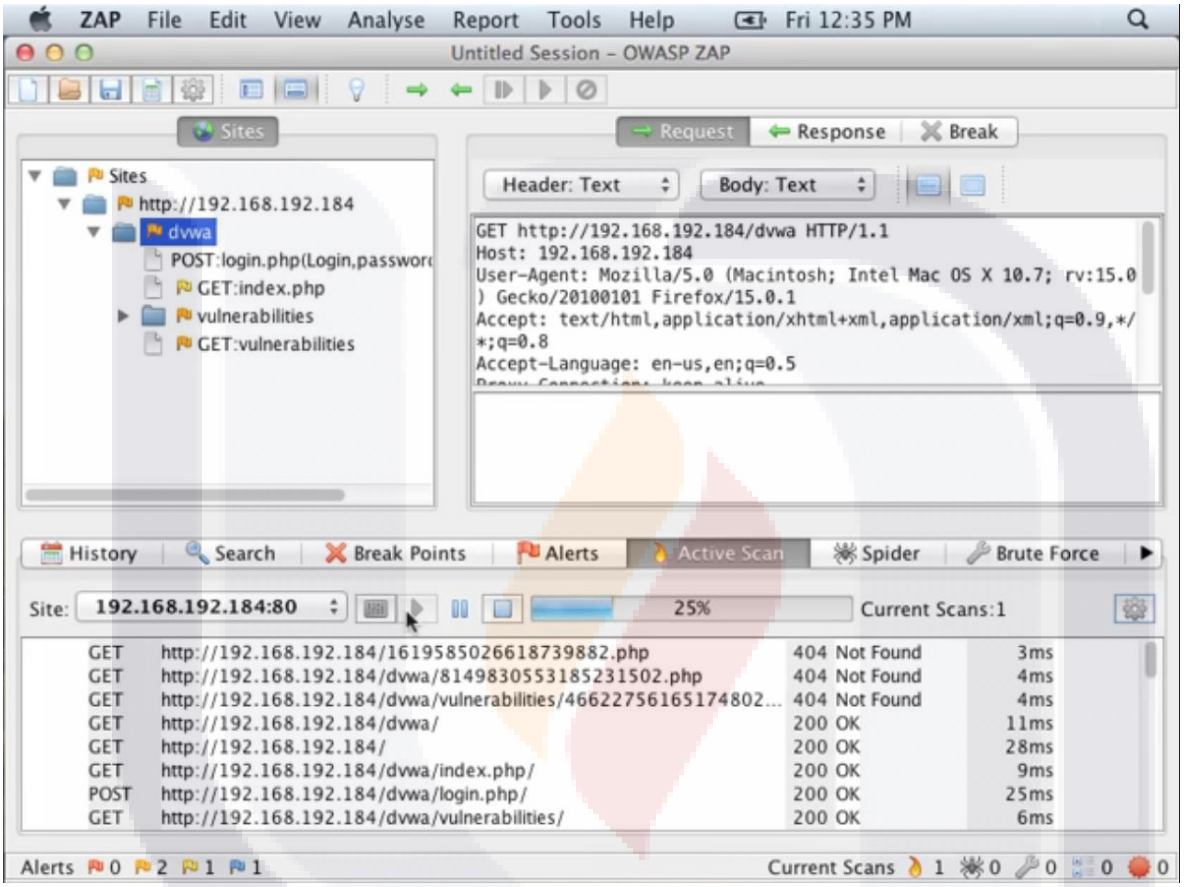


Imagen 16. Apartado Active Scan en ZAP.

Hay distintas opciones que se pueden configurar en ZAP en el menú herramientas, en el apartado de opciones como se puede ver en la Imagen 20.

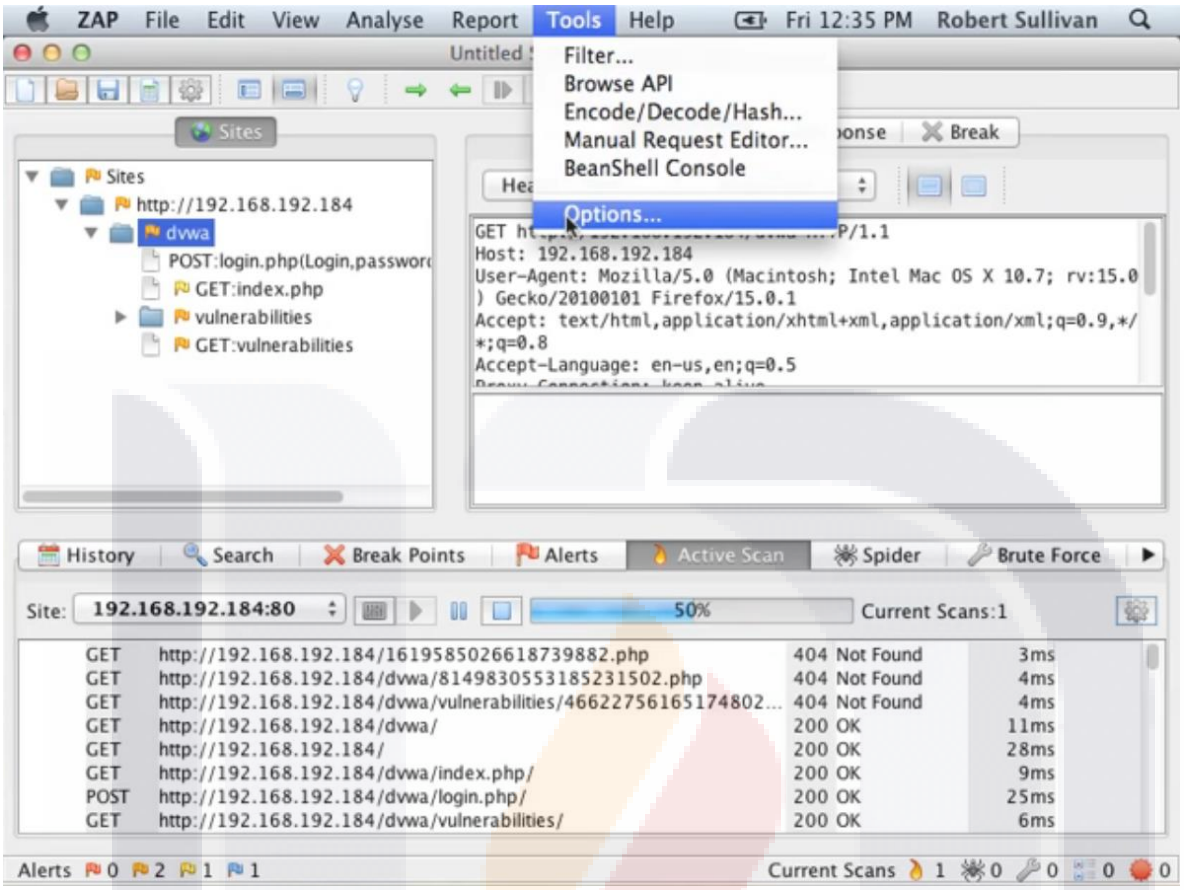


Imagen 17. Menú Herramientas > Opciones en ZAP.

En el apartado de conexión que podemos visualizar en la imagen 21, por ejemplo, se puede configurar si es necesario un Proxy Corporativo con el fin de analizar el tráfico de una organización que tiene su propio proxy para salir a la red.

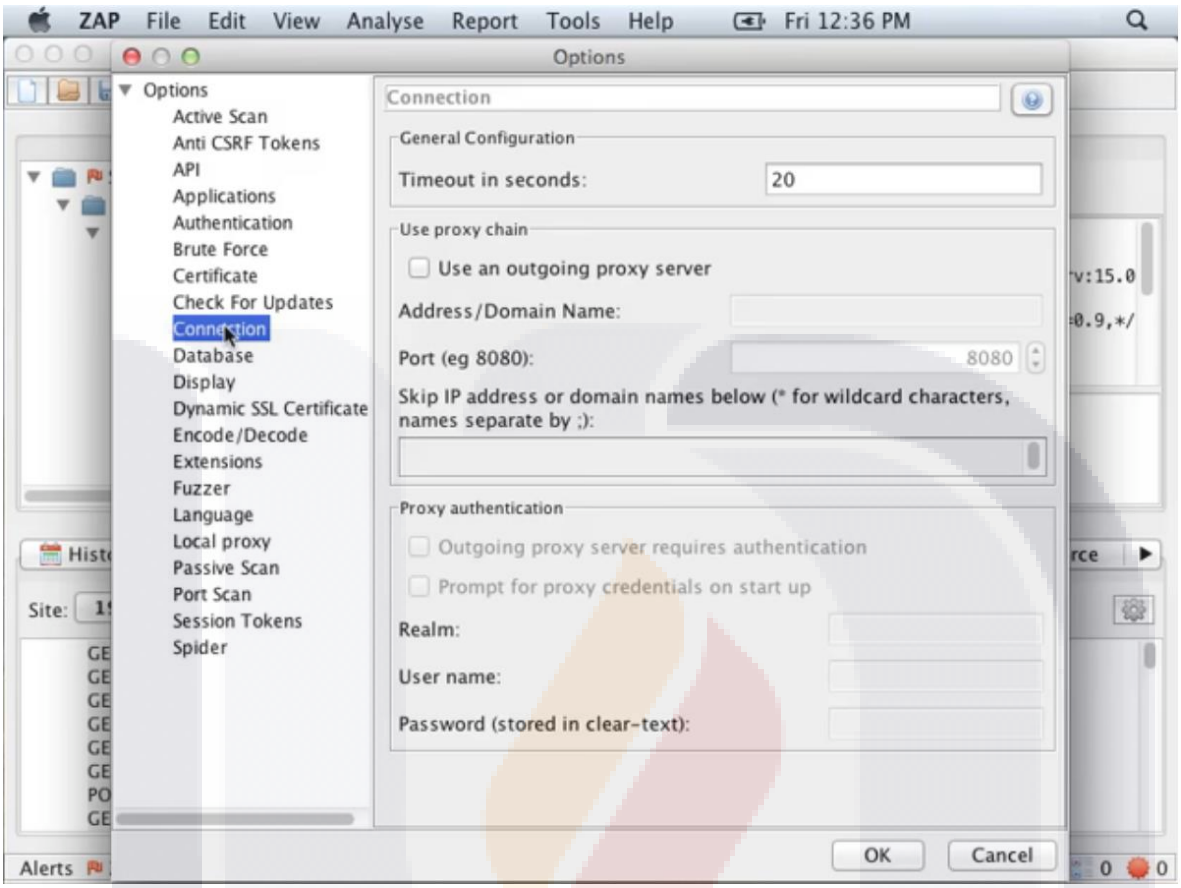


Imagen 18. Apartado Conexión en el menú Herramientas > Opciones de ZAP.

En el apartado de alertas, se encuentran los ataques que detectó la herramienta ZAP, en este caso, cuando se introdujo el número 1 en el campo User ID, lo toma como un posible SQL Injection, esto lo vemos en la Imagen 22.

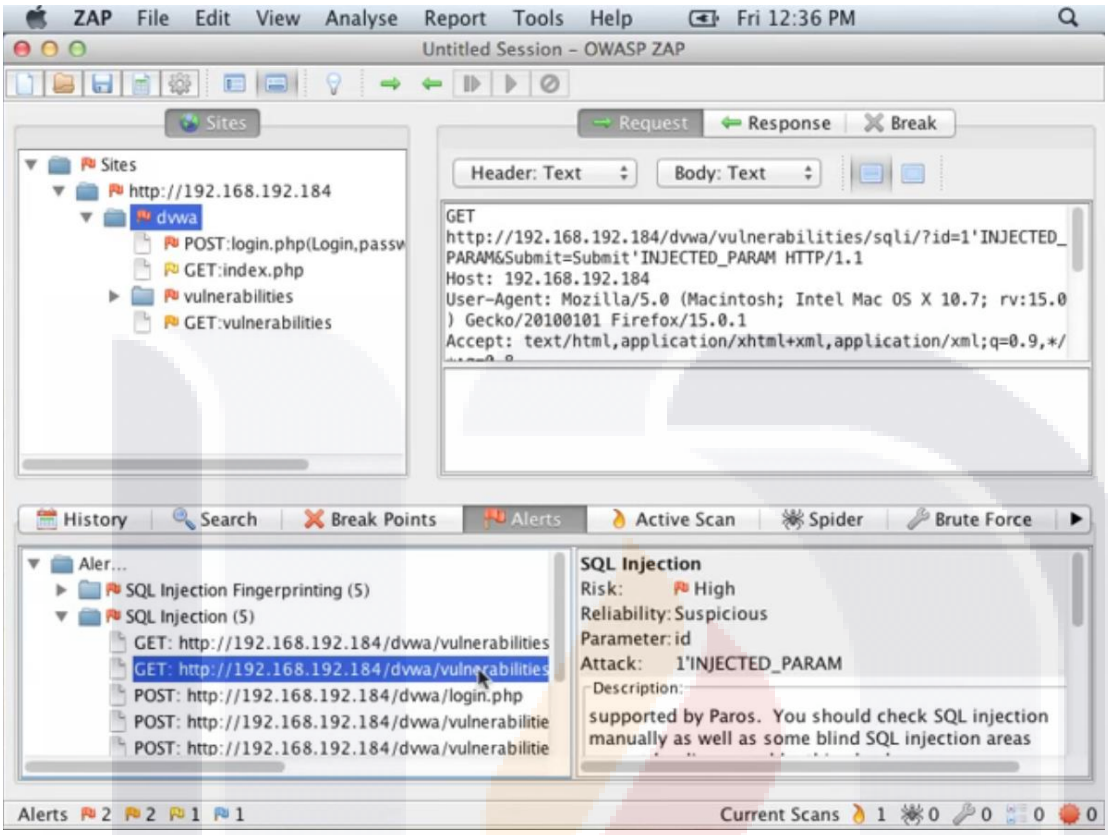


Imagen 19. Apartado Alertas en la Aplicación ZAP.

Al analizar la Alerta detectada podemos observar que se realizó una Inyección SQL a partir del valor 1 que introdujimos en el campo UserID (Imagen23).

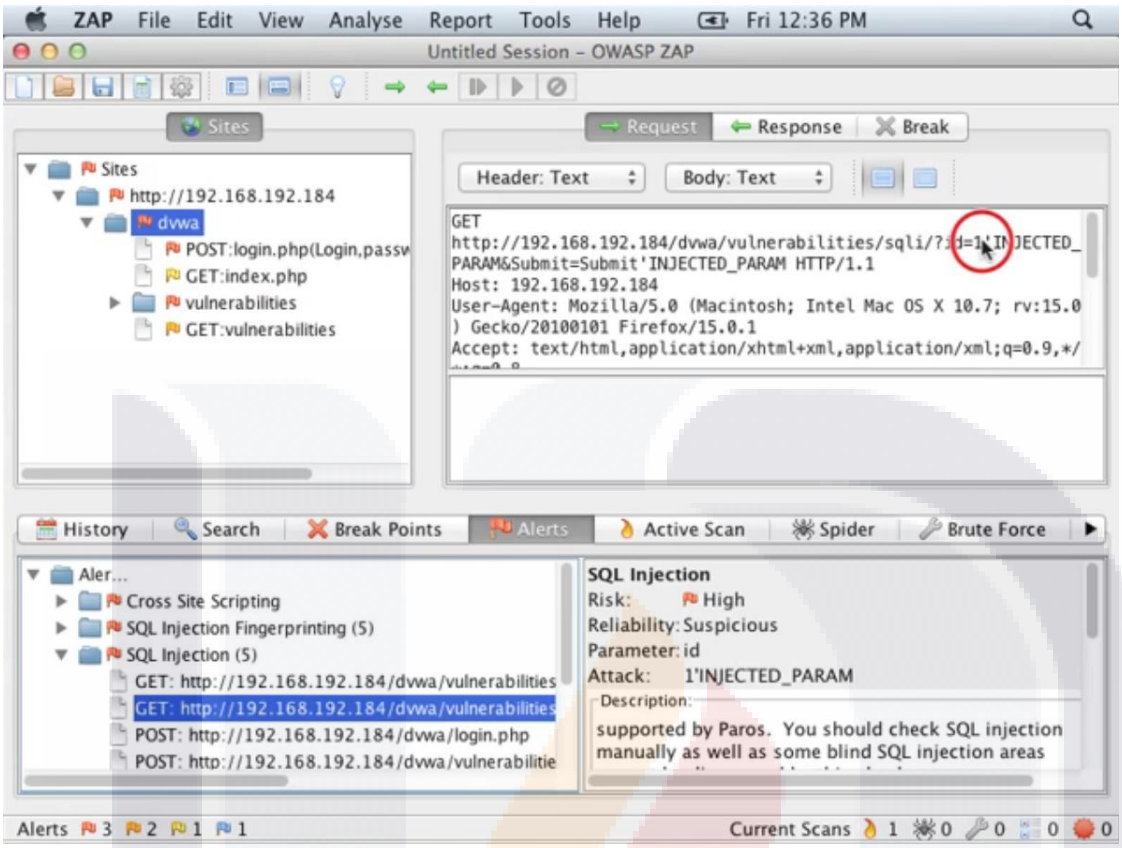


Imagen 20. Ataque tipo Inyección SQL detectado en ZAP.

Analizaremos más a fondo usando el Fuzzer, accedemos dando clic derecho en el parámetro de ataque detectado por ZAP como se muestra en la Imagen 24.

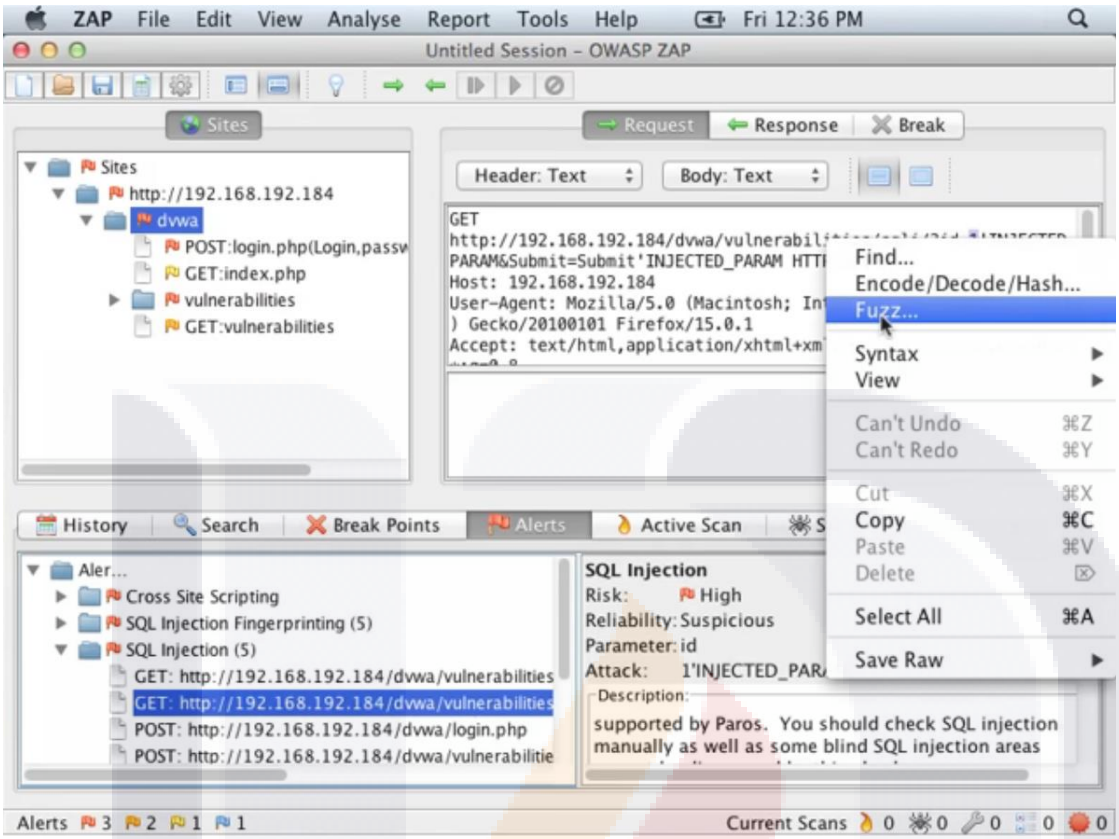


Imagen 21. Menú Secundario del Parámetro de Ataque detectado por ZAP.

Al hacer esto, la aplicación despliega una ventana donde podemos seleccionar la Categoría y el archivo que utilizará el Fuzzer (Imagen 25).

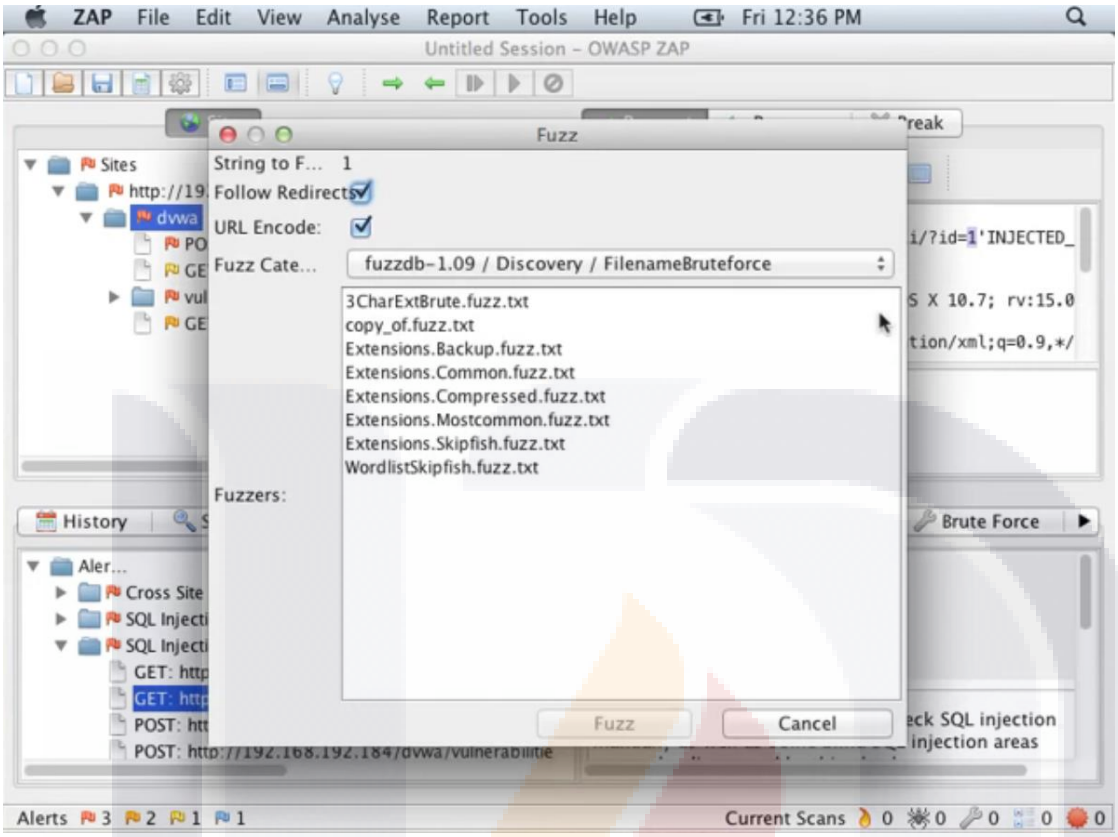


Imagen 22. Ventana del Fuzzer en la Herramienta ZAP.

En la imagen 26 seleccionamos SQL Injection de la lista Fuzz Category.

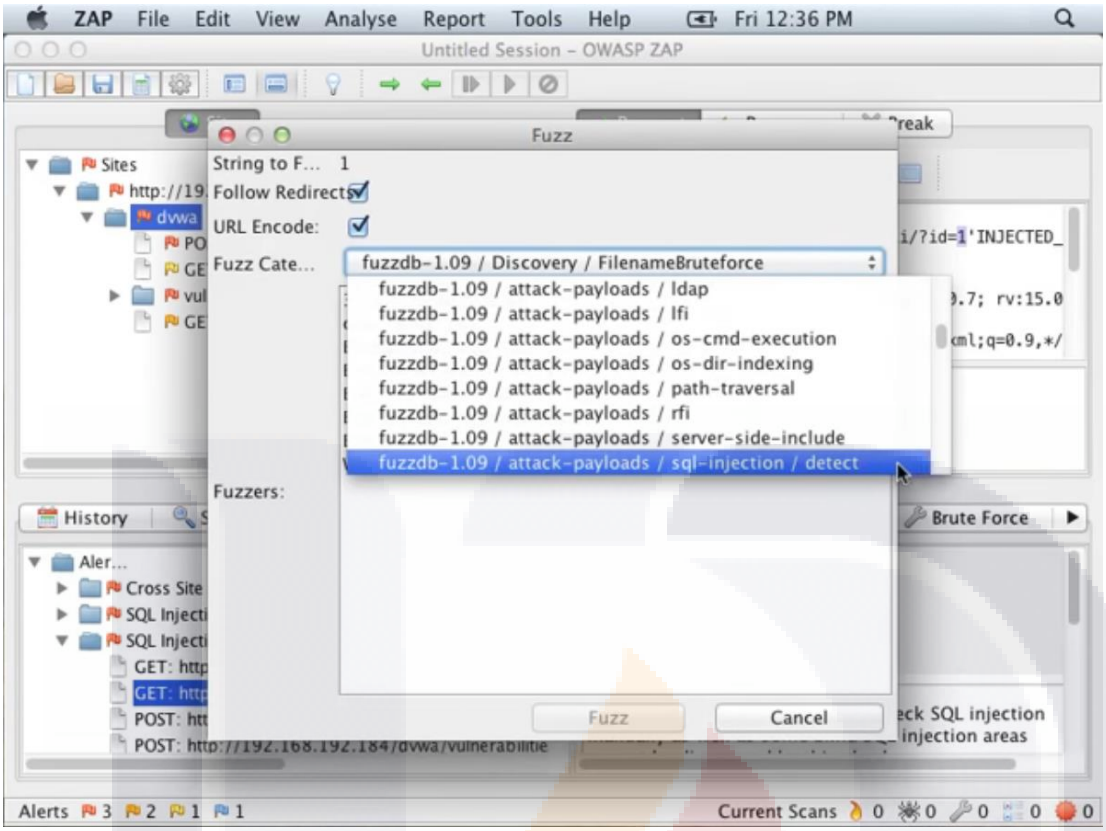


Imagen 23. Opciones Disponibles para SQL Injection en la lista Fuzz Category.

Y seleccionamos el archivo MySQL_Fuzz.txt, y a continuación damos clic en el botón Fuzz para comenzar el proceso como se muestra a continuación en la Imagen 27.

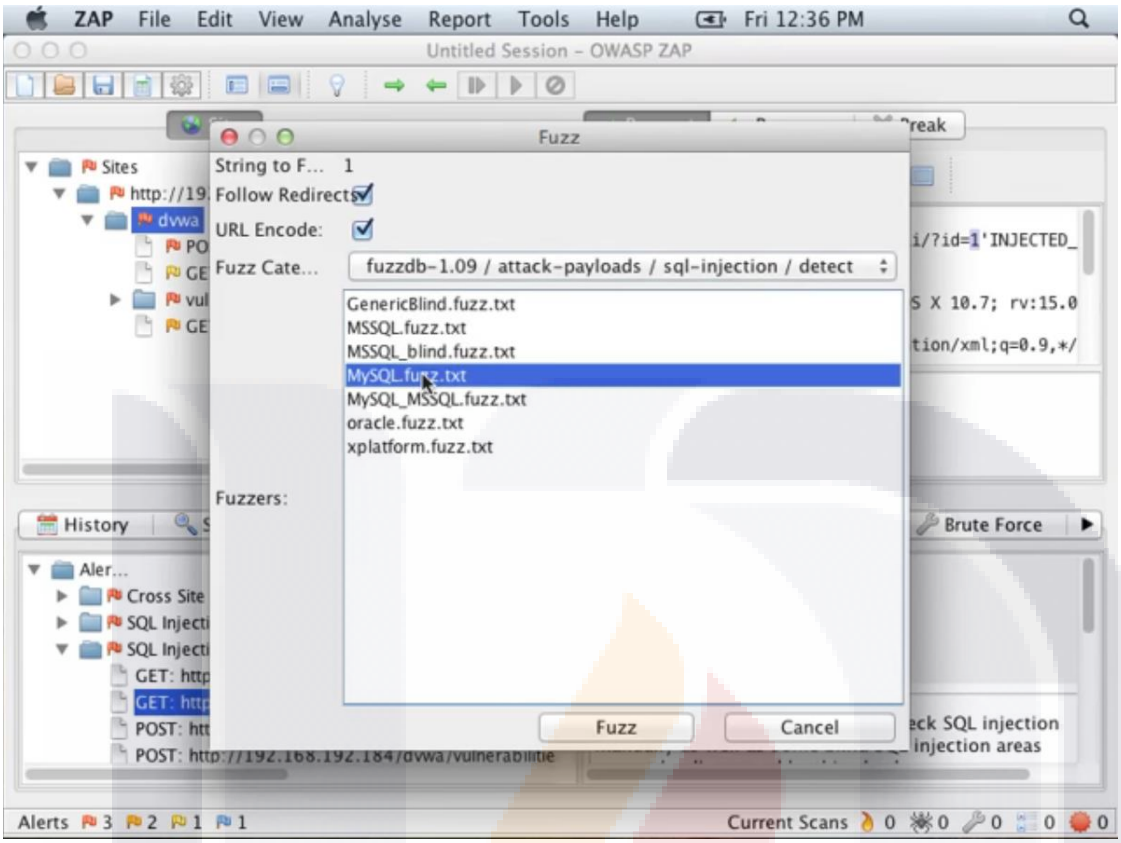


Imagen 24. Selección de Archivo a Utilizar con el Fuzzer de ZAP.

Al dar clic en el botón Fuzz, la herramienta comienza el proceso, podemos observar el avance en la barra inferior junto con el botón para detener (stop) o pausar (Imagen 28).

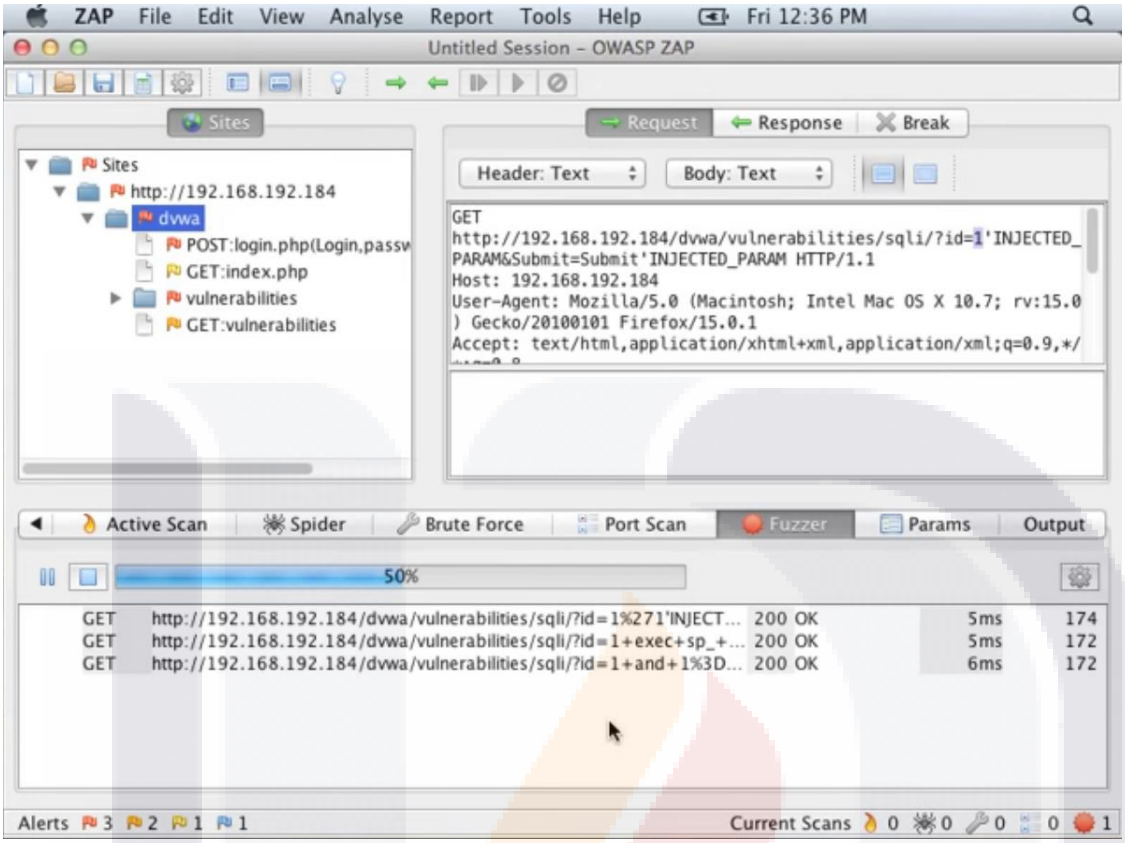


Imagen 25. Proceso de Fuzz de la Herramienta ZAP.

En la siguiente imagen (Imagen 29) podemos ver el archivo que contiene secuencias de comandos que observamos en el OWASP Top 10 para simular el ataque.

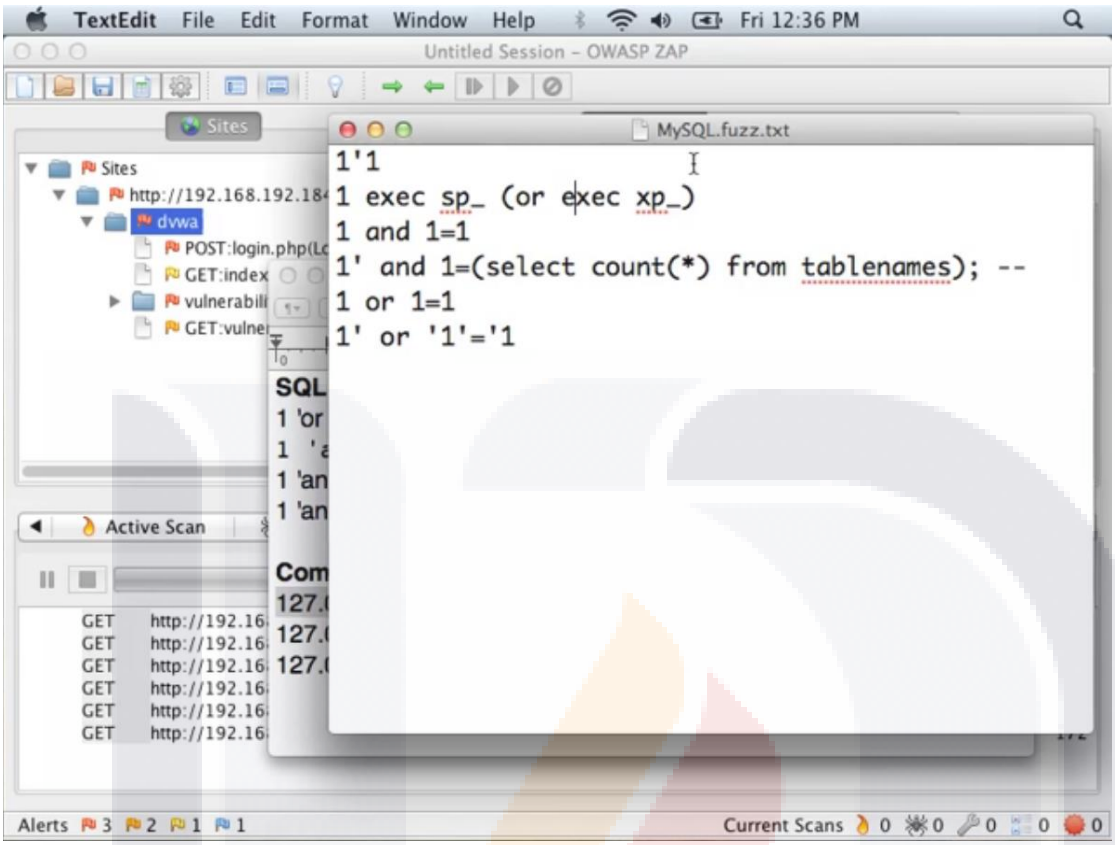


Imagen 26. Contenido del Archivo MySQL_fuzz.txt de la Herramienta ZAP.

Ahora vamos a analizar los resultados del proceso del Fuzzer de ZAP como se muestra en la Imagen 30.

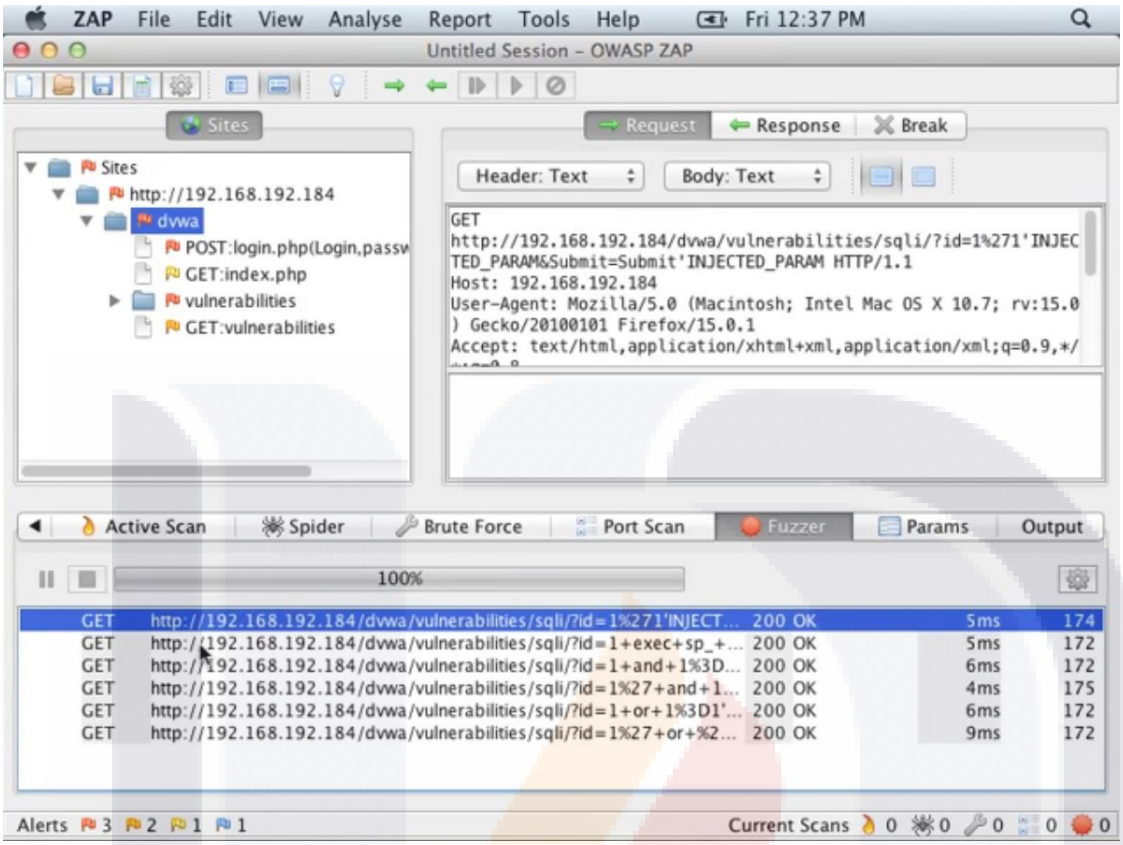


Imagen 27. Resultados Proceso Fuzzer de ZAP.

Vamos al Apartado de Response para analizar la respuesta a la petición enviada por el Fuzzer de ZAP (Imagen 31).

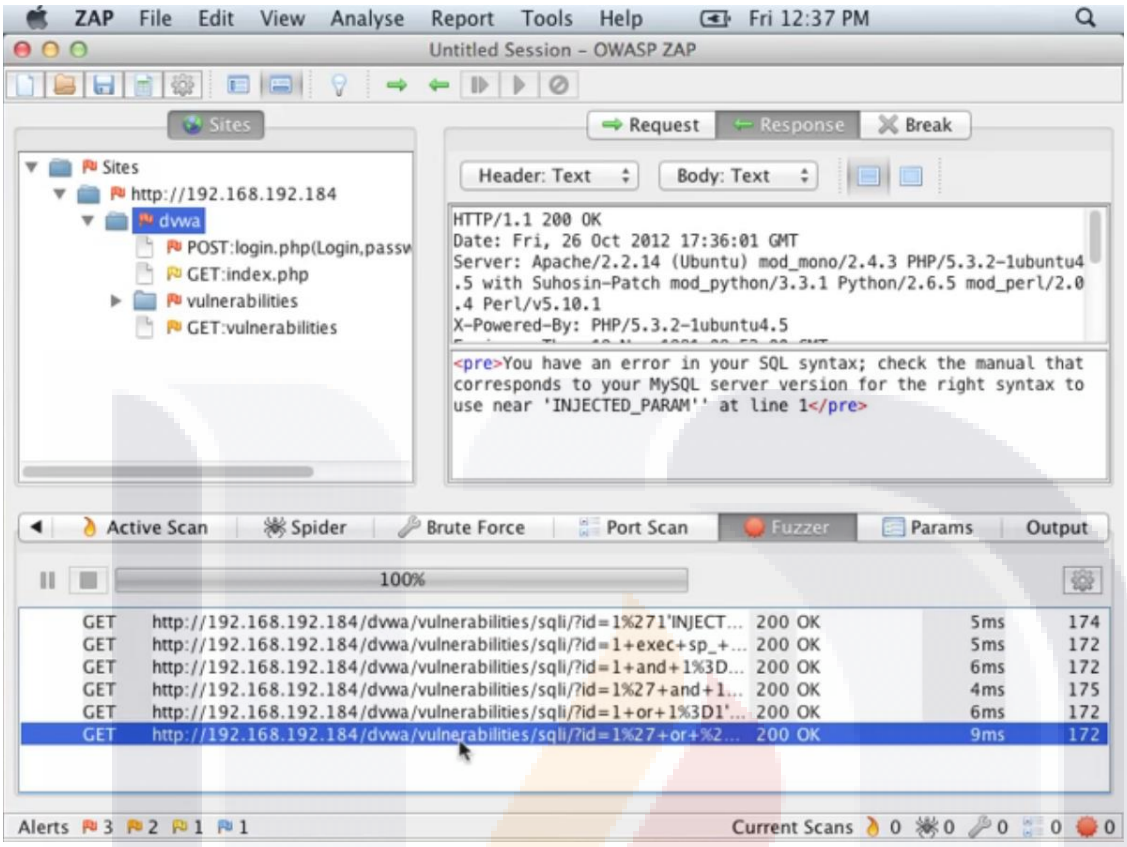


Imagen 28. Apartado Response como resultado del proceso Fuzz de ZAP.

Usando el comando seleccionado en la siguiente imagen (Imagen 32) trataremos de obtener los usuarios y passwords o contraseñas.

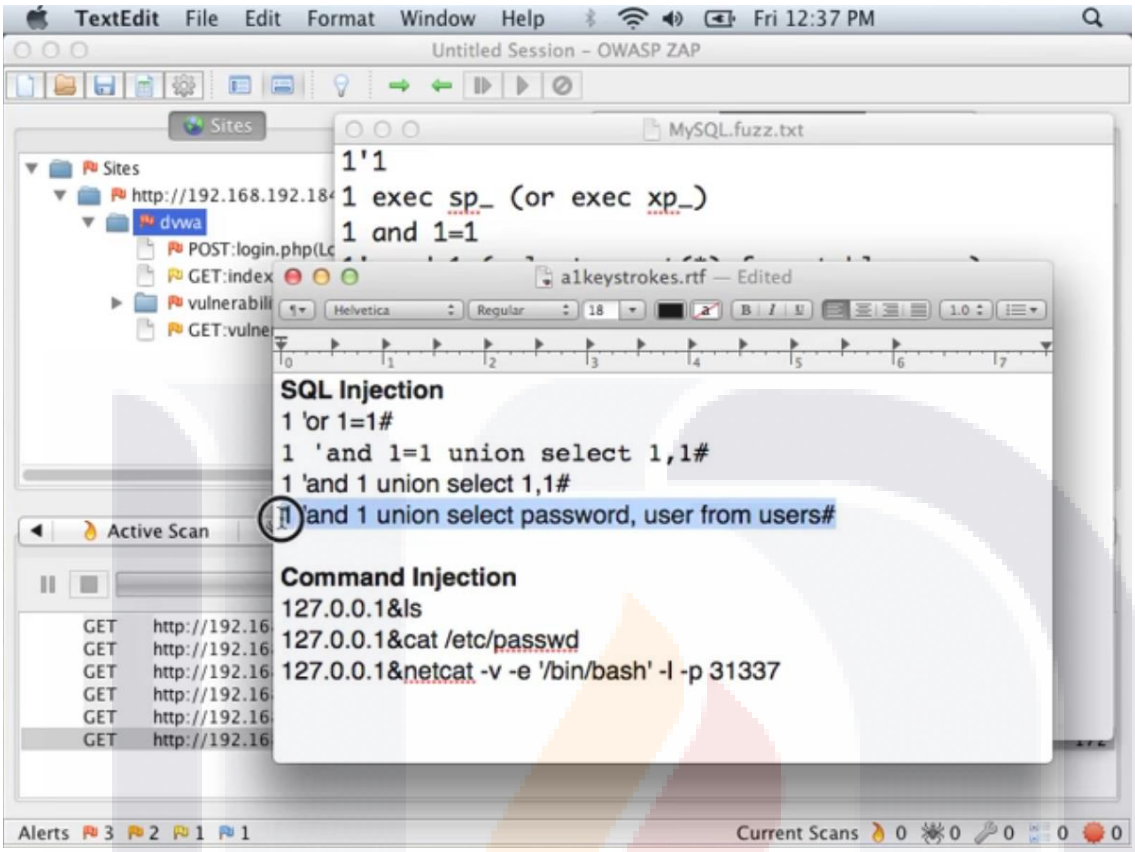


Imagen 29. Comando para obtener usuarios y contraseñas de la Aplicación DVWA.

Introducimos el comando en el campo User ID de la aplicación DVWA y presionamos Enter o clic en el botón Submit como se muestra en la Imagen 33.

Imagen 30. Introducción del Comando en el Campo User ID de la aplicación DVWA.

Con esta información podemos descifrar si el usuario/contraseña para admin usa hash md5 (Imagen 34).

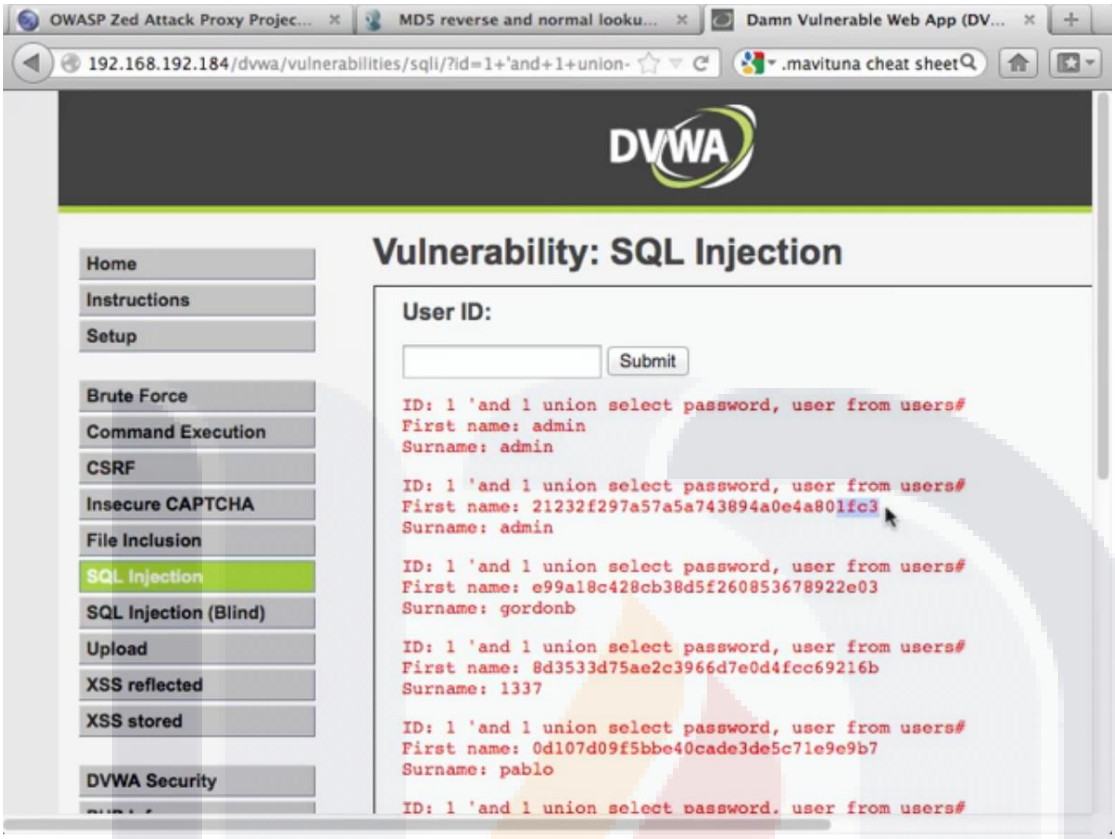


Imagen 31. Resultados después de Insertar el Comando en el Campo User ID del Apartado SQL Injection en la Aplicación DVWA.

Con esta información trataremos de descifrar la contraseña usando el método Hash, para esto, vamos al menú herramientas (mostrado en la Imagen 35) y después la opción Encode/Decode/Hash en la herramienta ZAP.

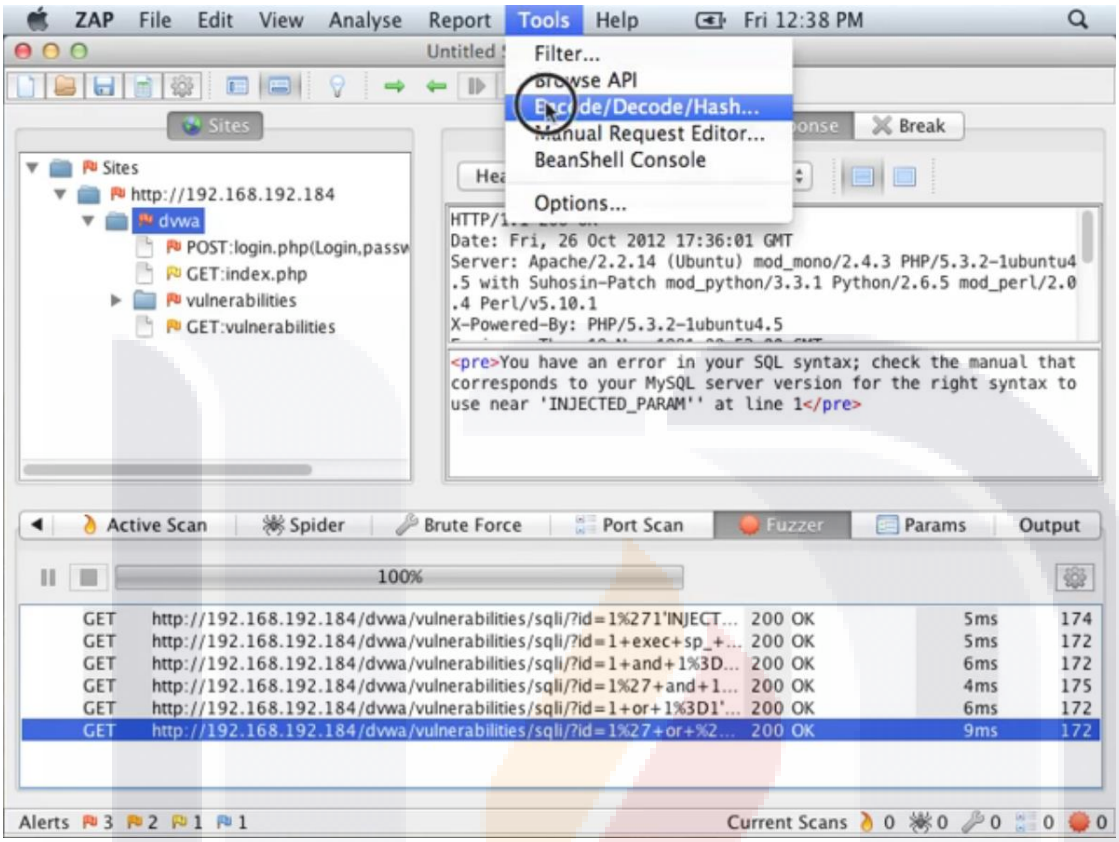


Imagen 32. Opción Encode/Decode/Hash en el menú Herramientas de ZAP.

Damos click en el apartado Hash y despúes escribimos en el campo “Text to be encoded/decoded/hashed” el usuario admin e inmediatamente muestra los resultados del Hash en los campos SHA1 Hash y MD5 Hash, que es el metodo que nos interesa como se muestra en la Imagen 36.

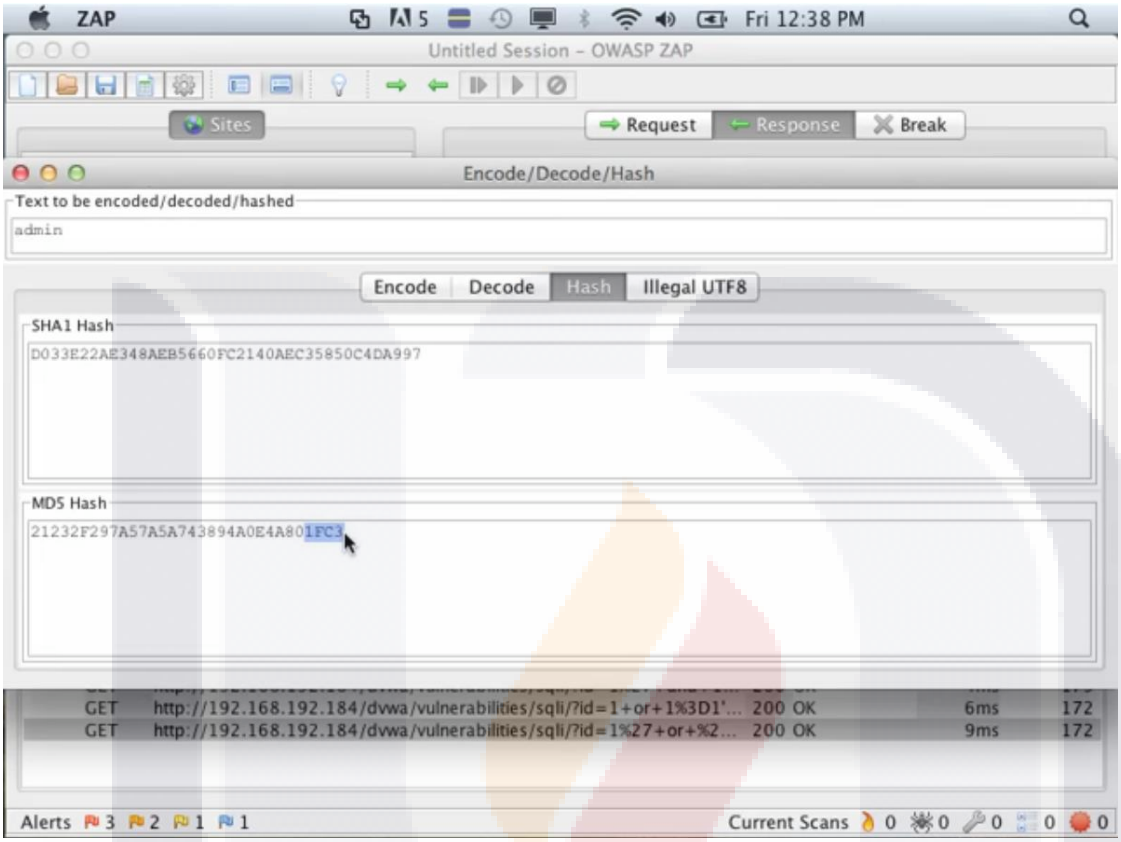


Imagen 33. Resultados Hash para el usuario Admin en la Herramienta ZAP.

Intentaremos ahora con el password hash para el usuario gordonb, obtenido después de introducir el comando SQL Injection en la Aplicación DVWA en la Imagen 37.

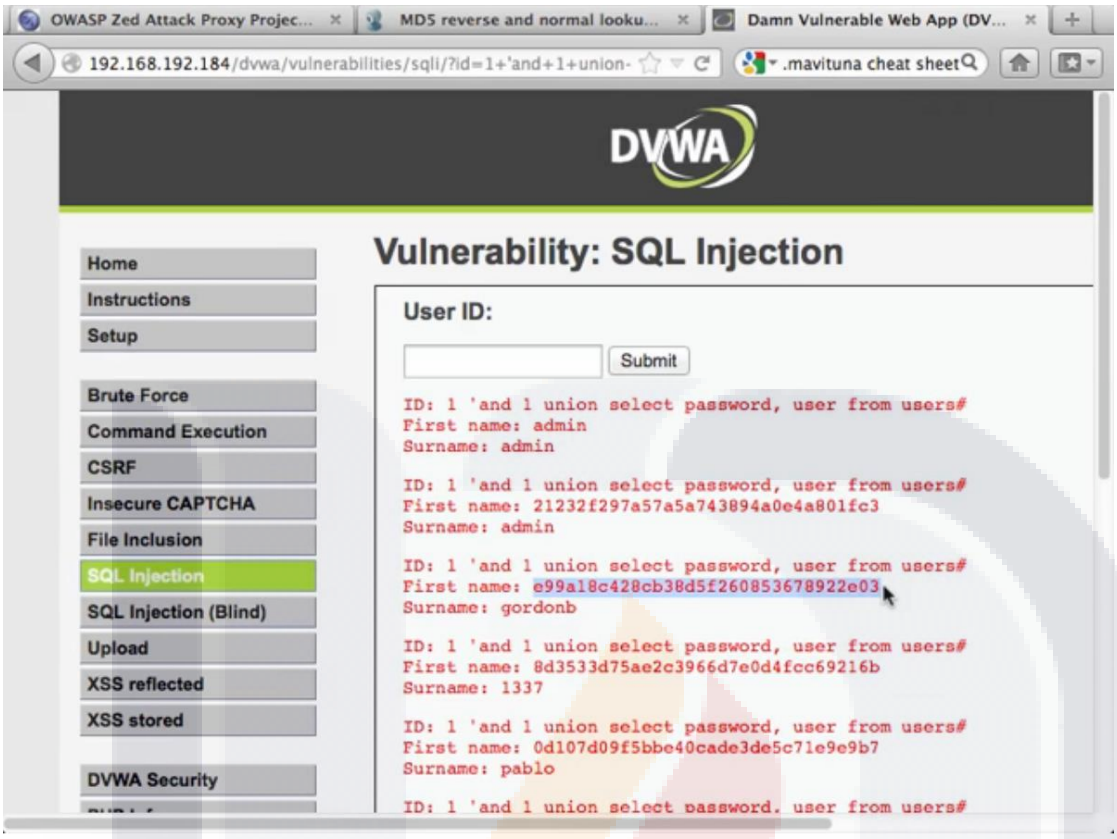


Imagen 34. Dato Hash para el usuario gordonb.

Cerramos la sesión del usuario admin e Iniciaremos sesión nuevamente pero con el usuario gordonb como se muestra en la Imagen 38. Verificamos que la sesión sea para el usuario gordonb, en la parte inferior izquierda Username y a la derecha del apartado About el mensaje “You have logged in as ‘gordonb’ que podemos apreciar en la Imagen 39.

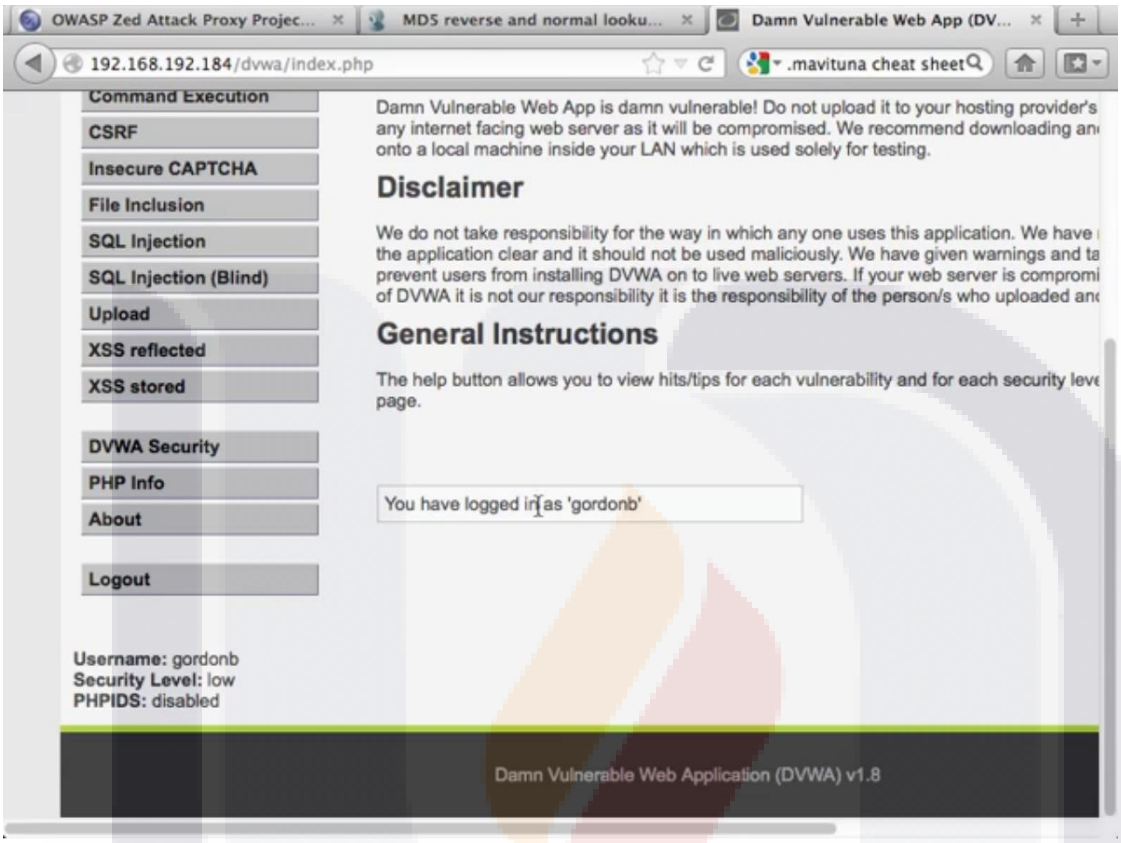


Imagen 35. Pantalla Principal de la Aplicación DVWA, sesión con el usuario gordonb.

También se pueden usar distintos comandos para inyectar información, no solo obtener usuarios y passwords, en la sección de Alertas de la herramienta ZAP, podemos observar que la última alerta de tipo SQL Injection detectada, en el apartado de Request, se puede introducir basura después de la IP 127.0.0.1, lo cual podemos usar a nuestro favor para introducir otro comando diferente, Imagen 40.

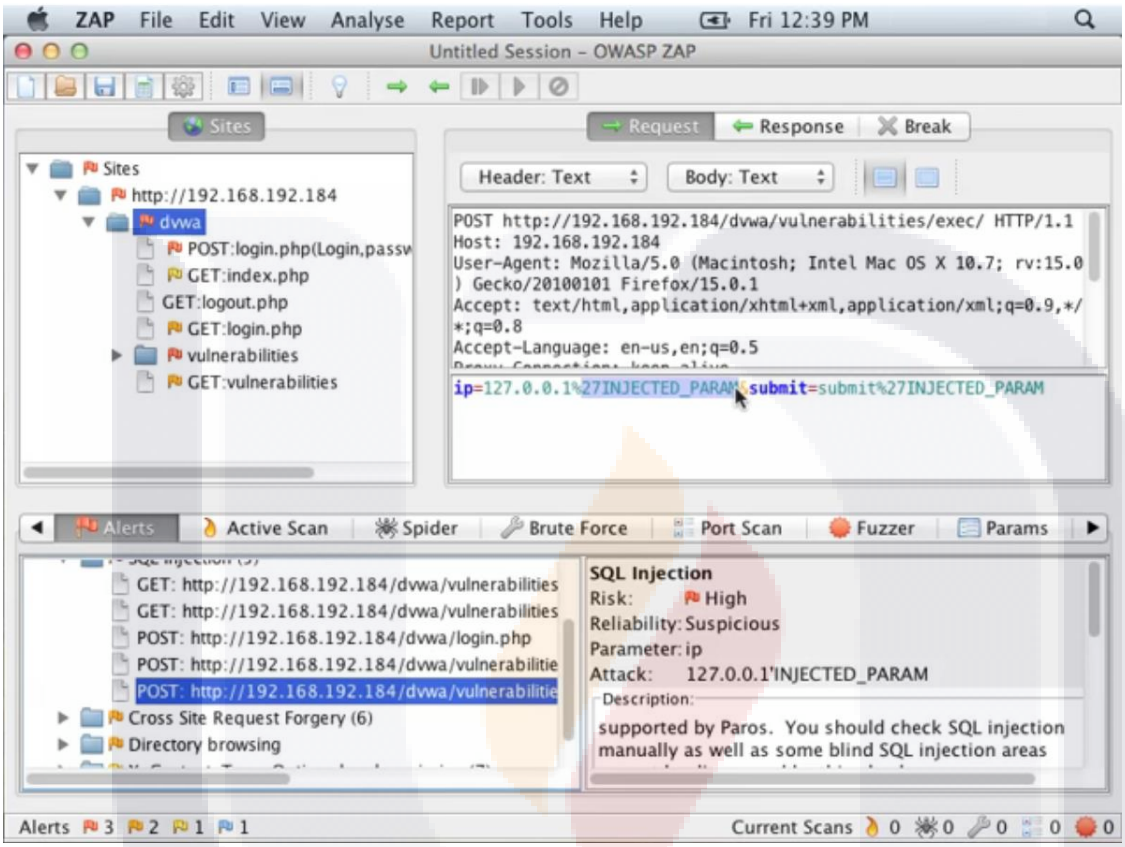


Imagen 36. Alerta SQL Injection con método POST para introducir otro comando como parámetro en el Request.

Modificando ese parámetro, trataremos de averiguar si podemos inyectar un comando &ls (Imagen 41) que nos traiga la lista de directorios.

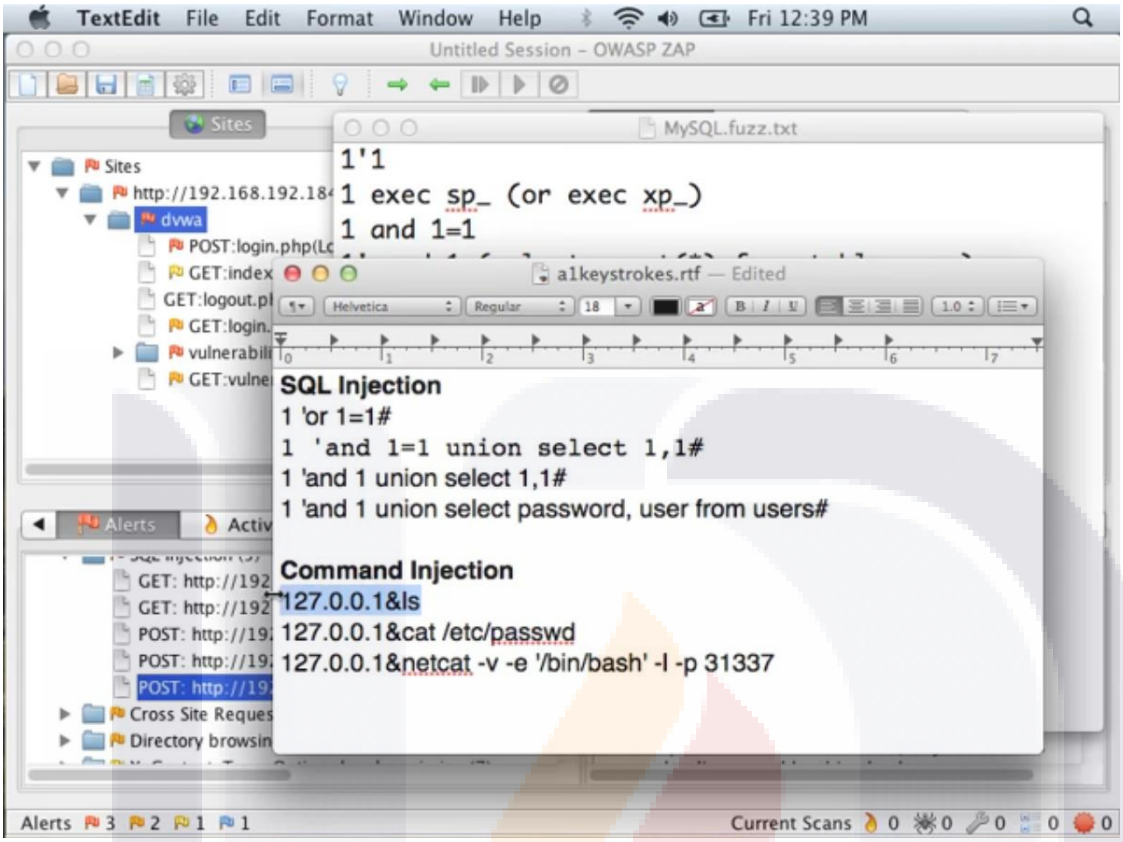


Imagen 37. Comando a utilizar en el Request para SQL Injection.

Vamos a la aplicación DVWA en el apartado Command Execution del lado izquierdo, e introducimos el comando seleccionado anteriormente (Imagen 42) y damos Enter o click en el botón Sumit para que la aplicación lo ejecute. El comando &LS utilizado en la aplicación nos dio la lista de archivos, podemos intentar algo más agresivo y tratar de obtener los passwords (Imagen 43).

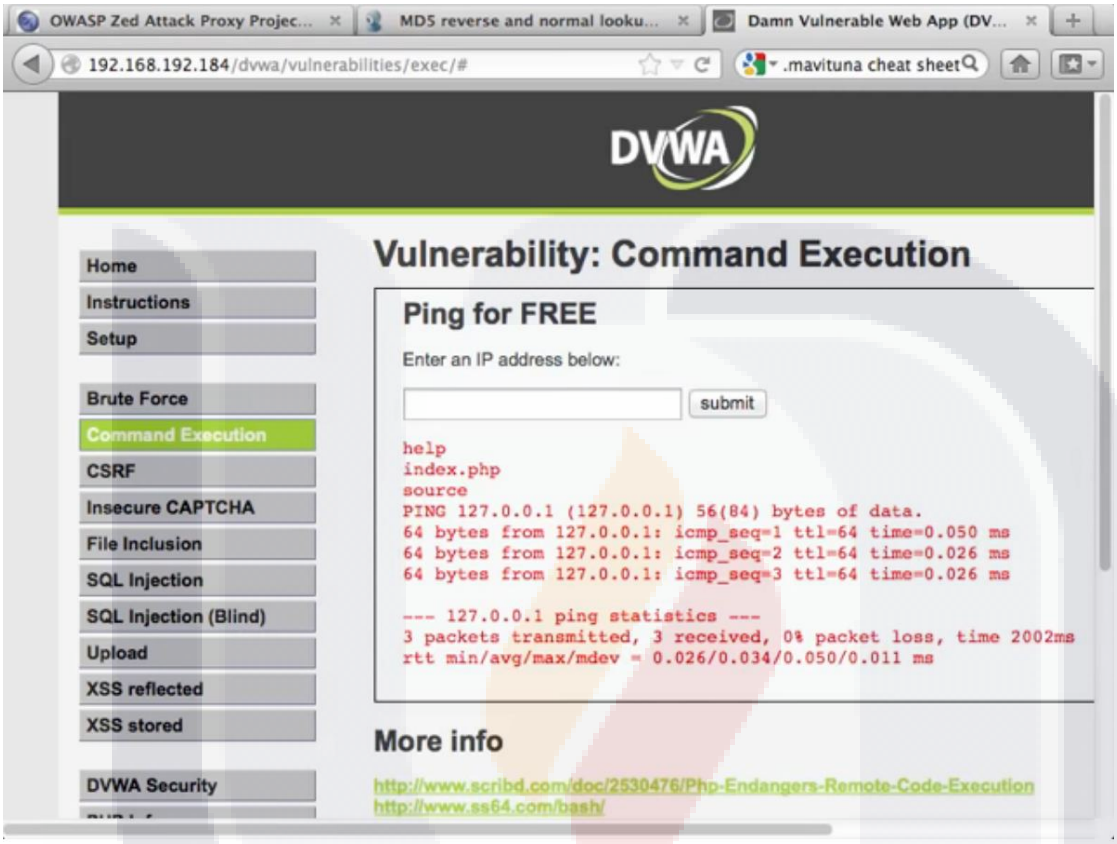


Imagen 38. Resultados del comando LS utilizado en la aplicación DVWA.

Ahora que nos damos cuenta que funciona, podemos hacer uso de comandos más agresivos para obtener las contraseñas, después de introducir un comando distinto que traiga las contraseñas, podemos observar los resultados que trae este comando que es mas agresivo en la siguiente imagen (Imagen 45) que obtuvo una lista de contraseñas.

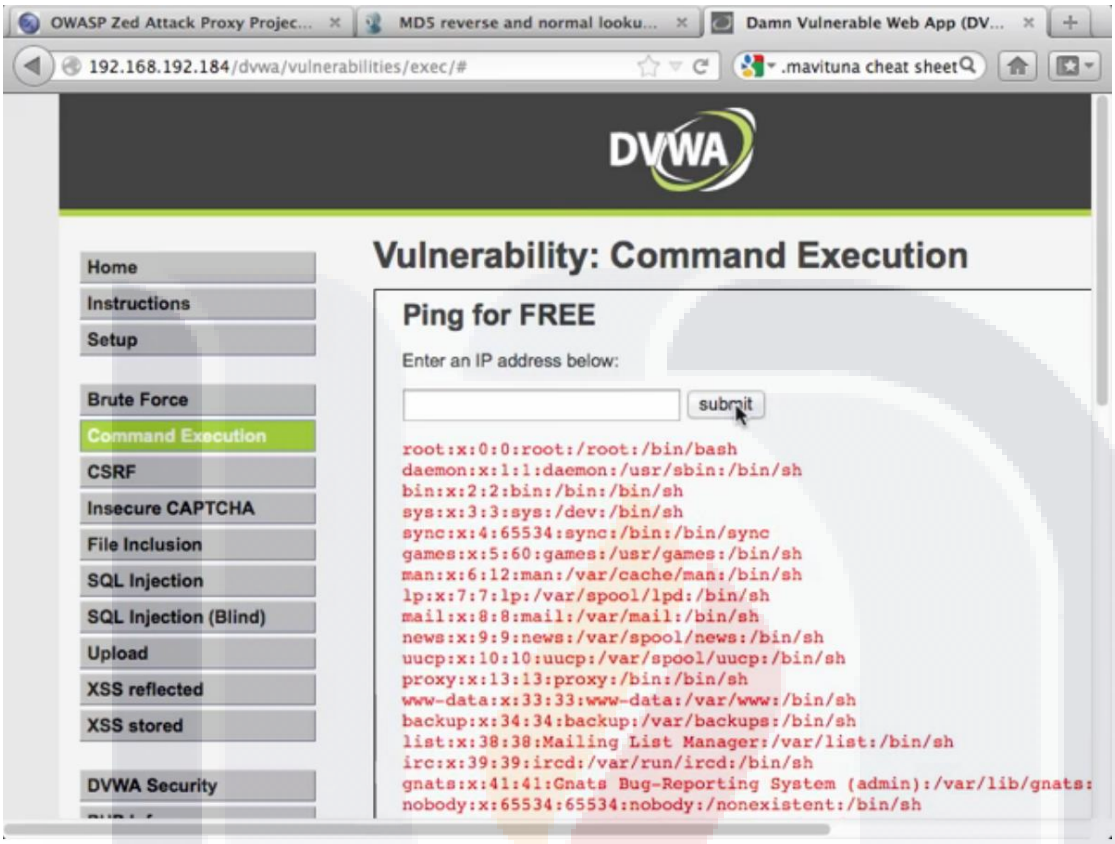


Imagen 39. Resultados del comando SQL Injection para la obtención de contraseñas en la aplicación DVWA.

6. Resultados y validación de la intervención

6.1. Resultados

Por la confidencialidad de la información proporcionada para este caso práctico por parte de INEGI a continuación solo se presentaran los datos a nivel general para los años anteriores al 2017.

Se analizaron doscientas aplicaciones con tecnología Web 2.0 de INEGI que fueron escritas o codificadas en las plataformas Oracle Java y Microsoft .Net como se puede observar en la siguiente imagen (Imagen 46).

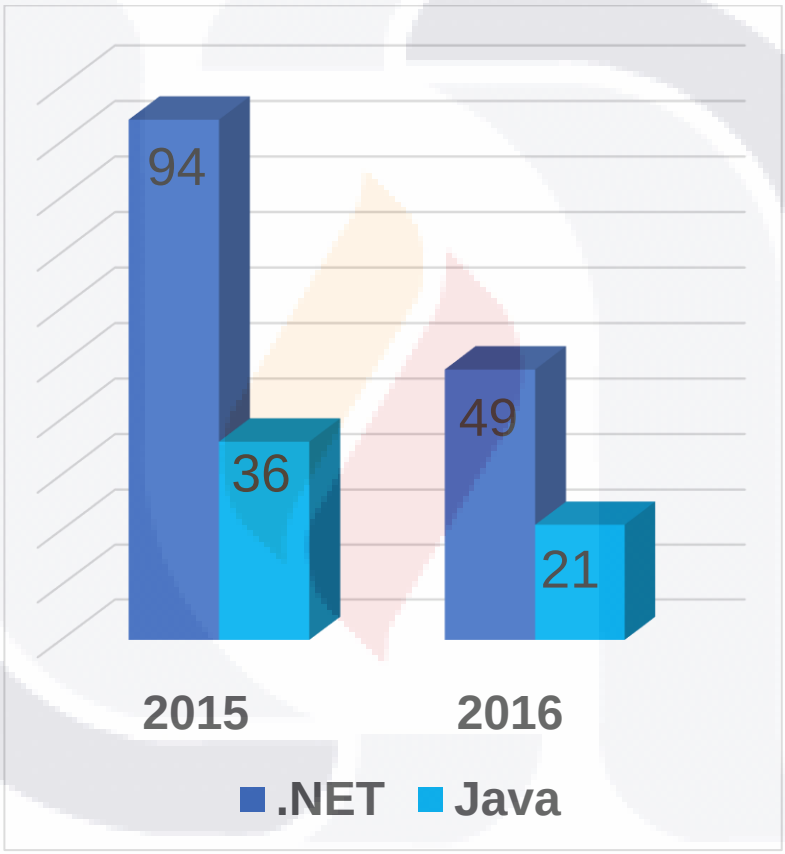


Imagen 40.Distribución de Proyectos de INEGI en 2015 y 2016 según su Tecnología.

En la siguiente imagen (Imagen 47) podemos observar la distribución de las vulnerabilidades encontradas en las aplicaciones, podemos darnos cuenta que la mayoría, el 65%, son del tipo Secuencia de Comandos en Sitios Cruzados o XSS, 30% para Inyección y solo el 5% de APIs no protegidas.

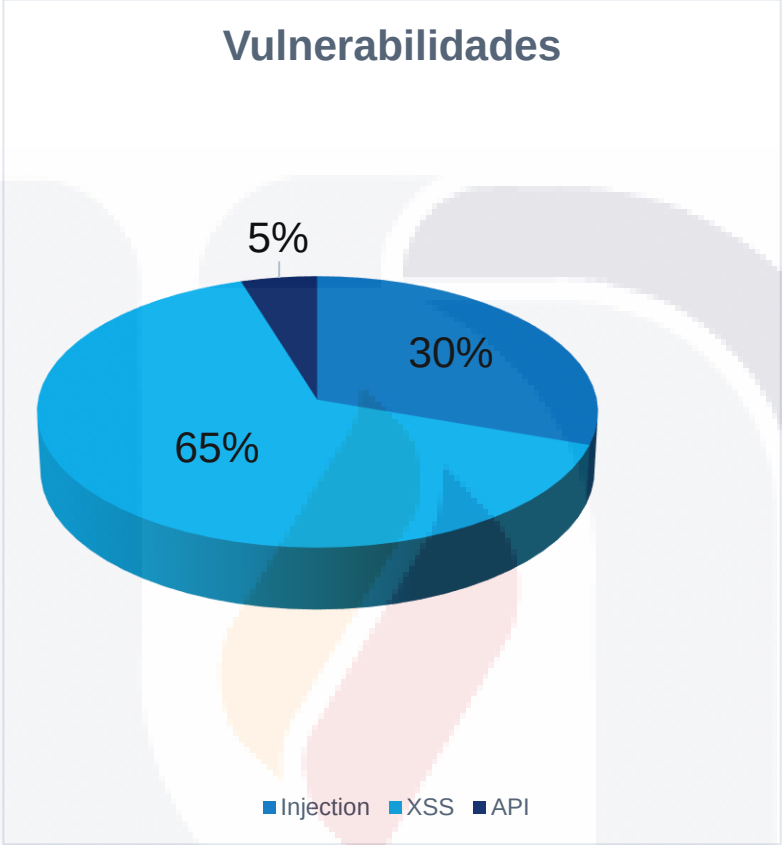


Imagen 41. Distribución de las Vulnerabilidades por Tipo en Porcentaje.

A partir de esas 200 aplicaciones se encontraron vulnerabilidades severas del tipo Inyección, Secuencia de Comandos en Sitios Cruzados y APIs no Seguras, en 29 proyectos, encontrando un total de 396 vulnerabilidades que se distribuyen de la siguiente manera: 120 del tipo Inyección, 257 del tipo Secuencia de Comandos en Sitios Cruzados y 19 para APIs no Seguras. Según se muestra en la Imagen 48.

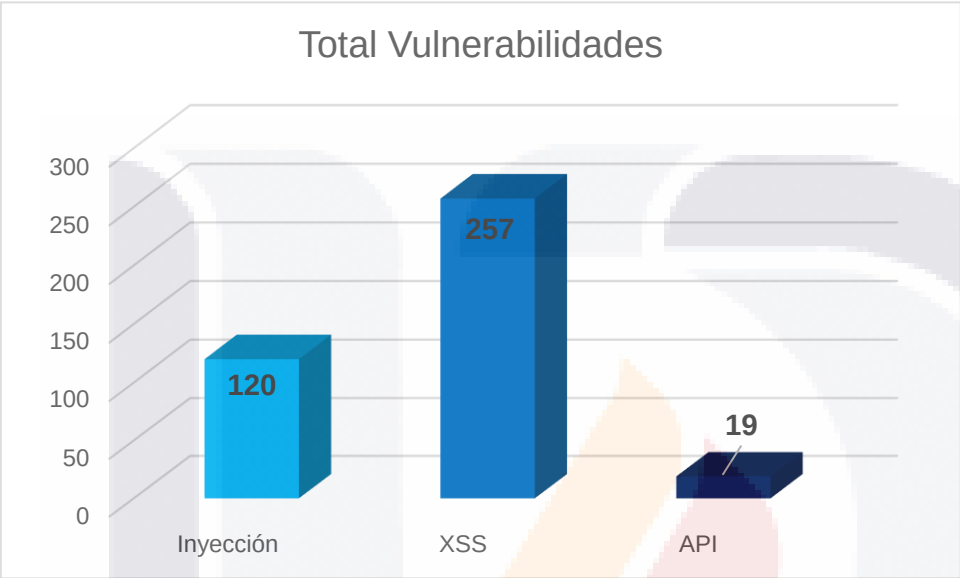


Imagen 42. Total Vulnerabilidades por Tipo en Cantidad.

De los 29 proyectos analizados, se tomaron los cinco con mayor número de incidencias, las cuales se distribuyen de la siguiente manera.

Para el proyecto 5 podemos observar en la siguiente imagen (Imagen 49) que el total de vulnerabilidades encontradas de tipo XSS + Inyección suman 46, de los cuales solo uno es de Inyección SQL, esto no significa que si la cantidad es mínima podamos dejarlo pasar por alto, ya que es de los más peligrosos, el resto son de XSS.

Imagen 43. Proyecto 5, Distribución de Vulnerabilidades por Tipo.

Para el proyecto 9 la cantidad se incrementa, presentando un total de 78 ataques, siendo 2 de Inyección SQL y el resto 76 para XSS mostrados en la Imagen 50.

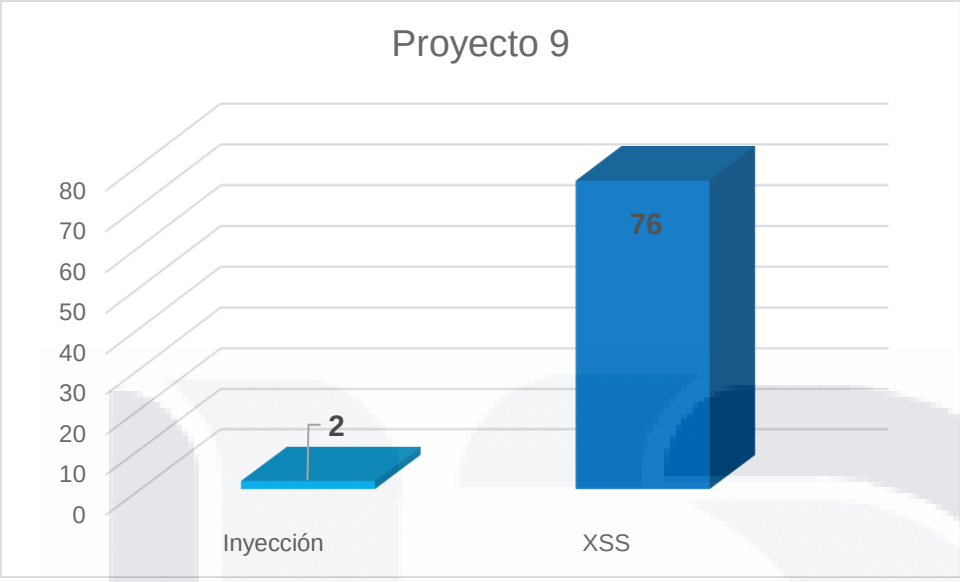


Imagen 44. . Proyecto 9, Distribución de Vulnerabilidades por Tipo.

En el proyecto 12 podemos ver que presenta los tres tipos de ataques, 7 de Inyección, 14 de XSS y 16 de APIs, dando un total de 37 ataques.

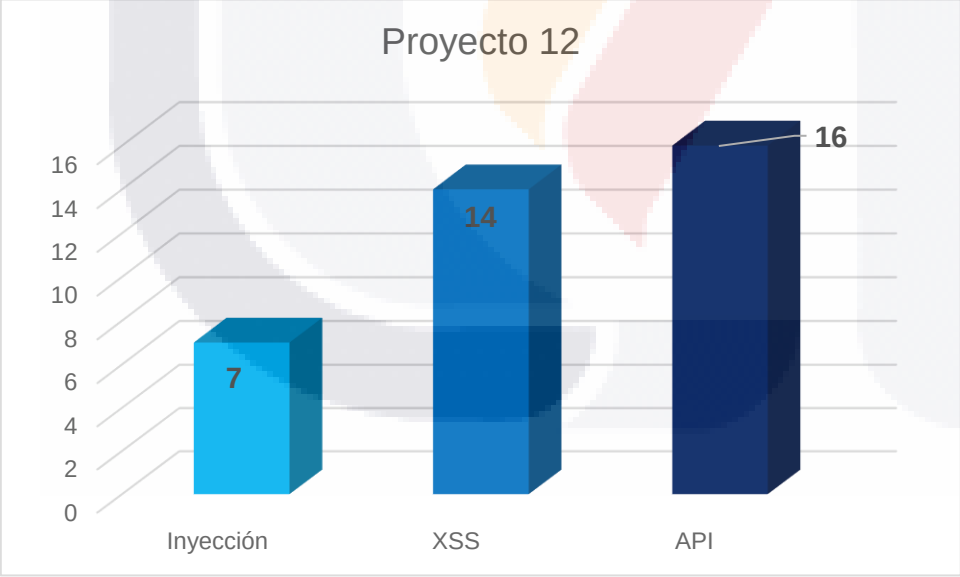


Imagen 45. . Proyecto 12, Distribución de Vulnerabilidades por Tipo.

Para el proyecto 16 obtuvimos 34 ataques, siendo la mayoría (33) de XSS y solamente uno para Inyección (Imagen 52).

Imagen 46. . Proyecto 16, Distribución de Vulnerabilidades por Tipo.

En el proyecto 27, consideramos que fue el peor proyecto al presentar 42 ataques de Inyección solamente como se puede observar en la Imagen 53.

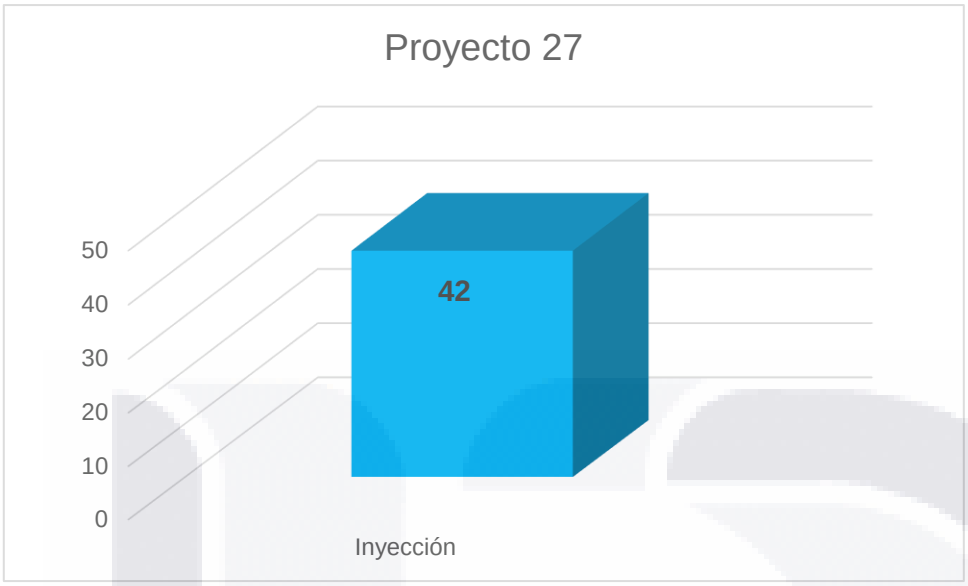


Imagen 47. . Proyecto 27, Distribución de Vulnerabilidades por Tipo.

Los tres tipos de vulnerabilidades en los cinco proyectos analizados se distribuyen de la siguiente manera, 168 pertenecen a Secuencia de Comandos en Sitios Cruzados (XSS) lo que representa el 70.89% del total, 53 vulnerabilidades del tipo Inyección (22.36%) y 16 vulnerabilidades encontradas para APIs no protegidas representa el 6.75% como se puede apreciar en la siguiente imagen (Imagen 54).

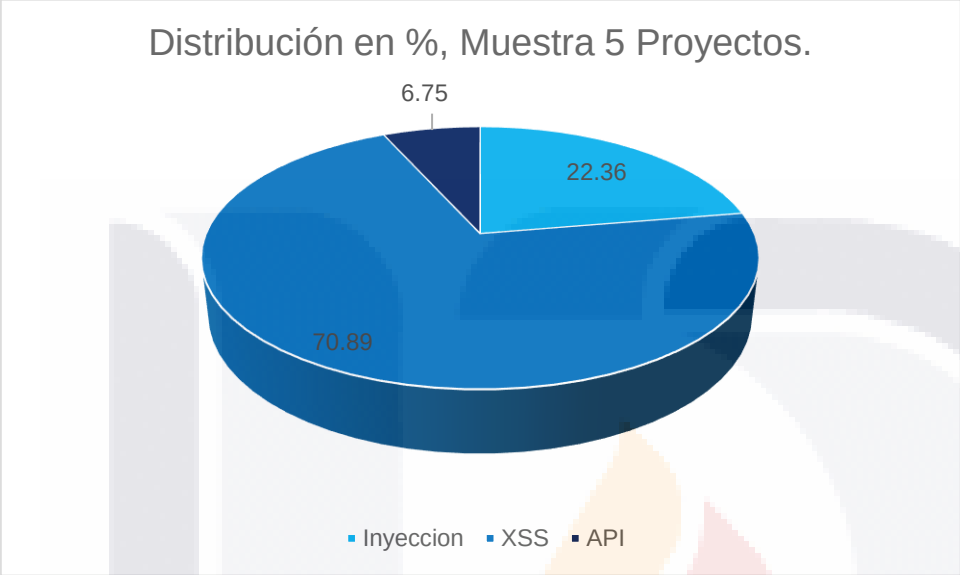


Imagen 48. Distribución en Porcentaje de la Muestra de 5 Proyectos.

6.2. Evaluación de la intervención

6.2.1. Valoración de los objetivos propuestos y alcanzados

Objetivo General: Evaluar las aplicaciones con tecnología Web 2.0 construidas en Oracle Java y Microsoft .NET por el Instituto Nacional de Estadística Geografía, con la finalidad de identificar vulnerabilidades y áreas de mejora en el sector de seguridad mediante el uso de una herramienta gratuita OWASP ZAP (Zed Attack Proxy).

Durante la duración de este trabajo practico se llevó a cabo el análisis en INEGI de sus aplicaciones Web 2.0 con la ayuda del proyecto Top 10, el marco de trabajo SAMM y la herramienta ZAP, todas de OWASP. Se pudieron encontrar

distintas vulnerabilidades y aprender diferentes temas desde el impacto que puede llegar a tener un ataque o vulnerabilidad detectada, las mejores prácticas que se deben seguir durante la vida del proyecto para evitar caer o provocar este tipo de vulnerabilidades en el código, arquitectura o diseño, así como simular con la herramienta ZAP un ataque directo y generar e interpretar los reportes de la misma.

Debido a lo mencionado anteriormente se puede decir que el objetivo general fue cubierto como se esperaba.

Objetivos Específicos:

- Analizar las aplicaciones con tecnología Web 2.0 construidas en Oracle Java y Microsoft .NET en búsqueda de las tres vulnerabilidades: A1- Inyección; A7- Secuencia de comandos en sitios cruzados (XSS por sus siglas en inglés) y A10- Interfaz de programación de aplicaciones (API por sus siglas en inglés), no protegidas.
- Presentar el reporte de OWASP ZAP sobre las aplicaciones evaluadas.
- Presentar el reporte final de la evaluación realizada a la organización.
- Presentar resultados y sugerencias a la compañía para mejorar la seguridad en aplicaciones Web 2.0.

En el caso de los objetivos específicos, de igual manera se cumplieron, desde el análisis de las tres vulnerabilidades seleccionadas para las aplicaciones Web 2.0 que usan tecnología Java y .Net en el instituto, una simulación de ataque en ZAP y los resultados y sugerencias para la organización, el reporte de la herramienta ZAP se presentó a la organización para su registro y análisis, al ser información sensible no puede ser presentado en este trabajo, los resultados fueron presentados a la organización y las sugerencias se mencionan en parte

en este trabajo en los apartados de las tres fases analizadas del marco de trabajo SAMM de OWASP.

Alcances y limitaciones de la intervención (problemas que surgieron)

En el apartado de alcance y limitaciones se pudo observar que la mayoría de la información utilizada, proyectos, URLs, nombres de variables etc., es información sensible del instituto y por lo tanto no puede ser divulgada, aun así de manera interna y confidencial se llevaron a cabo las tareas necesarias para el trabajo practico y de forma general se presentaron los resultados y reportes que la organización permite.

Otro de los problemas que se presentaron es que, hasta diciembre de 2017 se tenía contemplada una versión candidata del proyecto OWASP Top 10 que contemplaba un orden para las vulnerabilidades, un par de meses más adelante se hace la liberación oficial de la nueva versión del OWASP Top 10 y este orden cambió para las vulnerabilidades A3: Secuencia de Comandos en Sitios Cruzados que pasó a tomar la posición número 7 en la lista de 2018 y la vulnerabilidad candidata A10: APIs no Protegidas que tenía el último lugar en la versión de 2017 se desintegra en problemas similares y no aparece como tal en la versión 2018.

Aportes a la organización o beneficios

Como tal, todo el análisis llevado a cabo con ayuda del marco de trabajo SAMM de OWASP es un producto interno que se entregó al instituto para su aplicación en proyectos futuros, aparte de la guía de mejores prácticas indicada en el marco de trabajo, se espera que con esto se haya creado conciencia sobre la importancia de la seguridad en las aplicaciones web 2.0 y se tome como un caso de ejemplo para años posteriores.

Uno de los beneficios principales es demostrar y dejar en claro donde se encuentra la organización actualmente en materia de seguridad, no

simplemente destacar los problemas que pudieran tener hoy o en el futuro, sino obtener las áreas de mejora que se pudieran fortalecer para seguir generando productos de calidad que sean seguros y reconocer lo que se está haciendo bien dentro del instituto y con dedicación seguir por ese camino.

Recomendaciones para desarrollos futuros (aspectos que no se pudieron atender en el caso práctico o aspectos que convendría incluir o considerar).

Como una continuación a todo este tema de seguridad y considerando que se tuviera el tiempo necesario o bien que el instituto lo hiciera por sí mismo con ayuda de expertos en seguridad, auditores o recursos internos, sería un gran avance si se siguiera analizando completamente con el uso de las herramientas de OWASP la lista completa del Top 10 para una mejor protección en todos los sentidos, contemplar el marco de trabajo completo de SAMM e implementarlo desde el nacimiento de un proyecto hasta su finalización.

Se debe tomar en cuenta que OWASP no es el único organismo que habla de seguridad y de los ataques más fuertes, hay diferentes instituciones como SANS y las listas de vulnerabilidades no se limitan a solo 10, existe una gran variedad de ataques por lo cual la mejor estrategia es analizar que se adecua a nuestra organización para estar prevenidos y tener las mejores defensas.

7. Conclusiones

Con base al análisis realizado sobre seguridad en tecnologías web 2.0 dentro del instituto presentado en este documento, se concluye lo siguiente:

- Fue necesario participar en laboratorio de pruebas del instituto para analizar una población de aplicaciones Web 2.0 basados en los criterios previamente seleccionados.

- Experiencia en el proceso para el uso de la herramienta OWASP ZAP para que todos los proyectos pasen por esta etapa como medida de seguridad.

- Retroalimentación con base al análisis realizado que las aplicaciones Web 2.0 del instituto presentan vulnerabilidades y podría ser susceptible a ataques si no son corregidas.

- Observación en los proyectos analizados la mayoría de las vulnerabilidades encontradas fueron de tipo Inyección, seguidos por Secuencia de Comandos en Sitios Cruzados y por ultimo APIs no Protegidas con la menor cantidad de vulnerabilidades encontradas.

- Estrategia y revisión del equipo de seguridad que lleve la metodología propuesta en este trabajo usando el marco de trabajo y la herramienta proporcionadas por el organismo OWASP.

- Este documento permite proporciona una referencia para la continuación de trabajos futuros y para el Instituto.

- **Ad hoc:** del latín, para esta situación en específico.
- **API:** Por sus siglas en inglés Application Programming Interface, o Interfaz de Programación de Aplicaciones, es un conjunto de subrutinas, funciones y procedimientos o métodos en una biblioteca para su fácil acceso mientras se desarrolla una aplicación.
- **Amenaza:** es un posible daño que podría explotar una vulnerabilidad para violar la seguridad y por lo tanto causar daño.
- **Ataques:** son las técnicas que utilizan los atacantes para explotar las vulnerabilidades en una aplicación. Los ataques son frecuentemente confundidos con vulnerabilidades así que hay que asegurarnos que los ataques a los que nos referimos son acciones que un atacante tomaría en lugar de una debilidad en la aplicación.
- **Autenticación:** Establece la identidad del usuario.
- **Autorización:** El usuario debe recibir un servicio o realizar una acción para lo cual tenga permisos.
- **Compartimentación:** significa que hay un acceso limitado a la información a un número de personas asignadas a cierta tarea, de esta forma entre menos personas sepan los detalles de la asignación menor el riesgo de que sea comprometida, todo esto usando niveles de autorización.
- **Confidencialidad:** La información debe ser accesible solo para aquellos con acceso autorizado.
- **Crawler:** un web crawler o web spider es una herramienta/servicio que navega automáticamente una URL web para crear un índice de todos los sitios y contenido del sitio (Dvijesh Bhatt, Sharnil Pandya, & Daiwat Amit Vyas, 2015).
- **Cross-Site Scripting:** véase Secuencia de comandos en sitios cruzados.
- **Desperfecto:** La causa del error.

- **Defecto:** Una falla en un componente o sistema que puede provocar malfuncionamiento.
- **Disponibilidad:** Los servicios de información y comunicación deben estar listos siempre que se necesiten.
- **Endurecimiento:** es el proceso de asegurar un sistema al reducir la superficie de vulnerabilidad.
- **Eventualidad:** un posible evento o consecuencia.
- **Exploit:** véase Explotamiento.
- **Explotamiento:** en seguridad es tomar ventaja de la vulnerabilidad y usarla para sacar el mayor provecho.
- **Failure:** véase Falla.
- **Falla:** Desviación del componente o sistema del resultado o servicio esperado.
- **Fallo:** La inhabilidad de un sistema o componente para llevar a cabo sus funciones.
- **Fault: véase desperfecto.**
- **Flaw:** véase Defecto.
- **Fuzzer:** es una herramienta automatizada que utiliza la técnica de caja negra para encontrar defectos de implementación usando la inyección de datos malformados o semi-malformados de forma automática.
- **Hardening:** véase Endurecimiento.
- **Injection:** véase Inyección.
- **Integridad:** Medida que permite al receptor determinar si la información recibida es correcta.
- **Legacy Systems:** véase Sistemas Heredados.
- **No-Repudiación:** Previene la negación futura de una acción pasada (enviar, recibir).
- **Oscuridad:** en seguridad informática, la oscuridad u ocultación es un principio que intenta utilizar el secreto en las distintas fases (diseño, implementación, etc.) para garantizar la seguridad.

- **Parser:** herramienta que se usa para realizar parsing o como es también conocido análisis de sintaxis o análisis sintáctico es el proceso de analizar una cadena de símbolos.
- **Proxy:** es un sistema o aplicación que actúa como servidor intermediario para resolver las peticiones o requests del cliente, obteniendo recursos de otros servidores (Luotonen & Altis, 1994).
- **Security Testing:** Técnica para determinar si un Sistema de información protege los datos y mantiene la funcionalidad.
- **Sistemas Heredados:** son métodos, tecnologías, sistemas o aplicaciones antiguos o no actualizados que se siguen usando por su importancia o criticidad.
- **Sniffer:** también es conocido como analizador de paquetes, analizador de red o analizador de protocolos, puede ser un programa o aplicación o bien un dispositivo de hardware físico que puede interceptar y guardar información del tráfico que pasa por la red.
- **Vulnerabilidad:** es un hoyo o debilidad en la aplicación el cual puede ser un fallo en el diseño o un bug de implementación que permite a un atacante causar daño a los accionistas de la aplicación.
- **Web 1.0:** Principalmente trata lo que es el estado estático, es decir los datos que se encuentran en esta no pueden cambiar, se encuentran fijos, no varían, no se actualizan.
- **Web 2.0:** comprende aquellos sitios web que facilitan el compartir información, la interoperabilidad, el diseño centrado en el usuario y la colaboración en la World Wide Web.
- **Web 3.0:** también conocida como web semántica, es una expresión que se utiliza para describir la evolución del uso y la interacción de las personas en internet a través de diferentes formas entre las que se incluyen la transformación de la red en una base de datos, un movimiento social con el objetivo de crear contenidos accesibles por múltiples aplicaciones sin navegador (non-browser), el empuje de las tecnologías

- de inteligencia artificial, la web semántica, la Web Geoespacial o la Web 3D. La expresión es utilizada por los mercados para promocionar las mejoras respecto a la Web 2.0
- **Web 4.0:** es el próximo gran avance y se centrará en ofrecer un comportamiento más inteligente, más predictivo, de modo que podamos con sólo realizar una afirmación poner en marca un conjunto de acciones que tendrán como resultando aquello que pedimos o decimos.
- **XSS:** véase Secuencia de comandos en sitios cruzados.

Bibliografía

Betancur Cartagena, L. F. (2016). *Propuesta estratégica de prácticas seguras para el desarrollo de software con metodologías ágiles* (PhD Thesis). Universidad Nacional de Colombia-Sede Medellín.

Cedeño, E. Y. (2015). ANÁLISIS DE LAS HERRAMIENTAS PARA EL PROCESO DE AUDITORÍA DE SEGURIDAD INFORMÁTICA UTILIZANDO KALI LINUX, 113.

Chiem, T. P. (2014). *A study of penetration testing tools and approaches* (Thesis). Auckland University of Technology. Recuperado a partir de http://aut.researchgateway.ac.nz/handle/10292/7801

Dvijesh Bhatt, Sharnil Pandya, & Daiwat Amit Vyas. (2015). Focused Web Crawler. Recuperado a partir de http://www.krishisanskriti.org/vol_image/21Jul201512071432%20%20%20%20%20DAIWAT%20A%20%20VYAS%20%20%20%20%201-6.pdf

Díaz, Oswaldo, M. M. (2017). Fortaleciendo un enfoque DevOps con Seguridad y Gestión de Riesgos: una experiencia de su implementación en un Centro de Datos en una organización Mexicana. IEEE Xplore Digital Library.

INEGI, I. N. de E. y G. (1985a). Confidencialidad de la información estadística y geográfica. Recuperado el 4 de abril de 2017, a partir de http://www.beta.inegi.org.mx/inegi/contenido/confidencialidad.html

INEGI, I. N. de E. y G. (1985b). Difusión de la información. Recuperado el 4 de abril de 2017, a partir de http://www.beta.inegi.org.mx/inegi/contenido/difusion.html

López, A. (2008). Su portal: El Portal de ISO 27000 en español. *IS027000. es). URL: http://www. iso27000. es/sgsi. html.(Consulta: mayo 03, 2008)*.

Luotonen, A., & Altis, K. (1994). World-Wide Web proxies. *Computer Networks and ISDN Systems*, *27*(2), 147–154. https://doi.org/10.1016/0169-7552(94)90128-7

Lutz, M., Boucher, X., & Roustant, O. (2013). Methods and applications for IT capacity decisions: Bringing management frameworks into practice. *Journal of Decision Systems*, *22*(4), 332–355. https://doi.org/10.1080/12460125.2013.846600

Norse Attack Map. (s/f). Recuperado el 12 de abril de 2018, a partir de http://map.norsecorp.com/#/

OWASP. (2011, julio 26). OWASP SAMM Project - OWASP. Recuperado el 12 de abril de 2018, a partir de http://www.opensamm.org/downloads/SAMM-1.0-es_MX.pdf

OWASP. (2017a). OWASP Top 10 - 2017 Release Candidate1. Recuperado el 12 de abril de 2018, a partir de https://www.owasp.org/images/3/3c/OWASP_Top_10_-_2017_Release_Candidate1_English.pdf

OWASP. (2017b). Top 10-2017 Top 10 - OWASP. Recuperado el 12 de abril de 2018, a partir de https://www.owasp.org/index.php/Top_10-2017_Top_10

OWASP. (2001). OWASP. Recuperado el 12 de abril de 2018, a partir de https://www.owasp.org/index.php/Main_Page

Rodríguez Argueta, M. (2011). Análisis de la seguridad de sitios web.

2009-2017 | Statistic. Recuperado el 12 de abril de 2018, a partir de https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/

8. Anexos

INSTITUTO NACIONAL DE ESTADÍSTICA Y GEOGRAFÍA

ASUNTO: CARTA SATISFACCIÓN USUARIO

CONSEJO ACADÉMICO DEL PROGRAMA CONACYT-PNPC DE MAESTRÍA EN INFORMÁTICA Y TECNOLOGÍAS COMPUTACIONALES (MITC)
UNIVERSIDAD AUTÓNOMA DE AGUASCALIENTES

Estimados Integrantes del Consejo Académico del Posgrado CONACYT-PNP MITC:

Por medio de este conducto hago constar los siguientes hechos:

Que el MC(c) Alan Tonatiuh González Peña [ID91852], quien cursó sus estudios de Posgrado en el programa CONACYT-PNPC MITC, desarrolló el trabajo práctico titulado "SECURITY TESTING TECHNIQUES AND TOOLS: A STUDY FOR WEB BASED APPLICATIONS", como requisito para su proceso de titulación en nuestra organización durante el período "Generación 2016-2017", y en mi carácter de CO-TUTOR de dicho trabajo práctico manifiesto:

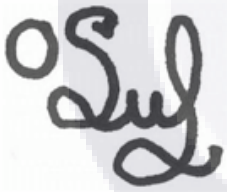
- **( ) Muy Alto grado de Satisfacción – CUMPLIMIENTO MAYOR A ESPERADO**
- **(X) Alto grado de Satisfacción – CUMPLIMIENTO ESPERADO**
- **( ) Moderado grado de Satisfacción – CUMPLIMIENTO MÍNIMO ESPERADO**
- **( ) Bajo grado de Satisfacción - INCUMPLIMIENTO**
- **( ) Muy Bajo grado de Satisfacción – INCUMPLIMIENTO**

en relación a la entrega final de dicho trabajo práctico.

Así mismo manifiesto de manera libre que considero que este tipo de Proyectos de Titulación servirá en nuestra organización para que el Grupo de Ingeniería de Sistemas adscrito al Instituto Nacional de Estadística y Geografía, emite el voto aprobatorio y utilizara los conocimientos adquiridos por esta investigación, en sus procesos y procedimientos "GIS-QA" [Pruebas de seguridad bajo la norma OWASP 2017] en el Centro de Datos.

Agradezco la atención a la presente carta.

Aguascalientes, Ags. Mayo, 2018



MSc. Edgar Oswaldo Díaz
Grupo de Ingeniería en Sistemas
Dirección de Cómputo y Comunicaciones

Avenida Héroe de Nacozari Sur 2301
Edificio de Informática, Nivel 1, Fraccionamiento Jardines del Parque
20276, Aguascalientes, Aguascalientes, Aguascalientes
Entre calle INEGI, Avenida del Lago y Avenida Paseo de las Garzas
910 53 00 ext. 4953
oswaldo.diaz@inegi.org.mx