UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Auditoría Interna a la CSI/UNAM-CERT conforme al estándar ISO 27001:2013

INFORME DE ACTIVIDADES PROFESIONALES

Que para obtener el título de

Ingeniero en Computación

P R E S E N T A

Jorge Alberto Hernández Cuecuecha

ASESOR DE INFORME

M.C. Paulo Santiago de Jesús Contreras Flores

Ciudad Universitaria, Cd. Mx., 2019

Contenido

INTRODUCCIÓN

Este documento se basa en las actividades que he realizado en la Coordinación de Seguridad de la Información/UNAM-CERT con el propósito de obtener el título de Ingeniero en Computación a través de la modalidad de titulación por trabajo profesional.

Presento las actividades que realicé en la Coordinación de Seguridad de la Información, así como dos proyectos, que considero que han sido los más relevantes para presentarlos en este escrito.

En la actualidad, cualquier organización de TI, diariamente se enfrenta a un enorme número de riesgos, ocasionando que la información se pueda ver comprometida. El implementar un Sistema de Gestión de la Información, basado en el estándar ISO/IEC 27001, es una de las soluciones más apropiadas para poder evaluar riesgos, establecer estrategias y controles para preservar la confidencialidad, la integridad y la disponibilidad de la información.

La Coordinación de Seguridad de la Información/UNAM-CERT adquirió la certificación en ISO 27001 en el año 2010, logrando analizar y gestionar los riesgos asociados al Proceso de Respuesta a Incidentes. Debido a esta certificación, la Coordinación debe cumplir con ciertas cláusulas que el ISO establece, una de ellas es la de llevar a cabo Auditorías Internas.

El primer proyecto que presento en este documento es la auditoría interna que realicé a la CSI/UNAM-CERT, donde contribuyo a notificar hallazgos, con el propósito de mantener la mejora continua del SGSI.

El segundo proyecto que abordo tiene que ver con la realización de un analizador estático de vulnerabilidades de código PHP, desarrollado con las mejores prácticas de la metodología ágil Scrum. El propósito del proyecto es detectar hallazgos que puedan ser aprovechados por usuarios mal intencionados y como resultado final generar reportes para la comprensión de estos.

Finalmente, este documento termina con las conclusiones para cada uno de los proyectos que presento, donde expreso lo que me aportó la Facultad de Ingeniería para llevar a cabo de manera satisfactoria los proyectos realizados.

El lector podrá consultar los tres Anexos en donde detallo las actividades que realicé durante en el desarrollo del analizador mediante la metodología ágil Scrum, un reporte completo en formato CSV y un diccionario de la base de datos que ocupé para el analizador. Además, el código fuente del analizador de vulnerabilidades está disponible en el sitio github, *https://github.com/ginohdz/Mr.-PHP-Analyzer*.

CAPÍTULO 1. COORDINACIÓN DE SEGURIDAD DE LA INFORMACIÓN/UNAM-CERT

1.1 MISIÓN

"Contribuir al desarrollo de la UNAM, a través de la prestación de servicios especializados, la formación de capital humano y el fomento de la cultura de seguridad de la información." (CSI/UNAM-CERT, Misión y Visión, 2018)

1.2 VISIÓN

"Consolidar a la UNAM como la entidad líder en materia de seguridad de la información en el país." (CSI/UNAM-CERT, Misión y Visión, 2018)

1.3 OBJETIVOS

- Proporcionar servicios de seguridad de la información para la UNAM y otras organizaciones.
- Promover la cultura de seguridad de la información.
- Formar especialistas que desarrollen y apliquen estrategias de protección de la información.
- Difundir contenidos especializados en seguridad de la información.
- Colaborar con instituciones nacionales e internacionales en materia de detección y respuesta a incidentes.
- Elaborar políticas y lineamientos de seguridad de la información para las dependencias y entidades académicas universitarias.

1.4 SERVICIOS BRINDADOS

Como lo mencionaba en el punto anterior, uno de los objetivos de la CSI/UNAM-CERT es proporcionar servicios de seguridad de la información ya sea para la UNAM o para otras organizaciones con el propósito de aumentar la seguridad de la infraestructura tecnológica de éstas. Los servicios proporcionados son los siguientes:

- Implementación de SGSI (Sistema de Gestión de Seguridad de la Información) de acuerdo con el estándar ISO 27001:2013.
- Auditoria informática.
- Análisis forense.
- Análisis de vulnerabilidades y pruebas de penetración.
- Análisis de tráfico de red.
- Análisis de riesgos.
- Respuesta a incidentes de seguridad de la información.
- Revisión de configuraciones.
- Creación de políticas de seguridad de la información.

- Revisiones de seguridad para aplicaciones web.

1.5 Capacitación

La CSI/UNAM-CERT brinda servicios de capacitación, tanto a personal externo como personal interno a ésta, los cuales describo a continuación.

1.5.1 Plan de Becarios en Seguridad Informática

La CSI/UNAM-CERT desde 2003 ha impartido cada año un plan de becarios en donde se busca capacitar a los asistentes en temas relacionados con seguridad informática, desde cuestiones técnicas, como lo es programación, hasta temas no tan técnicos como legislación relacionada.

El plan de becarios está abierto a cualquier estudiante de la UNAM u otras universidades, en donde inicia con cursos básicos para entender las bases de la seguridad de la información, para que así todos los estudiantes sin importar sus antecedentes, al terminar estos cursos básicos estén en un mismo nivel y logren entender los siguientes cursos que son más especializados en la materia.

Se sabe que todo lo relacionado con tecnología va cambiando, como consecuencia de esto, la CSI/UNAM-CERT va actualizando el contenido de sus cursos año con año. En el plan de becas 2016-2017, 11ª generación, en el cual yo fui becario, se impartieron los siguientes cursos:

Curso 1: Seguridad en redes y sistemas operativos.

- Introducción a la seguridad informática
- Introducción al sistema operativo Linux/Unix
- Utilerías y programación Shell
- Administración y seguridad en redes
- Administración y seguridad en Windows I
- Administración y seguridad en Linux/Unix
- Administración y seguridad en Windows II
- Scripts para administración en Windows

Curso 2: Programación orientada a la seguridad.

- Programación con Python
- Seguridad en bases de datos
- Lenguaje C y llamadas al sistema
- Programación orientada a objetos (C#)
- Seguridad en aplicaciones web

Curso 3: Análisis de vulnerabilidades y hacking ético.

- Pruebas de penetración
- Análisis de vulnerabilidades
- Criptografía y sus aplicaciones

Curso 4: Monitoreo de seguridad en redes y respuesta a incidentes.

- Análisis de software malicioso
- Detección de intrusos y tecnologías Honeypot
- Seguridad perimetral
- Respuesta a incidentes
- Análisis forense
- Legislación
- Seminario: Threat intelligence
- Seminario: Seguridad de la información: perspectivas y tendencias

Curso 5: Gestión de la seguridad de la información.

- ISO 27000 y auditorías
- Administración de proyectos
- Seminario: Definición de estrategias para el manejo de grupos de seguridad
- Seminario: Seguridad en SCADA
- Seminario: Fraude y riesgo tecnológico
- Seminario: Big data

1.5.2 Congreso Seguridad en Cómputo

La Universidad Nacional Autónoma de México a través de la CSI/UNAM-CERT organiza el Congreso de Seguridad en Cómputo desde 1998 con el propósito de especializar a los asistentes en diversos temas de seguridad de la información, con una serie de cursos especializados y un ciclo de conferencias sobre seguridad de la información que reúne expertos nacionales e internacionales.

1.5.3 AdminUNAM

Es un evento que se desarrolla con el fin de fortalecer la colaboración, discusión e instauración de soluciones en el ámbito de Tecnologías de la Información y Comunicación entre las dependencias y entidades de la UNAM, la Dirección General de Cómputo y de Tecnologías de Información y Comunicación (DGTIC), organiza un ciclo de conferencias en donde invita a administradores de sistemas, responsables de cómputo y responsables de seguridad de la información de las diferentes entidades de la UNAM y de empresas privadas.

La CSI/UNAM-CERT participa en este evento realizando el Programa de Actualización en TIC, en donde se imparten cursos, por ejemplo:

- Implementación de servicios en redes Windows.
- Desarrollo seguro de aplicaciones web.
- Monitoreo de seguridad de red.
- Pruebas de penetración y hacking ético.

1.6 Estructura Organizacional de la CSI/UNAM-CERT

1.6.1 Dirección General de Cómputo y de Tecnologías de Información y Comunicación

La Dirección General de Cómputo y de Tecnologías de Información y Comunicación (DGTIC) es la dependencia encargada de "impulsar la transformación sustancial de la UNAM y su impacto en la sociedad, mediante el uso de tecnologías de información y comunicación en apoyo de las actividades sustantivas de nuestra máxima casa de estudios: docencia, investigación y difusión de la cultura". (UNAM, 2018)

Dentro de los servicios que la DGTIC ofrece son los siguientes:

- Seguridad de la Información
- Acervos digitales,
- Biblioteca,
- Firma electrónica,
- Observatorio IXTLI y
- Supercómputo

1.6.2 Organigrama DGTIC

La DGTIC está constituida por diez áreas, todas a cargo del Dr. Felipe Bracho Carpizo. En la *imagen 1.1*, destaco la Dirección de Sistemas y Servicios Institucionales (DSSI) ya que es donde se encuentra adscrita la CSI/UNAM.CERT, la DSSI está a cargo del Act. José Fabián Romo Zamudio. A continuación, muestro el organigrama (DGTIC, 2015) de cómo está conformada dicha Dirección:

- **Coordinación de Seguridad de la Información (CSI/UNAM-CERT)**
- Supercómputo
- Departamento de Visualización y Realidad Virtual
- Departamento de Firma Electrónica Avanzada
- Departamento de Administración de Servidores
- Servicios del Centro de Datos

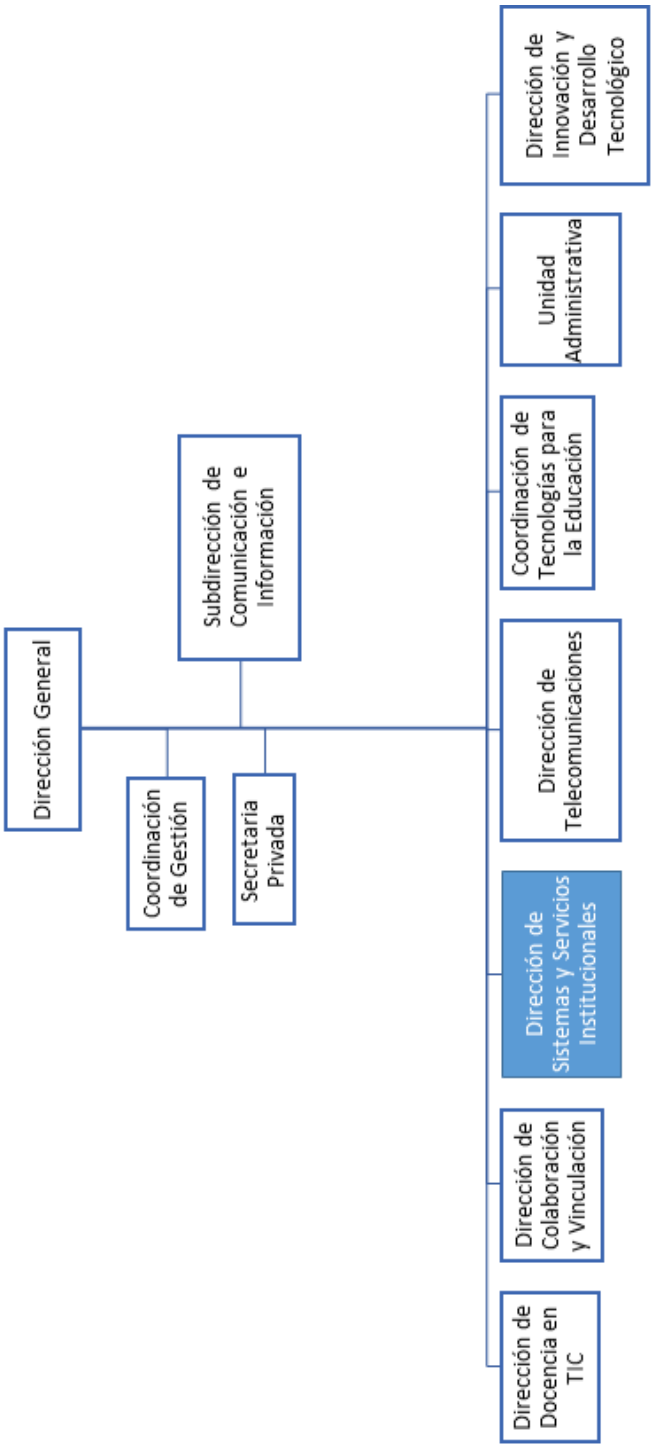


Imagen 1.1. Organigrama DGTIC.

1.6.3 ORGANIGRAMA CSI/UNAM-CERT

Como se observa en la *imagen 1.2*, la Coordinación de Seguridad de la Información (CSI) está conformada por cinco áreas, el responsable de la CSI/UNAM-CERT es el M.C. José Roberto Sánchez Soledad, Coordinador de Seguridad de la Información. En el organigrama destaco mi puesto, el cual es el Auditor del SGSI (Sistema de Gestión de Seguridad de la Información), en el siguiente capítulo explico a fondo mi rol en la CSI/UNAM-CERT.

Coordinación de Seguridad de la Información/UNAM-CERT

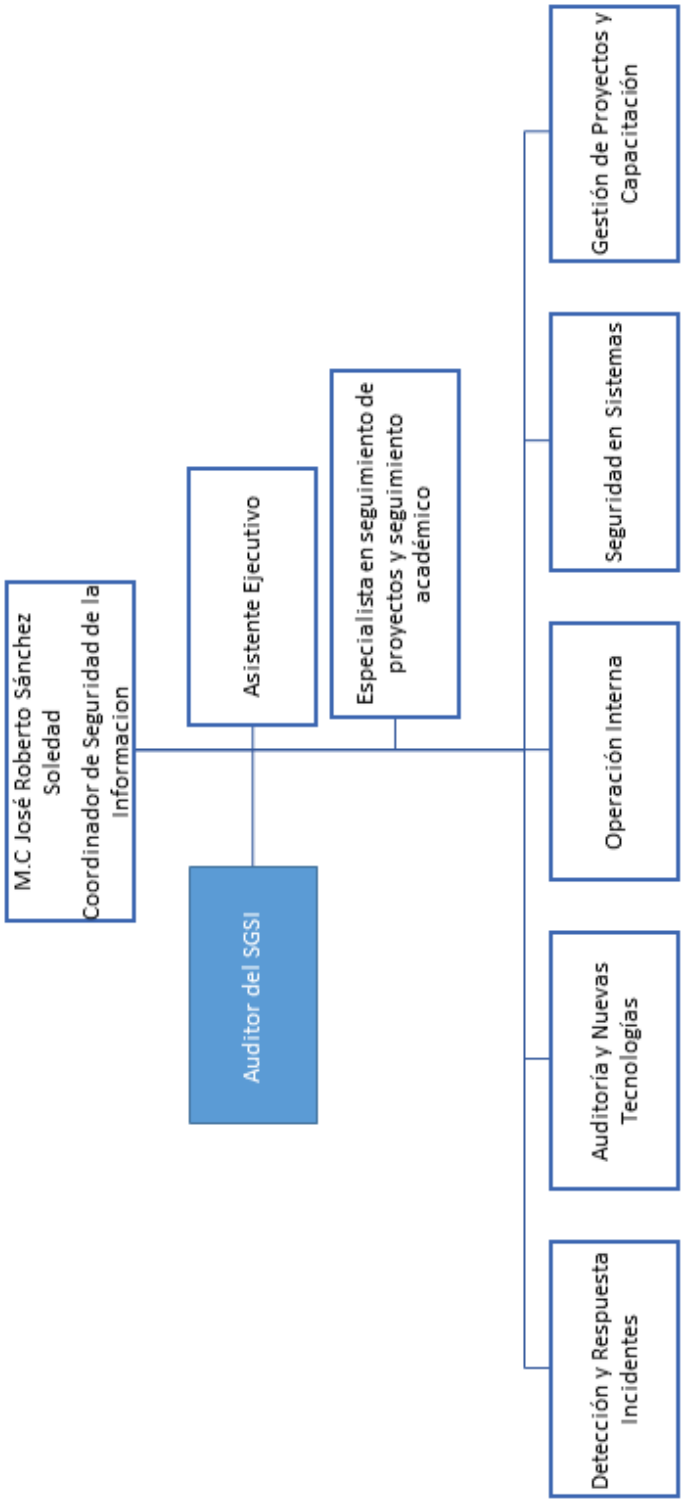


Imagen 1.2. Estructura organizacional CSI/UNAM-CERT.

CAPÍTULO 2. PUESTO DE TRABAJO

2.1 INGRESO AL PUESTO DE AUDITOR DEL SGSI

Para empezar con la descripción, quisiera explicar la forma en que ingresé a laborar a la Coordinación de Seguridad de la Información.

Cursando el penúltimo semestre de la carrera de Ingeniería en Computación en la Facultad de Ingeniería (durante el semestre 2016-2), estando en el módulo de Redes y Seguridad, apliqué la solicitud para entrar al Plan de Becarios de Seguridad Informática impartido por la Coordinación de Seguridad de la Información. Fueron varios filtros para seleccionar a los becarios que formarían parte de la 11ª generación, uno de ellos consistió en un examen general de conocimientos (matemáticas, programación, razonamiento matemático, etc.), el cual no tuvo mucha dificultad ya que tengo cierto dominio sobre dichos temas, los cuales adquirí a lo largo de los semestres cursados en la carrera. Después de haber pasado los filtros fui seleccionado entre más de 100 aspirantes para formar parte de este Plan de becarios.

Estando en el plan de becas, fui superando uno a uno cada curso, llegando el quinto curso (Gestión de la seguridad de la información), se impartió el primer módulo, ISO 27001 y auditorias, al finalizarlo, debido a la rotación del personal de la CSI/UNAM-CERT, se presentó la oportunidad de ingresar a la CSI/UNAM-CERT con el puesto de Auditor del SGSI.

Como parte del proceso de selección, presenté el examen de conocimientos, el cual aprobé satisfactoriamente. Una vez aprobado el examen, se me comunicó que obtuve el puesto, el cual he desempeñado desde octubre de 2017.

2.2 AUDITOR DEL SGSI

Para empezar con la descripción del puesto, quisiera explicar por qué no pertenezco a ninguna de las áreas que se encuentran en la CSI/UNAM-CERT, *Imagen 1.2*, es por una sencilla razón, un auditor debe ser independiente de los gerentes operativos de las funciones a ser auditadas con el objetivo de evitar conflicto de intereses y poder mantener una objetividad a lo largo del proceso de auditoría.

2.2.1 ACTIVIDADES

Las actividades que realizo en la Coordinación de Seguridad de la Información están orientadas principalmente a la verificación del cumplimiento, por parte de su personal, de las políticas implementadas en materia de seguridad de la información, además de revisar y reportar el cabal cumplimiento del SGSI de acuerdo con la norma ISO/IEC 27001:2013.

Las actividades que realizo dentro de la Coordinación son las siguientes:

- Realización de auditorías de acuerdo con el estándar ISO/IEC 27001:2013 de Sistemas de Gestión de Seguridad de la Información para mejorar la seguridad de la información de las organizaciones a nivel institucional.
- Evaluar controles de seguridad existentes.
- Identificar las nuevas amenazas que atenten contra la seguridad de la información de los activos y realizar las observaciones correspondientes.
- Evaluar mejores prácticas de seguridad de la información y su relación con el ISO/IEC 27001:2013.
- Participar en proyectos en los que se requiera personal que cuente con conocimientos a un nivel técnico elevado en el área de seguridad de la información, puede no estar relacionado directamente con el puesto de Auditor del SGSI. Dichos proyectos podrán ser internos o externos a la CSI/UNAM-CERT.

2.2.2 Revisión de Cuentas y Revisión de Conocimiento sobre el SGSI por parte del Personal de la CSI/UNAM-CERT

En la Seguridad de la Información es de suma importancia el control de cuentas de usuario, a continuación, explicaré en que consiste el control de acceso basado en roles, ya que es como se maneja en la CSI/UNAM-CERT, de tal forma que el rol que desempeña el integrante tenga los privilegios mínimos para poder llevar a cabo sus actividades.

2.2.2.1 Control de acceso basado en roles

Es un método para regular el acceso a los recursos de alguna computadora o red, basado en las funciones de los usuarios dentro de la empresa. El acceso es la capacidad de un usuario para realizar una tarea específica, como ver, crear o modificar un archivo. Los roles se definen según la competencia laboral, autoridad y responsabilidad dentro de la empresa (TechTarget, 2012).

A modo de ejemplo de cuentas de usuario a continuación se enlistan algunos tipos y sus permisos, aplicado a cualquier organización:

Administración: el contacto principal para una cuenta o rol específico. Tiene acceso a archivos confidenciales.

Técnico: asignado a los usuarios que realizan tareas técnicas. Tiene acceso a archivos internos a la empresa y que tengan que ver con su área.

Usuario final: acceso para usuarios que realizan tareas administrativas. Tiene acceso a archivos internos a la empresa y que tengan que ver con su área.

Invitado: usuarios externos a la empresa cuyo acceso sólo es a archivos públicos.

Al agregar un usuario a un grupo de roles, el usuario tiene acceso a todos los roles en ese grupo. Si se eliminan, el acceso queda restringido. Los usuarios también pueden ser asignados a múltiples grupos en caso de que necesiten acceso temporal a ciertos datos o programas y luego se eliminen una vez que el proyecto esté completo.

Con base en el punto anterior, la CSI/UNAM-CERT maneja distintos roles asignados a las cuentas de los usuarios. Dichas cuentas consisten en poder acceder a un determinado servicio, siempre y cuando esté autorizado.

Dentro de la CSI/UNAM-CERT se cuenta con una política que trata específicamente sobre el tipo de cuenta que está asociada a cada usuario, como se describía anteriormente, de tal forma que éstos estén enterados de qué actividades pueden llevar a cabo, para evitar que haya pérdida, daño, divulgación no autorizada, o eliminación de información que sea propiedad de la CSI/UNAM-CERT, en la política de manera general se describen los siguientes aspectos:

- tipos de cuentas: se define los distintos tipos de cuentas que existe dentro de la CSI/UNAM-CERT;
- las cuentas deberán ser únicas y personales: no pueden existir cuentas repetidas y estas no se pueden prestar;
- baja de cuentas: de acuerdo con la política de condiciones del empleo, en el apartado de terminación del empleo, se debe notificar la baja de personal al área correspondiente para proceder a inhabilitar la cuenta;
- cambio de contraseñas en un determinado tiempo: con el motivo de ir fortaleciendo las contraseñas anteriores y sea más complejo que alguien pueda conseguir contraseñas;
- bloqueo de cuentas después de varios intentos fallidos: con el motivo de evitar que usuarios mal intencionados tengan la oportunidad de intentar ingresar a algún servicio de manera continua hasta tener las credenciales correctas.

Lo antes descrito lo tengo que revisar en las áreas de Operación Interna, Seguridad en Sistemas, Detección y Respuesta a Incidentes y Gestión de Proyectos y Capacitación. Donde primeramente selecciono una muestra del personal para la revisión, para luego realizar dicha revisión. Tengo que documentar los hallazgos encontrados en un informe, donde detallo lo que reviso. Al termino de dicho reporte, tengo que presentar los

resultados ante los responsables de cada área y al Coordinador de la CSI/UNAM-CERT, para que si se llegara a presentar un incidente[1] se pudiera resolver a la brevedad.

Al realizar dicha revisión, de acuerdo con el Anexo A del ISO 27001:2013 valido la correcta aplicación de los siguientes controles asociados a la revisión:

- 9.2.1 Registro y baja de usuario: procedimiento formal de registro y retirada de usuarios que haga posible la asignación de los derechos de acceso.
- 9.2.2 Provisión de acceso de usuario: procedimiento formal para asignar o revocar los derechos de acceso para todos los tipos de usuarios de todos los sistemas y servicios.
- 9.2.5 Revisión de los derechos de acceso de usuario: los propietarios de los activos deberán revisar los derechos de acceso de usuarios a intervalos regulares.
- 9.3.1 Uso de la información secreta de autenticación: que usuarios sigan las prácticas de la organización en el uso de la información secreta de autenticación.
- 9.4.3 Sistema de gestión de contraseñas: los sistemas para la gestión de contraseñas deberán ser interactivos y establecerán contraseñas seguras y robustas. (Normalización, 2013)

Por otro lado, en cuanto a la revisión de conocimiento sobre el SGSI por parte del personal de la CSI/UNAM-CERT, me encargo de documentar el nivel de conocimiento que tienen los integrantes ya que de esta forma como Auditor puedo validar que la CSI/UNAM-CERT está madurando, es decir, que los integrantes sepan que están dentro de un SGSI, que entiendan lo que es un SGSI, su rol dentro de éste, porque de esta forma el SGSI de la CSI/UNAM-CERT podrá irse desarrollando de la mejor manera.

En este apartado me concentro en evaluar el conocimiento de los integrantes sobre las políticas de la CSI/UNAM-CERT, verifico que tengan instaladas en sus equipos de cómputo herramientas que ayuden a establecer la confidencialidad, integridad y disponibilidad de la información.

2.2.3 Proyecto externo: Evaluación de la seguridad de la información en el XLI ENARM

2.2.3.1 Objetivo

Evaluar el cumplimiento de las normas de aplicación del XLI ENARM (Examen Nacional de Aspirantes a Residencias Médicas) y la seguridad de la información del proceso.

[1] Incidente: cualquier evento que afecte la integridad, disponibilidad y/o confidencialidad de la información, así como violar alguna política de la CSI/UNAM-CERT.

2.2.3.2 Antecedentes

La UNAM, con la DGTIC ha participado en el proceso de aplicación del ENARM desde el año 2009, llevando 9 años consecutivos brindando su servicio.

2.2.3.3 Descripción

Con el propósito de que el XLI ENARM (Examen Nacional para Aspirantes a Residencias Médicas) siga brindando un servicio confiable y de calidad por medio de la implantación de los procedimientos necesarios para alcanzar un alineamiento a las mejores prácticas en el manejo seguro de la información, la Dirección General de Calidad y Educación en Salud, entidad dependiente de la Secretaría de Salud federal, solicita a la DGTIC-UNAM un servicio de apoyo integral que permita la evaluación enfocada al Fortalecimiento de la Calidad y Seguridad de la Información para los Procesos del XLI ENARM.

2.2.3.4 Actividades Realizadas

La descripción que hago en este Informe de Actividades Profesionales de las tareas que realicé durante este proyecto se apega a lo señalado en el artículo 36 de la Ley Reglamentaria del artículo 5° Constitucional (Cámara de diputados, 2018), por lo que no revelo información considerada como confidencial, haciendo uso únicamente de la información considerada como pública.

Las actividades que realicé fueron parte de la línea de ***Servicio de evaluación de seguridad de la información en las sedes de aplicación*.**

Dicha actividad consistió en tres tareas: evaluar, observar e identificar oportunidades de mejora, las cuales explico a continuación:

- Observar, con base en los procedimientos descritos en los documentos internos de la Comisión Interinstitucional para la Formación de Recursos Humanos para la Salud (CIFRHS) y DGTIC (UNAM), el cumplimiento de las condiciones de seguridad de la información de las instalaciones e infraestructura tecnológica a utilizar durante la aplicación del ENARM. A partir de la información pública (http://www.cifrhs.salud.gob.mx/site1/enarm/docs/enarm-convo_2017.pdf) puedo describir en este reporte algunas de las revisiones que llevé a cabo, las cuales consistieron en observar el cumplimiento de los puntos siguientes del CIFRHS:
    - 8.2 Verificación de la identidad de los sustentantes
    - 9.1 Verificación del procedimiento de entrega de resultados al sustentante
    - 10.11 Detección de situaciones fraudulentas durante el ENARM
    - 10.15 Detección de dispositivos electrónicos ajenos al ENARM por parte del sustentante

- De forma general puedo mencionar sin afectar la confidencialidad del proyecto, que verifiqué que el recinto tuviera condiciones de seguridad que permitieran resguardar la información, tales como controles a las zonas definidas en las instalaciones de acuerdo con el rol del personal que laboró en el ENARM e indicaciones de protección civil.
- Y que los equipos de cómputo contaran con las especificaciones en materia de seguridad de la información dictadas previamente. Esto lo realicé verificando que los equipos sólo tuvieran el software necesario para las actividades a desarrollar.
- Observar la seguridad física y lógica de las instalaciones e infraestructura tecnológica a utilizar durante la aplicación del ENARM.
- Identificar oportunidades de mejora para fortalecer la seguridad de la información del proceso de aplicación del ENARM.

2.2.3.5 RESULTADOS

Se entregaron reportes diarios con los hallazgos que se encontraron con el propósito de la mejora continua que esto implica el resguardo y continuidad de la aplicación del ENARM.

CAPÍTULO 3. PARTICIPACIÓN EN PROYECTOS EN LA CSI/UNAM-CERT

3.1 AUDITORIA INTERNA

3.1.1 OBJETIVO

Evaluar el cumplimiento, a partir de revisiones objetivas e imparciales, del Sistema de Gestión de Seguridad de la Información (SGSI) adoptado por el CSI/UNAM-CERT, que debe estar apegado al estándar ISO/IEC 27001:2013. Esta evaluación incluye actividades sobre el cumplimiento de los objetivos, planes, programas, proyectos y procesos, así como la identificación de desviaciones en la operación del CSI/UNAM-CERT.

3.1.2 ANTECEDENTES

El ámbito de la seguridad de la información es dinámico, tiene cambios continuos, nuevas amenazas son desarrolladas, nuevas vulnerabilidades son descubiertas, por lo que incidentes de seguridad se presentan de manera continua, sólo es cuestión de tiempo para padecer las consecuencias de esas amenazas y vulnerabilidades si no se tiene un control adecuado.

La UNAM no queda exenta de recibir algún ataque informático, por lo que la Coordinación de Seguridad de la Información al contar con un SGSI puede gestionar la seguridad de la información, al conocer los riesgos, asumirlos, gestionarlos o minimizarlos.

El SGSI de la Coordinación está orientado al servicio de Respuesta a Incidentes, y está basado en el estándar ISO/IEC 27001:2013. Al estar alineado a este estándar, se tiene que cumplir con una serie de cláusulas para que la CSI/UNAM-CERT mantenga la certificación, la cual adquirió en el año 2010. Las cláusulas obligatorias que tiene que cumplir son las siguientes (advisera, 2018):

4. Contexto de la organización
    - Define los requerimientos para comprender cuestiones externas e internas, también define las partes interesadas, sus requisitos y el alcance del SGSI.
5. Liderazgo
    - Define las responsabilidades de la dirección, el establecimiento de roles y responsabilidades y el contenido de la política de alto nivel sobre seguridad de la información.
6. Planificación
    - Define los requerimientos para la evaluación de riesgos, el tratamiento de riesgos, la Declaración de Aplicabilidad (SoA), el Plan de tratamiento

de riesgos y la Determinación de los objetivos de seguridad de la información.

7. Soporte
    - Define los requerimientos sobre disponibilidad de recursos, competencias, concienciación, comunicación y control de documentos y registros.
8. Operación
    - Define la implementación de la evaluación y el tratamiento de riesgos, como también los controles y demás procesos necesarios para cumplir los objetivos de seguridad de la información.
9. Evaluación del desempeño
    - Define los requerimientos para monitoreo, medición, análisis, evaluación, auditoría interna y revisión por parte de la dirección.
10. Mejora
    - Define los requerimientos para el tratamiento de no conformidades, correcciones, medidas correctivas y mejora continua.

Cada año, la organización define las actividades a realizar para cumplir con las cláusulas del ISO/IEC 27001:2013, estas actividades se ven reflejadas a lo largo de un año, aunque hay ocasiones que se excede de un año, al desarrollo de estas actividades se le llama ciclo del SGSI, el cual está orientado al ciclo Plan, Do, Check, Act (PDCA), como se presenta a continuación:

Imagen 3.1. Ciclo PDCA. (All About Lean, 2016)

- Plan: Se establecen los objetivos a alcanzar, las metas y la asignación de personas responsables de que los objetivos y metas se alcancen.
- Do: Es poner en práctica lo establecido en la fase Plan.

- Check: Comparación, análisis y evaluación de los resultados reales obtenidos en la fase Do con los esperados en la fase Plan.
- Act: Esta fase es necesaria para corregir los aspectos negativos obtenidos en la fase anterior Check y que puede implicar la modificación del Plan. Lo importante de esta fase son las lecciones aprendidas para cuando empiece la fase de Plan del siguiente ciclo.

La auditoría se lleva a cabo en la fase "Check" ya que se verifica lo que se hizo a lo largo del ciclo y se reportan los hallazgos en el informe final.

Existen tres tipos de auditorías (Temprado, 2015):

- Auditoría de primera parte: Llamadas auditorías internas, son realizadas por o en nombre de la propia organización para fines internos, con el fin de la mejora continua.
- Auditoria de segunda parte: Se desarrollan por el interés de una organización hacia un tercero, son las realizadas a los proveedores.
- Auditoría de tercera parte: Realizadas por organizaciones auditoras independientes y externas a la organización, que proporciona la certificación de que el sistema cumple con el estándar de referencia.

La auditoría interna viene especificada dentro de la cláusula 9, en el punto 9.2 de la norma ISO/IEC 27001:2013, por lo que es importante realizar dicha actividad, ya que forma parte de los requerimientos obligatorios. Además, dicha auditoría debe realizarse antes de la auditoría externa (auditoria de tercera parte), ya que en ésta se revisa la cláusula que mencioné anteriormente.

Hasta el año 2017, se han realizado en la CSI/UNAM-CERT 10 auditorías, siendo la número 10 la que yo realicé.

3.1.3 Descripción (trabajo en grupos multidisciplinarios)

El propósito de la auditoría interna es llevar a cabo una revisión imparcial y objetiva, a partir de evidencias, sobre el cumplimiento de los objetivos, planes, programas y procesos, así como la identificación de desviaciones en la operación del CSI/UNAM-CERT, esto mediante trabajo en grupos multidisciplinarios, cuyos perfiles son técnicos (conocimiento de sistemas operativos, redes, programación, etc.) y no técnicos (actividades de documentación, uso de paquetería office, etc.). Del mismo modo, verificar el apego a los requisitos y controles seleccionados del estándar ISO/IEC 27001:2013.

Los resultados de la revisión apoyan en la toma de decisiones por parte de la alta dirección, en la implementación de controles necesarios para corregir fallas o en la aplicación de acciones de mejora.

La auditoría interna al Sistema de Gestión de Seguridad de la Información (SGSI) debe contemplar los siguientes elementos: (udo, 2011)

- Elaboración de un programa de auditoría.
    - Alcance del Sistema de Gestión de Seguridad de la Información;
    - Frecuencia para llevar a cabo las auditorías;
    - Auditores responsables de llevar a cabo la auditoría.

- Creación del plan de auditoría para cada auditoría programada.
    - Objetivo para el programa de auditoría y auditorías individuales;
    - alcance, número, tipos, duración, ubicación y cronograma de las auditorías;
    - procedimientos del programa de auditoría;
    - criterios de auditoría;
    - métodos de auditoría;
    - selección de equipos auditores;
    - recursos necesarios;
    - procesos para manejo de confidencialidad, integridad y disponibilidad de la información.
- Verificación de la correcta aplicación de la normatividad que autorregula el sistema de control interno, así como los procedimientos, instructivos e instrumentos que garantizan el diseño, implementación, mantenimiento y evaluación del sistema.
- Inclusión de los hallazgos en el informe de auditoría interna, luego del proceso de auditoría.
- Discusión previa de los informes de auditoría con todos los niveles de autoridad y responsabilidad correspondientes.
- Entrega y difusión oportuna de los informes de auditoría para la dirección de la CSI/UNAM-CERT.

- Presentación del informe final a las instancias responsables o auditadas, y la formulación de un plan de mejoramiento para subsanar sus hallazgos.
- Seguimiento posterior a la auditoría con el fin de verificar la implementación de mejoras.

3.1.3 Actividades Realizadas

De acuerdo con el procedimiento de auditoría, elegí un Auditor Adjunto que me auxilió a lo largo de la Auditoría Interna, después elaboré un plan de auditoría el cual consistió en redactar un objetivo, un alcance (necesario para delimitar lo que se auditaría) y definir un criterio para la auditoría (elementos del SGSI susceptibles a ser revisados). En el plan definí por días lo que se iría revisando para que los auditados (en este caso, el personal de la CSI/UNAM-CERT) estuvieran enterados y disponibles, para dar inicio formal tuve que realizar una junta de apertura donde las actividades que se realizaron fueron:

- Confirmación del plan de auditoria.
- Un breve resumen de cómo se llevarían a cabo las actividades.
- Duración de la auditoría (5 días hábiles).
- Confirmar los canales de comunicación.

Una vez explicado esto, proseguí a realizar la revisión e ir documentando los hallazgos encontrados.

A continuación, explico en qué consistió la revisión con base en el ISO 27001:

4. Contexto de la organización
    - Revisé alcance del SGSI en la CSI, de tal forma que se pueda entender a la organización y su contexto, así como las necesidades y expectativas de las partes interesadas. También, revisé que tengan definido una política de seguridad.
5. Liderazgo
    - Mediante entrevistas que realicé a los integrantes a la CSI, pude saber el conocimiento que tienen sobre el sistema, así como su rol dentro del SGSI, de tal forma que la alta dirección esté tomando su papel de liderazgo, para concientizar a su personal.
6. Planificación
    - Me concentré en el análisis de riesgos que tienen definido, donde los puntos que revisé son los siguientes:
        - La definición sobre qué criterios tomaron en cuenta para los riesgos de seguridad de la información.

        - De acuerdo al análisis de riesgos, cada uno de esos riesgos esté asociado a un dueño, es decir, a un responsable.
        - En la evaluación de riesgos, que tengan al menos impacto y probabilidad. Donde el impacto, en cuanto a la seguridad de la información, está definido por la confidencialidad, integridad y disponibilidad.
    - La CSI definió un tratamiento de riesgos, de tal forma para reducir ese impacto. La revisión de este, consistió en:
        - Observar si hubo reducción de riesgo.
        - Que se haya aprobado el plan por parte de la alta dirección.
        - De acuerdo al Declaración de Aplicabilidad (SoA), revisar que contengan los 114 controles del Anexo A del ISO 27001, así como a la justificación del porqué se aplican o no. Así como a la revisión de la documentación del CSI que aplican a los controles.
7. Soporte
    - Revisé que la CSI tenga definido competencias mínimas que los integrantes deban tener. Con las entrevistas que mencioné que realicé, pude obtener evidencia sobre si el personal sabe cómo contribuye al SGSI y crear una concientización sobre qué pasa si no realiza sus actividades. Y, por último, revisé la forma en que la CSI trata su documentación, es decir, como controla su documentación, desde cómo la crea, identificación, descripción, revisión y aprobación, así como la preservación del archivo muerto y la forma en que desecha un documento.
8. Operación
    - En esta parte ya se ejecutó el análisis de riesgos, por lo que la revisión consistió, en cómo se llevó a cabo. Que, el documento del análisis de riesgos esté disponible, así como de los resultados del plan de tratamiento de riesgos.
9. Evaluación del desempeño
    - Esta etapa revisé los resultados después del tratamiento de riesgos, es decir, su monitoreo, medición, análisis y evaluación. Además, en esta parte, como lo comentaba anteriormente, realizo la auditoría interna. Una vez que se termina, la alta dirección le debe dar seguimiento a los resultados de esta. Por último, reviso los resultados que la alta dirección documentó, de tal forma para validar que se llevó a cabo la revisión.
10. Mejora
    - De las no conformidades que reporté, la CSI, debe realizar un plan de acciones correctivas, es decir, un análisis de causa para saber el origen del problema, como auditor revisé dicho análisis.

Terminé con la revisión y generé mi informe de la auditoria donde documenté los hallazgos de acuerdo con su nivel de criticidad:

- No conformidad mayor
  - Incumplimiento de un requisito del estándar ISO 27001:2013 que, por su gravedad impide el desarrollo del ciclo del SGSI.
- No conformidad menor
  - Incumplimiento de un requisito del estándar ISO 27001:2013 que, por su gravedad, es leve, ya que sólo afecta a algún punto del estándar.
- Observaciones
  - Comentario que se realiza sobre una deficiencia encontrada.
- Oportunidades de mejora
  - Situaciones identificadas en la ejecución de un proceso que no son consideradas una observación, pero que atenderlas o mejorarlas fortalece el proceso.
- Fortalezas
  - Hallazgos que indican un punto fuerte del sistema o proceso.

Una vez documentado los hallazgos en el informe, lo presenté ante la dirección en la junta de cierre donde el objetivo fue dar a conocer al auditado los hallazgos encontrados, expliqué la razón por la cual se reportó, una vez que el auditado estaba de acuerdo con los hallazgos presentados, se finalizó la junta.

En mutuo acuerdo con el auditado, se establece una fecha para la solución de dichos hallazgos, sobre todo con las no conformidades. Como actividad final, de acuerdo con los hallazgos les di seguimiento para verificar que se completaron las acciones y su eficacia.

3.1.4 RESULTADOS

El informe de auditoría proporciona un registro completo, preciso, conciso y claro de la auditoría y hace referencia a lo siguiente:

- objetivos de la auditoría;
- alcance de la auditoría, particularmente la identificación de las unidades de la organización;
- la identificación del cliente de la auditoría;
- la identificación del equipo auditor y de los participantes del auditado en la auditoría;
- las fechas y ubicaciones en donde se realizaron las actividades de la auditoría;
- los criterios de la auditoría;
- los hallazgos de la auditoría y las evidencias relacionadas;

- las conclusiones de la auditoría.

Los resultados de la auditoría ayudaron de tal forma que la CSI/UNAM-CERT se diera cuenta de:

- problemas que se identificaron y corrigieron antes de que se llevara la auditoría externa,
- oportunidades de mejora,
- concientización al personal para el cumplimiento de políticas de acuerdo con el SGSI.

Con la realización de la auditoría se cumplió con un requisito importante del estándar, para así estar preparados a la auditoría externa (auditoría de tercera parte).

3.2 Análisis Estático de Vulnerabilidades de Código PHP

"La metodología ágil Scrum es un marco de trabajo por el cual las personas pueden acometer problemas complejos adaptativos, a la vez que entregar productos del máximo valor posible productiva y creativamente" (Guía Scrum, 2013). Los beneficios de utilizar Scrum son los siguientes:

- entrega mensual o quincenal de los requisitos más prioritarios, lo que proporciona la mitigación sistemática de los riesgos del proyecto ya que el cliente va viendo los avances que se tienen y en caso de alguna modificación hacerla lo antes posible y no cuando el producto esté concluido;
- productividad y calidad;
- alineamiento entre el cliente y el equipo de desarrollo.

Para el desarrollo de este proyecto se tomaron las mejores prácticas de Scrum. La razón del porqué no se siguió al pie de la letra todo lo que involucra (o "todo lo que dicta") Scrum, fue porque no se contaba con el personal necesario para tomar el papel de los diferentes roles que especifica esta metodología, más adelante detallo esta explicación.

A continuación, describo los diferentes roles que especifica la metodología Scrum:

- Dueño de Producto (Product Owner): responsable de maximizar el valor del producto y del trabajo del Equipo de Desarrollo.
- Equipo de Desarrollo (Development Team): profesionales que desempeñan el trabajo de entregar un incremento de producto "Terminado", que potencialmente se pueda poner en producción.
- Scrum Master: responsable de asegurar que Scrum es entendido y adoptado. Los Scrum Masters hacen ésto asegurándose de que el equipo Scrum trabaja ajustándose a la teoría, prácticas y regla de Scrum.

Se generó una épica[2] en donde se describe de manera general, el propósito de la herramienta y con base en esta épica, se desarrollan historias de usuarios, las cuales consisten en describir las funcionalidades que el proyecto tendrá. Una vez definido dichos documentos, el equipo de desarrollo define una lista de tareas donde cada integrante del equipo tomará una tarea y definirá el tiempo que tardará en terminarla, no pasando de una jornada laboral (8 hr.) para cada tarea. (Guía Scrum, 2013)

Los documentos que anteriormente se mencionaron se pueden ver en el Anexo A.

3.2.1 OBJETIVO

De acuerdo con la épica (Anexo A) dada por el Product Owner (este rol lo asumió el responsable del Área de Auditoria y Nuevas Tecnologías de la CSI/UNAM-CERT) se describe que se tiene que desarrollar una herramienta que realice análisis estático de código fuente en aplicaciones desarrolladas en PHP, buscando funciones inseguras y malas prácticas de programación, es decir, que se estén ocupando funciones obsoletas o que no estén bien implementadas y por lo tanto puedan tener brechas de seguridad. Además, como resultado del análisis, mostrar en un reporte en formato CSV y HTML los hallazgos detectados, con la finalidad de mostrar ante directivos de una forma clara y visual lo que la aplicación carece, en el caso del reporte en HTML.

3.2.2 ANTECEDENTES

Con base en los requerimientos que se proporcionaron, se estudiaron los antecedentes que el Product Owner dio a conocer, los cuales describo a continuación detalladamente.

Con el rápido crecimiento que ha tenido el internet, es importante realizar pruebas de seguridad al software. Es recomendable hacerlo durante la fase de desarrollo, debido a que permite corregir los errores antes de ponerse en producción. Generalmente lo anterior no se realiza, esto representa un costo económico considerable, que pocas organizaciones están dispuestos a pagar. Una vulnerabilidad común en las aplicaciones web son las inyecciones SQL (SQLi, del inglés SQL injection), si no se realizan las correctas validaciones para cuando el usuario ingresa un dato, un usuario malintencionado pudiera sacar provecho de ésto al obtener información de la base datos que posiblemente sea confidencial.

Personas que no tienen suficiente conocimiento sobre la seguridad de la información la tratarán en segundo plano y como consecuencia, se detectan errores de programación en la aplicación, en su mayoría, después de ponerse en producción, ya sea a través del uso

[2] Se denomina épica a una historia de usuario que, por su gran tamaño, el equipo descompone en historias con un tamaño más adecuado para ser gestionada con los principios y técnicas ágiles. (Scrum Manager, 2014)

del mismo software o con los ataques detectados y reportados. Estos errores se controlan con parches, los cuales se desarrollan para corregir dichos errores, pero hay ocasiones que pueden originar nuevas vulnerabilidades, que, si no se atienden, podrían originar un riesgo importante para el sistema en donde se ejecuta. En el software se tienen que incorporar características de seguridad durante todo el proceso de desarrollo.

Durante el 2016, el lenguaje de programación PHP estuvo dentro de los diez lenguajes más utilizados por los desarrolladores de aplicaciones (QuincyLarson, 2016). En la siguiente *imagen* 3.2, muestro los lenguajes de programación ordenados de acuerdo con su porcentaje de uso en diferentes plataformas (sitios web, celulares, computadoras y hardware), se observa que PHP se ubica en el séptimo lugar con 82.8% que, además, sólo es utilizado para el desarrollo de sitios web.

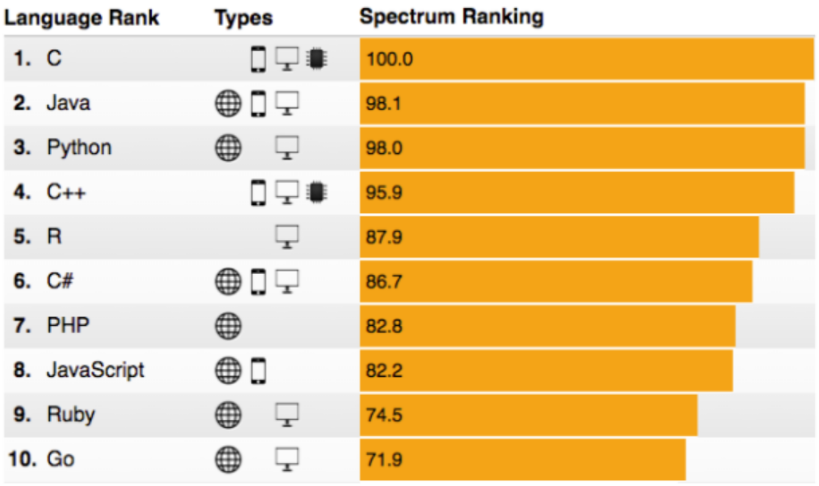


Imagen 3.2. Lenguajes de programación utilizados en aplicaciones (QuincyLarson, 2016).

Para darnos una idea de cuantos sitios utilizan PHP, al día 19 de julio de 2018 al ejecutar la herramienta Shodan[3], en la *imagen 3.3*, nos muestra un top 5 de los países, tipo de servicio y organizaciones en donde utilizan PHP.

[3] Shodan es el primer motor de búsqueda para dispositivos conectados a internet (shodan, 2018) (routers, servidores, etc.)

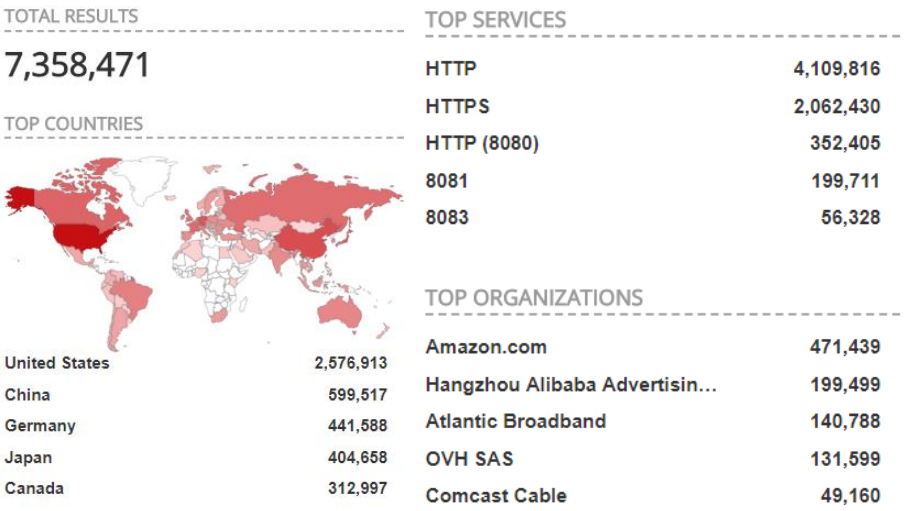
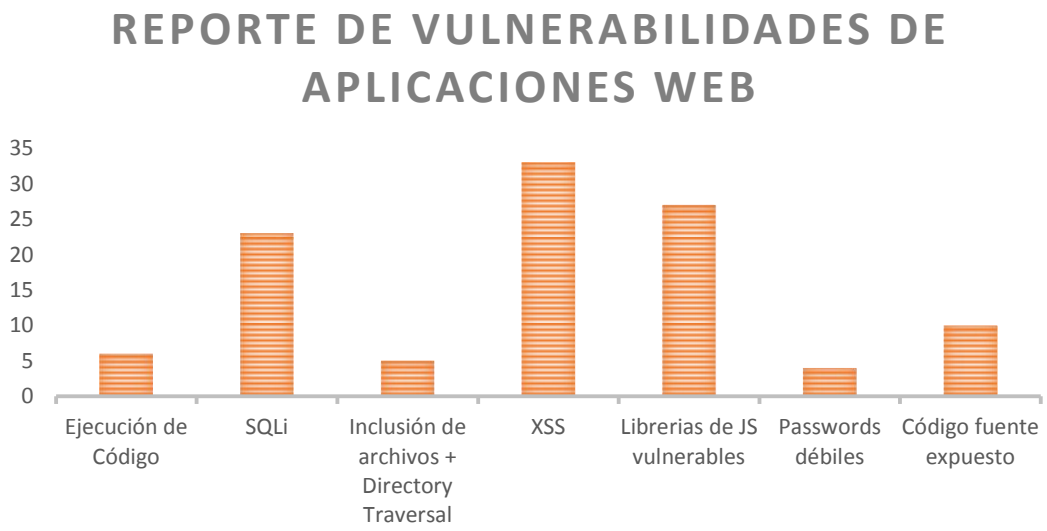


Imagen 3.3. Resultados de los lugares donde utilizan PHP.

Ahora, viendo que PHP es uno de los lenguajes que más se utiliza, en la *gráfica 3.1* se muestra las vulnerabilidades en aplicaciones web que la herramienta Acunetix [4] detectó durante el año 2016.



Gráfica 3.1. Reporte de vulnerabilidades de aplicaciones web (Acunetix, 2016).

El lenguaje de programación PHP se utiliza mucho para el desarrollo de aplicaciones web, por lo que es de suma importancia el saber cómo implementar las funciones que se han

[4] Acunetix es un escáner de vulnerabilidades de aplicaciones web. (Acunetix, 2017)

ido actualizado a lo largo de cada versión de PHP, ya que, al utilizar funciones obsoletas o emplearlas de forma equivocada, se pueden explotar vulnerabilidades (véase Gráfica 1) asociadas a éstas y generar un peligro considerable, como el robo de datos. Por lo que, el lenguaje en sí no es vulnerable, sólo es cuestión de implementar de forma correcta las funciones.

Razón por la cual se decidió hacer la herramienta para ayudar a los programadores a detectar esas vulnerabilidades y así, poder corregirlas en un menor tiempo.

3.2.3 Descripción

Una forma de mitigar esas vulnerabilidades es mediante el análisis de código estático, el cual consiste en analizar código a alto nivel sin ejecutarlo, se lleva a cabo en la fase de implementación (Owasp, 2017). Este análisis, ayuda a identificar posibles vulnerabilidades asociadas al software, analiza la complejidad involucrada (ya sea por la cantidad de líneas de código) y el impacto en las experiencias del usuario al solucionar estas vulnerabilidades a través de controles de seguridad apropiados. El análisis de código estático facilita la evaluación de los riesgos involucrados sólo mitigando o dejando estas vulnerabilidades sin supervisión, las consecuencias de posibles ataques es el costo de desarrollar controles de seguridad e integrarlos al software después de dicho ataque.

El proyecto lo elaboré con una compañera. Realizamos una lista de tareas que se puede ver en el Anexo A, las cuales salieron de la épica, en la que posteriormente cada uno seleccionó las tareas que de acuerdo con nuestras aptitudes podíamos realizar. La ventaja de haber realizado este tipo de actividad facilitó el desarrollo de estas, ya que cada uno sabía lo que se le facilitaba y las cosas que no, entre los dos las desarrollábamos, es una de las ventajas de ocupar Scrum, no sólo centrarnos en nuestras tareas sino también involucrarnos en las demás.

Las vulnerabilidades que la herramienta detecta son las siguientes:

- Inyección de código SQL (SQLi): inserción o "inyección" de una consulta SQL a través de los datos de entrada del cliente a la aplicación. (W3SCHOOLS, 2018)
- Cross Site Scripting (XSS): abarca cualquier ataque que permita ejecutar código de "scripting", como VBScript o Javascript, en el contexto de otro dominio. Estos errores se pueden encontrar en cualquier aplicación HTML, no se limita a sitios web, ya que puede haber aplicaciones locales vulnerables a XSS, o incluso el navegador en sí. (OWASP, 2018)
- Generación débil o inapropiada de cookies de sesión: se centra en la predicción de valores de ID de sesión que permiten a un atacante eludir el esquema de autenticación de una aplicación. Al analizar y comprender el proceso de

generación de ID de sesión, un atacante puede predecir un valor de ID de sesión válido y obtener acceso a la aplicación. (OWASP, 2011)

- Envío de información sensible a través de métodos inseguros: consisten en métodos de envío donde la información viaja en claro, no tiene ningún modo de cifrado para impedir que la información sea leída.
- Implementación de funciones obsoletas: Consiste en funciones que fueron identificadas con vulnerabilidades y fueron sustituidas por otras que solucionaban el error asociado a esa vulnerabilidad.
- Local File Inclusion (LFI): Es el proceso de incluir archivos, que ya están presentes localmente en el servidor, a través de la explotación de los procedimientos de inclusión vulnerables implementados en la aplicación. (Acunetix, 2017)
- Remote File Inclusion (RFI): Es el proceso de incluir archivos remotos a través de la explotación de los procedimientos de inclusión vulnerables implementados en la aplicación. (Acunetix, 2017)
- Path Traversal: Consiste en acceder a los archivos y directorios que se almacenan fuera de la carpeta raíz web. Al manipular variables que hacen referencia a archivos con secuencias "punto-punto-barra (../)" y sus variaciones o mediante el uso de rutas de archivos absolutas. (OWASP, 2015)
- Inyección de comandos: Se refiere a la capacidad de un usuario, que controla la entrada de comandos (bien a través de un terminal de Unix/Linux o del interfaz de comando de Windows), para ejecutar instrucciones que puedan comprometer la integridad del sistema. (OWASP, 2018)
- Inyección de código fuente: Esta debida a la inclusión de la función include() la cual permite el enlace de archivos situados en otros servidores, mediante los cuales se puede ejecutar código PHP en el servidor. Se utilizan las funciones include, include_once, require y require_once, las cuales son utilizadas para incluir en una página web otras páginas, por tanto, el atacante podrá obtener una Shell en el servidor de la víctima y ejecutar un archivo. Para que se pueda ejecutar dicho archivo debe tener una extensión diferente a ".php" ya que, con esta extensión, el archivo se ejecutaría en el servidor del atacante y no en el de la víctima, así un archivo ".txt", ".gif", etc. serían algunos de los más adecuados. (OWASP, 2013)

3.2.4 DESARROLLO

En cuanto al lenguaje de programación para el desarrollo de la herramienta, elegimos Python, principalmente por la facilidad de aprendizaje, además, considero que es más intuitivo su uso comparado con otros lenguajes de programación. Muestra simplicidad en cuanto al manejo de archivos y el lenguaje cuenta con suficiente documentación para

consultar en caso de dudas, como, por ejemplo, el cómo utilizar ciertas funciones. Agregado a estas ventajas, Python cuenta con una gran comunidad de desarrolladores que están dispuestos a compartir su conocimiento a través de diversos foros en Internet.

Como al inicio lo comenté, tomamos las mejores prácticas de Scrum para realizar el proyecto, ya que facilita la administración de proyectos de cualquier tamaño y complejidad, así como el flujo de información y la comunicación entre el equipo de trabajo. Esta metodología la conocemos bien, ya que la llegamos a analizar durante nuestra estancia en la Facultad de Ingeniería en las materias de Desarrollo de Software Seguro y Administración de Proyectos de Software.

Si bien, existen varios roles al utilizar Scrum, al ser sólo dos personas desarrollando la herramienta, el rol de Scrum master lo llevamos a cabo entre las dos. Sólo estaba definido el rol de Product Owner y el del equipo de desarrollo.

Cuando recibimos el proyecto, realizamos nuestro Sprint[5] con todo el equipo Scrum, acordamos que sólo habría un Sprint (una reunión, para tratar la duración del desarrollo de la herramienta), en donde lo primero que hicimos fue realizar un panel de tareas, donde cada uno se asignaba una actividad, estableciendo un tiempo para hacerla, dicho panel lo dividimos en tres columnas que representaban el estado de la tarea:

- Por hacer (To Do)
- Haciendo (Doing)
- Terminado (Done)

A modo de ejemplo del panel y de las columnas, se muestra la *imagen 3.4*, donde el acomodo de estas tres columnas corresponde a la metodología Kanban

[5] Sprint es un bloque de tiempo (time-box) de un mes o menos durante el cual se crea un incremento de producto “Terminado”, utilizable y potencialmente desplegable. (Guía Scrum, 2013)

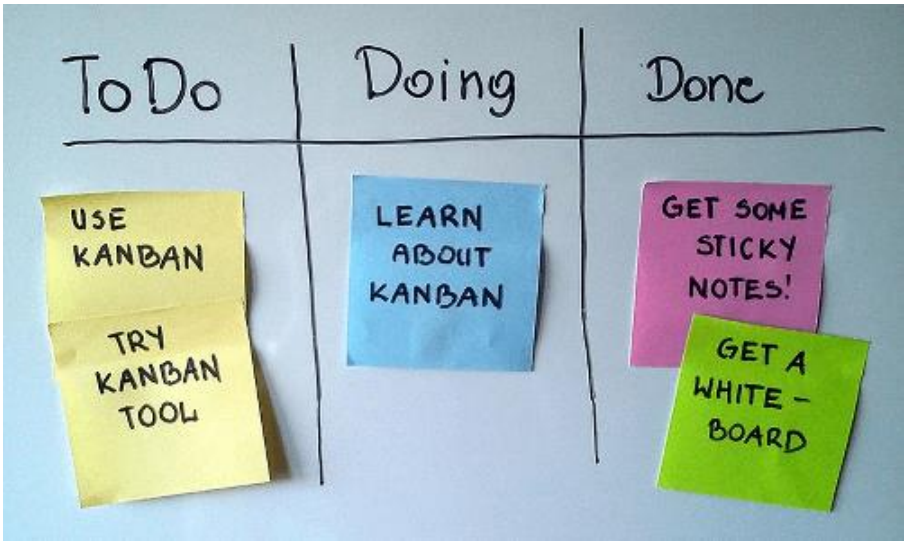


Imagen 3.4. Tablero de la metodología Kanban *(kanban tool, 2018)*

Acordamos diariamente mostrar nuestros avances para ver cómo iba el otro, en sesiones de máximo 10 min. Otra de las ventajas de utilizar Scrum consiste en revisar con tu equipo los avances constantes que se tienen y ayudarse mutuamente en caso de encontrar alguna dificultad.

Ya que teníamos un avance con las tareas realizadas, lo presentábamos con el encargado del proyecto, para validar que se estaban llevando a cabo de la mejor manera.

Las actividades que realicé fueron las siguientes:

- Generación de expresiones regulares para la detección de funciones obsoletas y vulnerabilidades para XSS, SQLi, generación insegura de cookies de sesión, envío de información a través de medios inseguros, local file inclusion, remote file inclusion, path traversal, inyección de comandos e inyección de código fuente.
- Generación de la base de datos para el almacenamiento de los reportes (en formatos CSV y HTML) y expresiones regulares.
- Generación de una función para la creación de reportes en formato HTML para mostrar los hallazgos encontrados.
- Scripts de instalación y desinstalación de la herramienta.

3.2.5 FUNCIONAMIENTO

La herramienta recibe archivos con extensión "*php*", independientemente de la opción que se eligió, la herramienta consultará a la base de datos para ocupar las expresiones regulares asociadas a las vulnerabilidades y hacer la detección. Por cada ejecución de la herramienta se generarán dos reportes, un reporte hecho en HTML que será más visual, ya que la información está ordenada en tablas y con gráficas que ayudan a identificar la cantidad de posibles vulnerabilidades. Y Un reporte CSV, para que se pueda procesar de acuerdo con las necesidades de cada programador.

Cada ejecución hecha a la aplicación irá guardando en la base de datos el nombre del reporte creado, así como las vulnerabilidades que se detectaron.

En la *imagen 3.5*, presento un diagrama de bloques representando la funcionalidad de la herramienta.

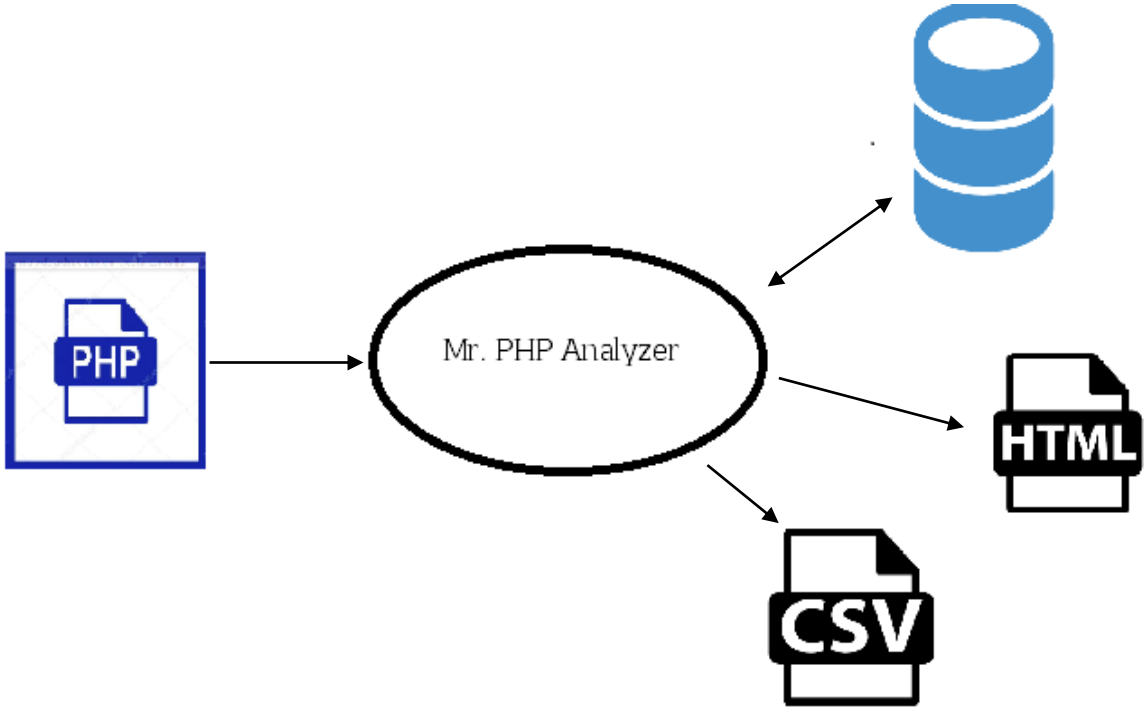


Imagen 3.5. Diagrama de bloques de la herramienta.

En el siguiente apartado daré una explicación general de lo que contiene cada archivo, así como también del funcionamiento de la herramienta, desde la instalación hasta su ejecución.

3.3.5.1 ESTRUCTURA DE LA HERRAMIENTA

A continuación, en la *imagen 3.6*, se detalla el contenido de cada directorio, así como la descripción de los scripts.

```
root@pfinal:/home/pfinal/Documents/Mr.-PHP-Analyzer# ls
base           Fraudatanalyzer.jpg  puzzle        reports
Documentacion  install.sh           README.md     uninstall.sh
file           mrphpanalyzer        reportesHTML
```

Imagen 3.6. Archivos que contiene la herramienta.

- base
  - Carpeta en donde se encuentra el script de la creación de la base de datos.
- Documentación
  - Carpeta que almacena el archivo que describe la forma de utilizar la herramienta.
- file
  - Carpeta que almacena un archivo "out" que contiene todos los archivos analizados con extensión php.
- Fraudatanalyzer.jpg
  - Imagen que contiene la estructura de la base de datos.
- install.sh
  - Script que hace la instalación de la herramienta.
- mrphpanalyzer
  - Comando que se estará utilizando para ejecutar la herramienta una vez que se haya instalado.
- puzzle

```
puzzle/
├── csvcreate.py
├── files.py
├── function.py
├── generateHtml.py
├── his.py
├── main.py
├── report.py
├── version.py
└── vulnerability.py
```

Imagen 3.7. Carpeta que contiene los archivos necesarios para que funcione la herramienta.

  - csvcreate.py
    Script en donde se hace la creación del reporte en formato "CSV" cada vez que la herramienta es utilizada.
  - files.py
    - Script en donde se hace la lectura de cada uno de los archivos con extensión php.
  - function.py

        - Script en donde se estará detectando las funciones obsoletas y se irán guardando en una lista para que sea utilizada por el archivo report.py. También se está creando parte del reporte en formato HTML para mostrarse con base en lo analizado.
    - generateHtml.py
        - Script que generará el archivo HTML, para la creación del reporte, está recolectado toda la información de los archivos function.py y vulnerability.py.
    - his.py
        - Script que genera un archivo HTML que mostrará los archivos que se han generado con la herramienta.
    - main.py
        - Script que contiene la parte central de la herramienta, muestra el menú y ayuda para ejecutar la herramienta.
    - report.py
        - Script donde dependiendo si es función obsoleta o vulnerabilidades se hará el análisis para ir agregando a la base de datos los datos encontrados.
    - version.py
        - Script en donde se detectará la versión de la aplicación al cual está analizando la herramienta.
    - vulnerability.py
        - Script en donde se está detectando las vulnerabilidades y se irán guardando en una lista para que sea utilizada por el archivo report.py. Los hallazgos encontrados se irán almacenando para mostrarse en el reporte en formato HTML.
- README.md
    - Archivo en donde se da una explicación general con lo que cuenta la herramienta.
-
    - Carpeta en donde se irán almacenando los reportes en formato "CSV.

- reportesHTML
    - Carpeta donde se irán almacenando los reportes en formato "HTML".

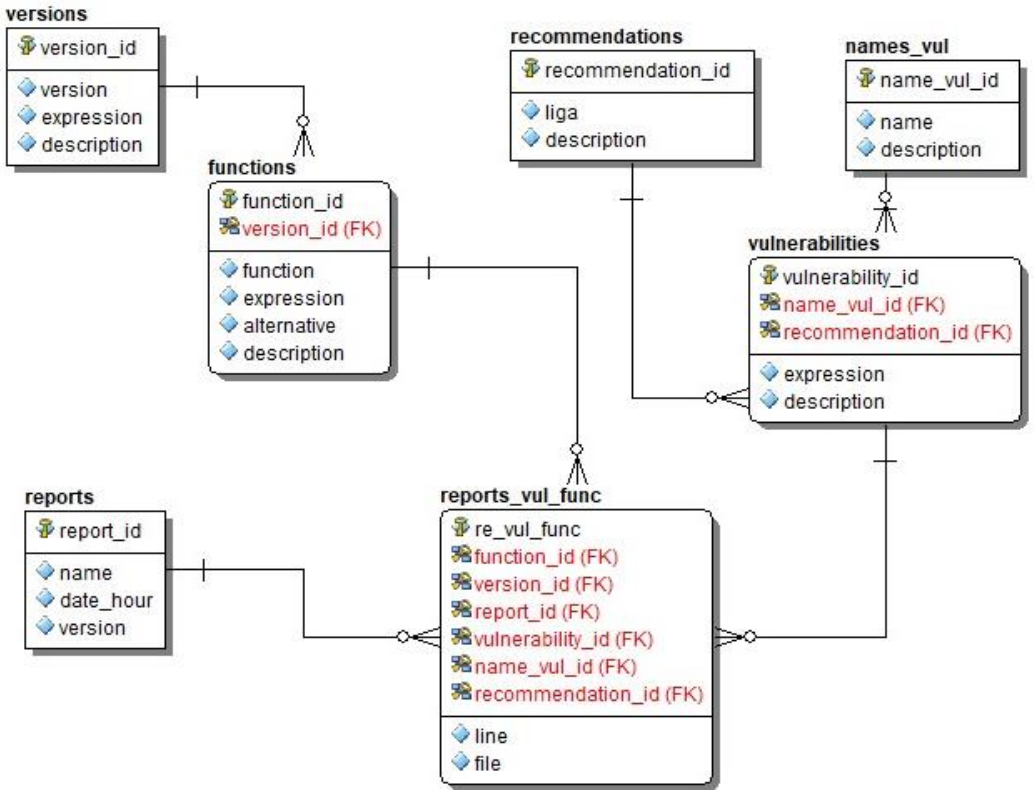
3.2.5.1 ESTRUCTURA DE LA BASE DE DATOS

En la *imagen* 3.8 muestro el diagrama entidad-relación que diseñé para la base de datos que usa la herramienta.

Al momento que realicé el diagrama, seguí las formas normales (Cruz, s.f.) y validé que se cumplieran.

- 1° Forma Normal
  - En cada valor válido de esa relación, toda tupla contiene exactamente un valor para cada atributo.
- 2° Forma Normal
  - Está en 1° FN y todo atributo que no sea clave es dependiente irreduciblemente de la clave primaria.
- 3° Forma Normal
  - Está en 2° FN y todos los atributos que no son clave son dependientes en forma no transitiva de la clave primaria.
  -
  -

EN LA SIGUIENTE IMAGEN SE MUESTRA EL DIAGRAMA QUE SE UTILIZ



Descripción de las tablas:

- versions: tabla que contiene las expresiones regulares para detectar la versión de PHP.

- functions: tabla que contiene las expresiones regulares para detectar las funciones obsoletas.
- recommendations: tabla que contiene enlaces que serán de ayuda para consultar sobre cómo mitigar las vulnerabilidades.
- names_vul: tabla que contiene el nombre de las vulnerabilidades.
- vulnerabilities: tabla que contiene las expresiones regulares para detectar las vulnerabilidades.
- reports: tabla que contiene el nombre del proyecto analizado, así como la fecha de creación del reporte.
- reports_vul_func: tabla que contiene la línea vulnerable, archivo, id de la vulnerabilidad e id del reporte.

En el Anexo C se detalla a profundidad el contenido de cada una de las tablas.

3.2.5.2 INSTALACIÓN

A continuación, se describe la forma de ejecutar la herramienta. En la *imagen 3.9,* se ejecuta el comando para instalar la herramienta:

```
root@pfinal:/home/pfinal/Documents/Mr.-PHP-Analyzer# bash install.sh
```

Imagen 3.9. Comando para la instalación de la herramienta.

Donde en la carpeta /opt/ se encontrará la herramienta con sus respectivos archivos.

```
root@pfinal:/opt# ls
mrphpanalyzer
```

```
root@pfinal:/opt/mrphpanalyzer# ls
base           Fraudatanalyzer.jpg  puzzle        reports
Documentacion  install.sh           README.md     uninstall.sh
file           mrphpanalyzer        reportesHTML
```

Imagen 3.10. Verificación de la instalación de la herramienta.

3.2.5.3 EJECUCIÓN

A continuación, muestro el proceso de ejecución de la herramienta.

Para mostrar la opción de ayuda basta con agregar la opción "-h" después del nombre de la herramienta.

```
root@pfinal:~# mrphpanalyzer -h
usage: main.py [-h] [-r R] [-v V] [--sos]

Finds vulnerabilities in php code

optional arguments:
  -h, --help  show this help message and exit
  -r R        path
  -v V        vulnerability
  --sos       actions list
```

Imagen 3.11. Opción de ayuda de la herramienta.

Para ver las vulnerabilidades disponibles es necesario agregar la opción "--sos".

```
root@pfinal:~# mrphpanalyzer --sos


                Herzlich willkommen bei Mr.-PHPAnalyzer
                Mr.-PHPAnalyzer find vulnerabilities in php code

                Options:

                1. SQLi
                2. XSS
                3. SESSION'S COOKIES
                4. SEND SENSIBLE INFORMATION
                5. LFI & RFI
                6. PATH TRAVERSAL
                7. COMMAND INJECTION
                8. SOURCE CODE INJECTION
                9. OBSOLETE FUNCTIONS
```

Imagen 3.12. Opción para realizar la búsqueda de vulnerabilidades.

3.2.6 PRUEBAS

A continuación, presento la funcionalidad de la herramienta con la comprobación de la detección de vulnerabilidades en 3 distintas aplicaciones, los cuales corresponden a tiendas online. Las pruebas se realizaron en el mismo equipo en que la herramienta se instaló, las aplicaciones no estaban en línea, se encontraban locales.

El comando que se utilizó para cada una de las vulnerabilidades fue el siguiente:

```
root@kali:~# mrphpanalyzer -v -1 -r /var/www/html/tienda/
```

Imagen 3.13. Comando utilizado para la ejecución de la herramienta.

Sólo estuve cambiando la opción de la vulnerabilidad.

3.2.6.1 Detección SQLi

Al correr la herramienta se generó el reporte HTML, el cual encontró 65 posibles vulnerabilidades.

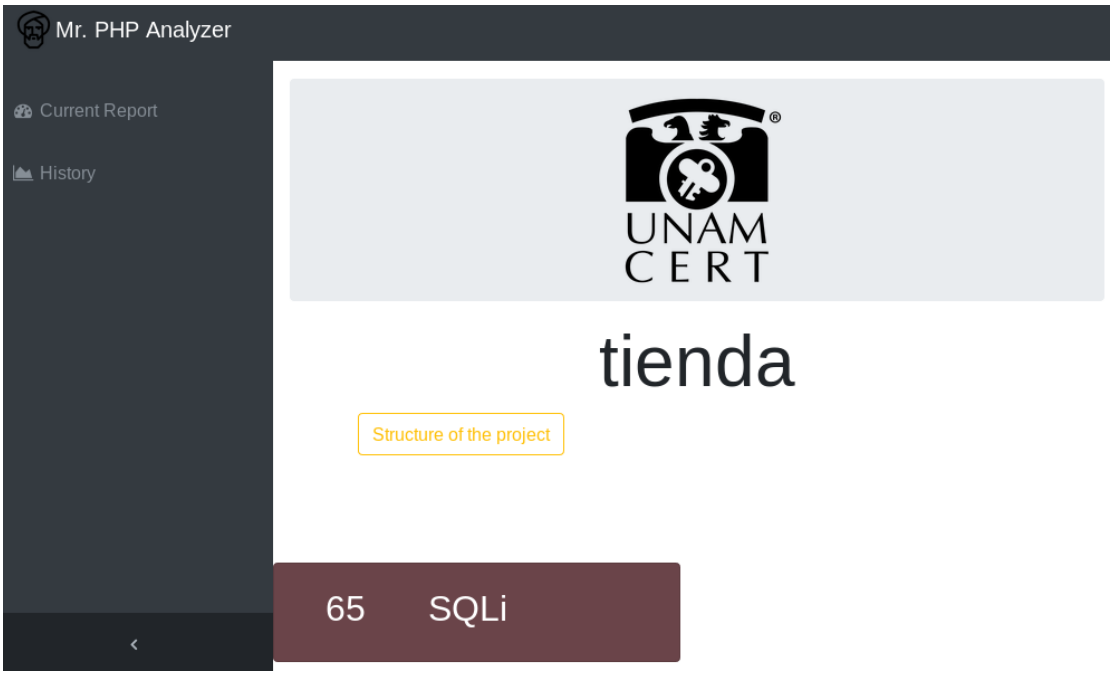


Imagen 3.14. Reporte de la aplicación tienda, SQLi.

Probé la primera, se encuentra en el archivo *consultapedido2.php*.

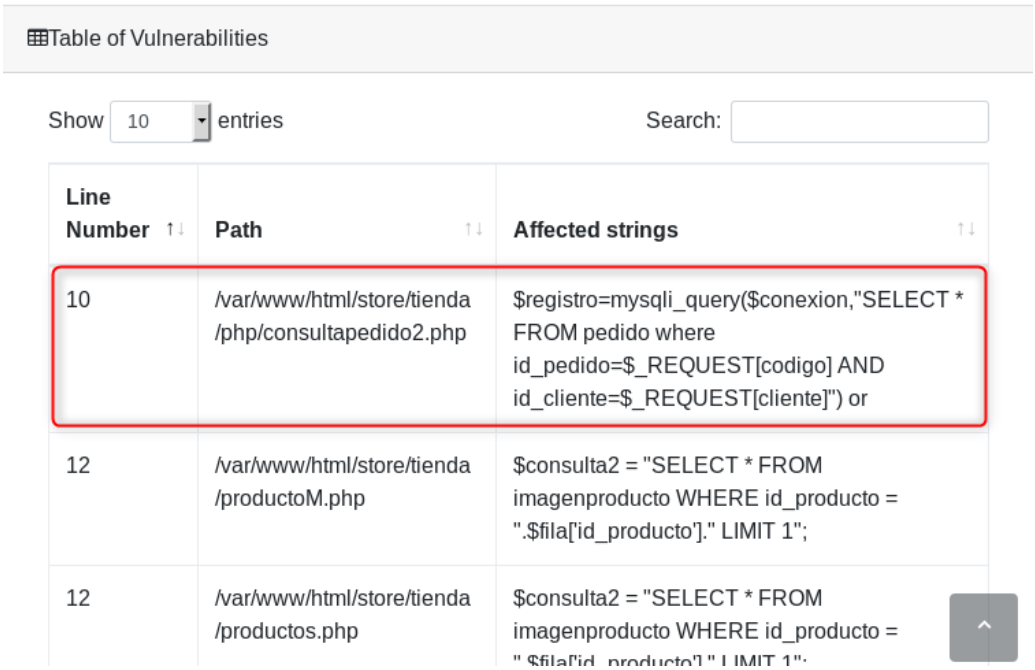


Imagen 3.15. Posibles vulnerabilidades detectadas.

Comprobé la tabla *pedidos* de la base de datos de la aplicación para validar que tiene datos.

```
MariaDB [q]> select * from pedido;
+-----------+------------+---------------------+---------+----------+----------+
| id_pedido | id_cliente | fecha_pedido        | estatus | Subtotal | Total    |
+-----------+------------+---------------------+---------+----------+----------+
|         1 |          1 | 2014-04-04 04:06:13 |       1 |   345.00 |  345.00 |
|         2 |          2 | 2014-04-05 14:22:23 |       2 |   345.00 |  345.00 |
|         3 |          4 | 2014-04-09 18:50:22 |       1 |   470.00 |  470.00 |
|         4 |          5 | 2014-04-09 19:05:00 |       1 |   445.00 |  445.00 |
|         5 |          4 | 2014-04-12 12:55:41 |       1 |   235.00 |  235.00 |
|         6 |          1 | 2014-04-20 19:22:46 |       2 |  1035.00 | 1035.00 |
|         8 |          1 | 2014-05-16 19:35:52 |       0 |   335.00 |  335.00 |
|         9 |          2 | 2014-05-28 18:46:21 |       0 |   660.00 |  660.00 |
|        10 |          2 | 2014-05-28 18:49:18 |       0 |   110.00 |  110.00 |
|        11 |         10 | 2014-05-30 16:34:00 |       2 |   460.00 |  460.00 |
|        12 |         16 | 2016-02-16 14:35:18 |       0 |     0.00 |    0.00 |
|        13 |         18 | 2016-05-31 10:14:02 |       0 |   235.00 |  235.00 |
+-----------+------------+---------------------+---------+----------+----------+
```

Imagen 3.16. Contenido de la tabla pedido.

Puse en ambos campos la inyección SQL con un número entero aleatorio ya que si no encuentra ese entra en la segunda parte de la inyección "*OR*" al poner "*1=1*" aseguramos que siempre se cumpla y así logar la inyección.

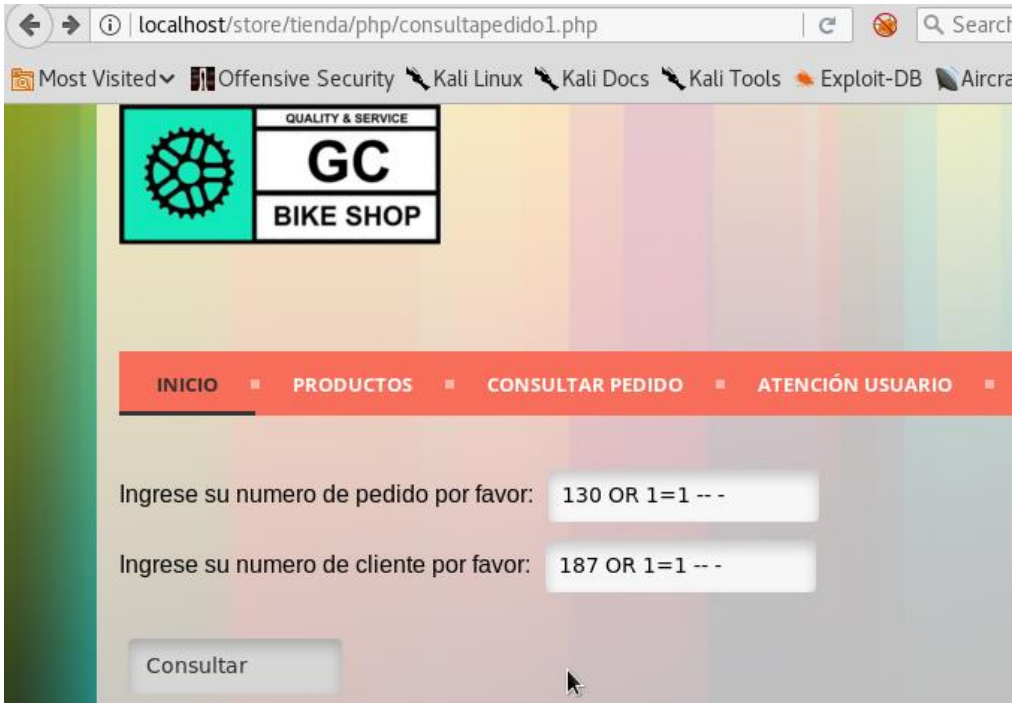


Imagen 3.17. Inyección SQL.

Observé que tiene asociado un id al campo, lo que provoca que la aplicación no reconozca los datos que metí.

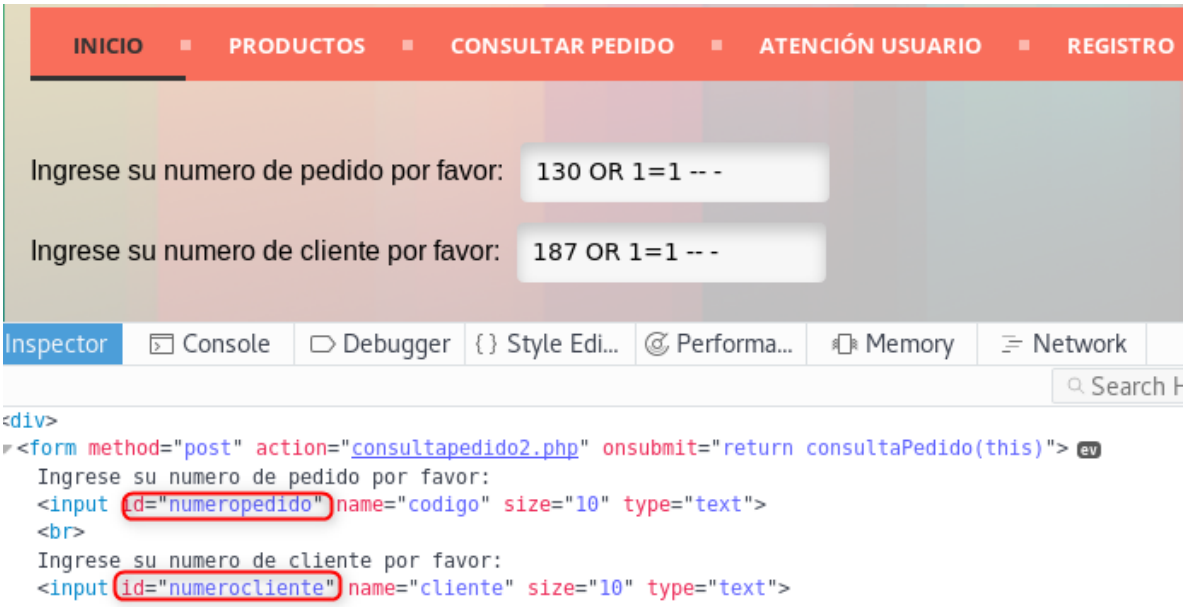


Imagen 3.18. Variable asociada a los campos.

Si no se quita dicho id mostrara el siguiente error.

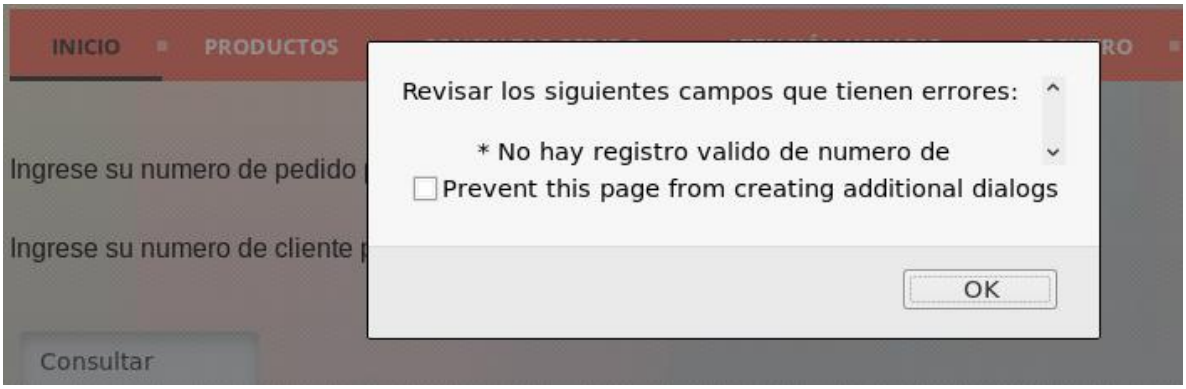


Imagen 3.19. Mensaje de error al introducir la inyección.

Una vez quitando los id se logra la inyección mostrando todos los datos asociados al *idCliente=1* y *IDproducto=1*.

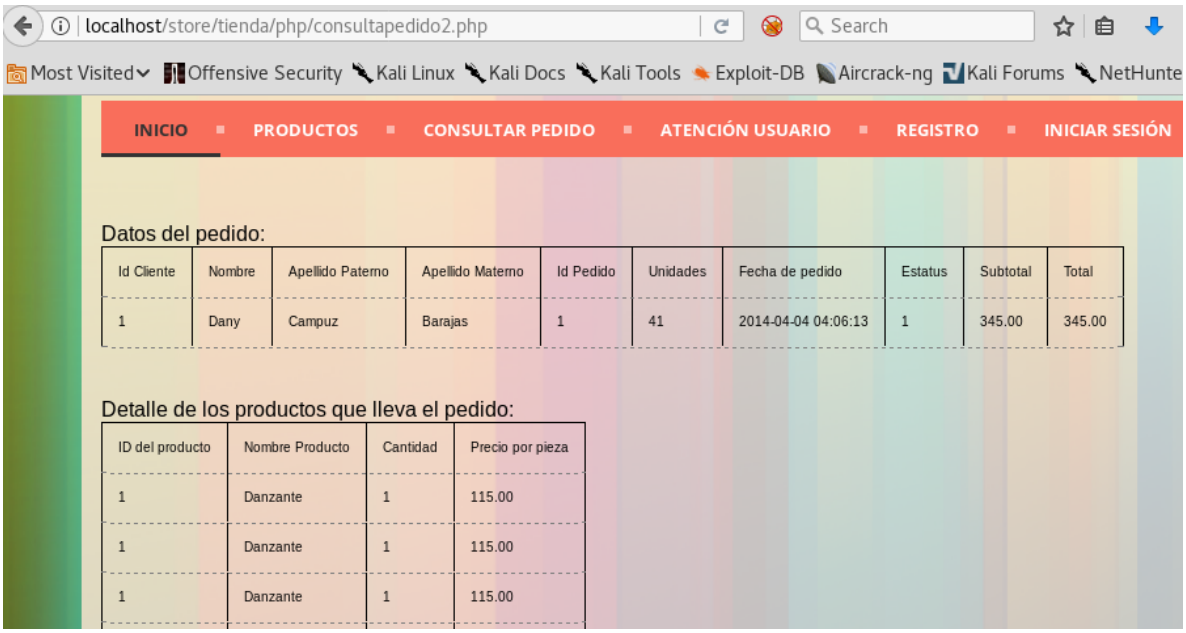


Imagen 3.20. Resultado después de la inyección.

3.2.6.2 Detección XSS

Al correr la herramienta se generó el reporte, el cual encontró 10 posibles vulnerabilidades.

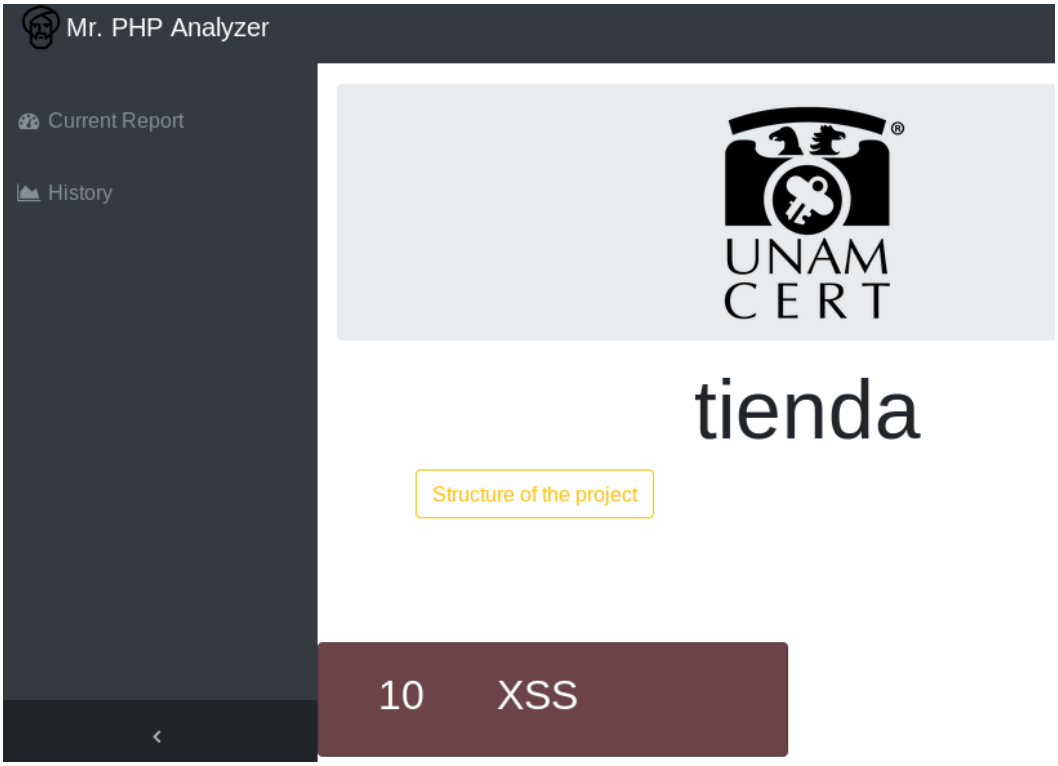


Imagen 3.21. Generación del reporte, XSS.

Probé la primera vulnerabilidad, se encuentra en el archivo *registrarse.php*.

Table of Vulnerabilities

Show 10 entries Search:

Line Number	Path	Affected strings
6	/var/www/html/tienda /registrarse.php	$nombre=($_POST["nombre"]);
7	/var/www/html/tienda /registrarse.php	$Apaterno=($_POST["Apaterno"]);
8	/var/www/html/tienda /registrarse.php	$Amaterno=($_POST["Amaterno"]);
9	/var/www/html/tienda /registrarse.php	$password=($_POST["password"]);

Imagen 3.22. Posibles variables afectadas.

En la siguiente imagen se muestra que no están correctamente sanitizadas[6] las declaraciones de variables.

```
 GNU nano 2.9.5                          registrarse.php

<?php
        session_start();
        include 'conec.php';

        if(isset($_POST["enviar"]) && isset($_POST["nombre"])!="" && isset($_POST["Apaterno"])!="" $
                        $nombre=($_POST["nombre"]);
                        $Apaterno=($_POST["Apaterno"]);
                        $Amaterno=($_POST["Amaterno"]);
                        $password=($_POST["password"]);

                        $sql="INSERT INTO usuario(nombre,Apaterno,Amaterno,password,rol_id)
                                                VALUES('$nombre','$Apaterno','$Amaterno','$password$
                $resultado=pg_query($conexion,$sql);
                ?>
```

Imagen 3.23. Variables mal sanitizadas.

Ingresé un nombre seguido de código JavaScript para el registro del usuario.

[6] En términos de programación, sanitizar se refiere a remover código obsoleto de tal forma para validar todas las entradas de usuario. (Hamila, 2018)

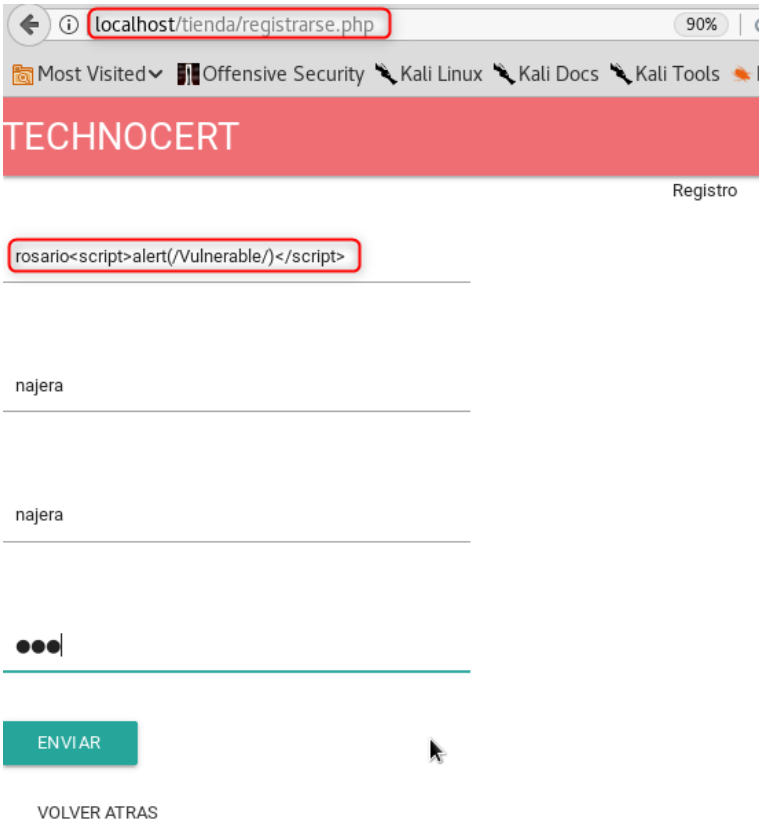


Imagen 3.24. Registro de usuario con código JS en la inserción del nombre.

Se ingresa con el usuario que se creó.

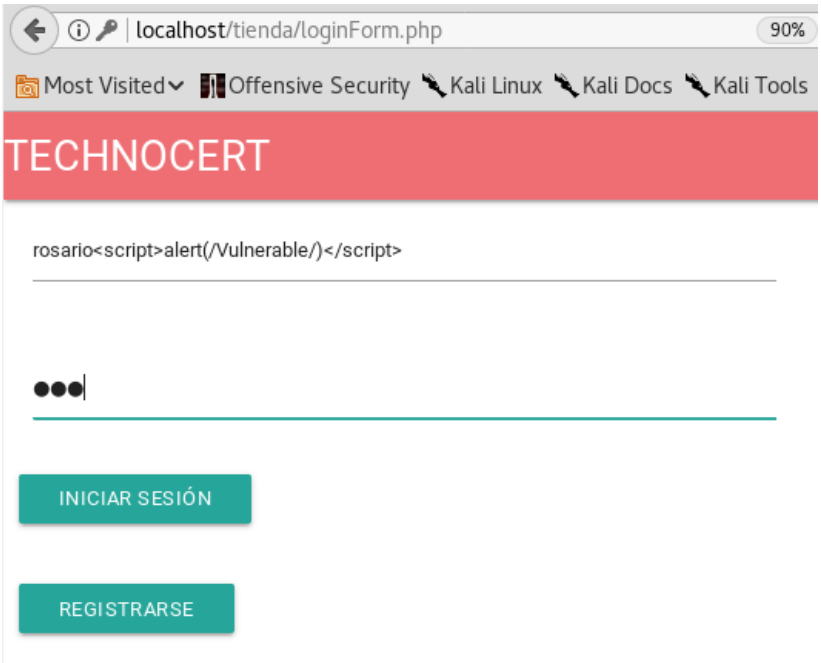


Imagen 3.25. Inicio de sesión con el usuario dado de alta.

Y cada vez que ingrese ese usuario estará apareciendo el siguiente cuadro de texto.

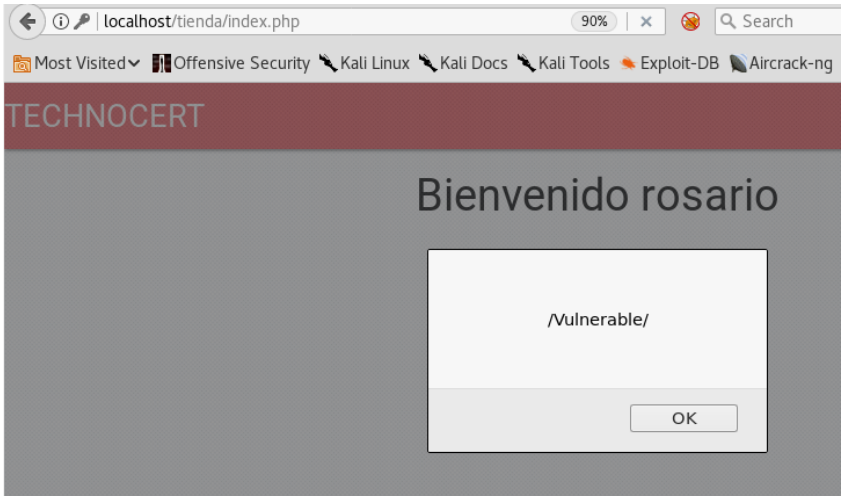


Imagen 3.26. Ejecución del XSS.

3.2.6.3 FUNCIONES OBSOLETAS

Al correr la herramienta se generó el reporte, el cual encontró 10 posibles funciones obsoletas.

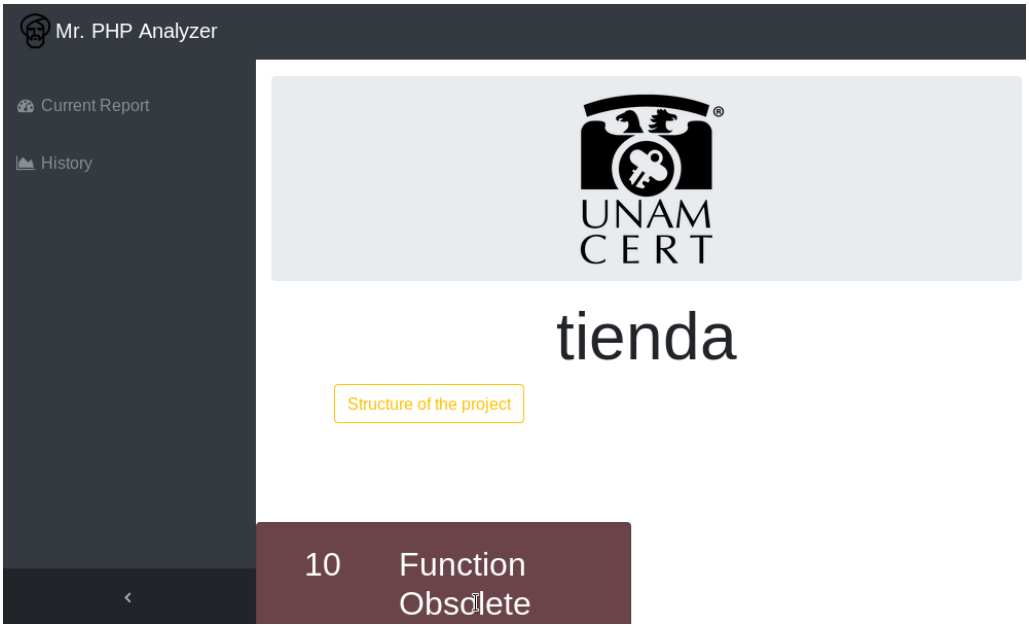


Imagen 3.27. Generación del reporte, funciones obsoletas.

Se muestra la función que está obsoleta y en la última columna la función alternativa por la que debe ser sustituida.

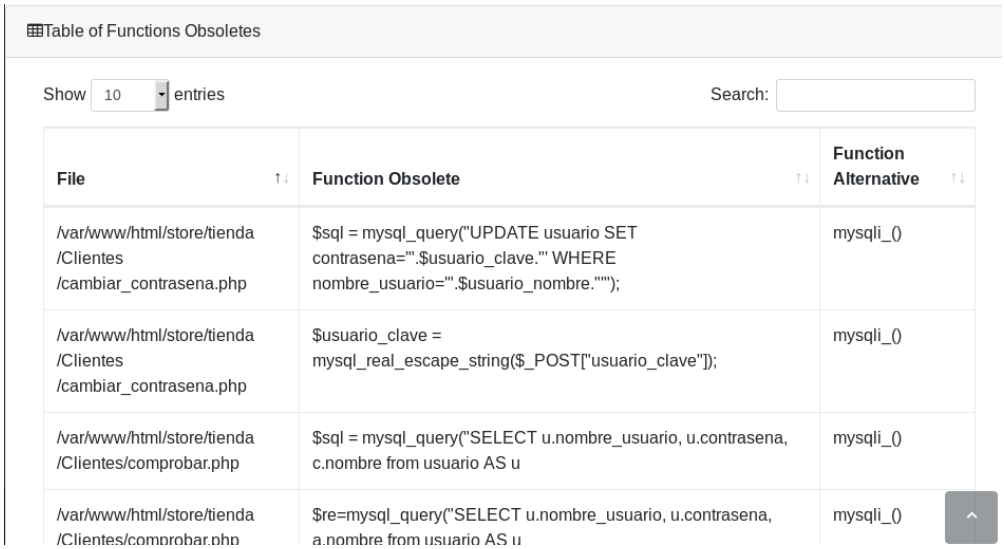


Table of Functions Obsoletes

Show 10 entries Search:

File	Function Obsolete	Function Alternative
/var/www/html/store/tienda /Clientes /cambiar_contrasena.php	$sql = mysql_query("UPDATE usuario SET contrasena='".$usuario_clave."' WHERE nombre_usuario='".$usuario_nombre."'");	mysqli_()
/var/www/html/store/tienda /Clientes /cambiar_contrasena.php	$usuario_clave = mysql_real_escape_string($_POST["usuario_clave"]);	mysqli_()
/var/www/html/store/tienda /Clientes/comprobar.php	$sql = mysql_query("SELECT u.nombre_usuario, u.contrasena, c.nombre from usuario AS u	mysqli_()
/var/www/html/store/tienda /Clientes/comprobar.php	$re=mysql_query("SELECT u.nombre_usuario, u.contrasena, a.nombre from usuario AS u	mysqli_()

Imagen 3.28. Funciones obsoletas con la función alternativa.

3.2.6.4 Detección inyección de comandos

Al correr la herramienta se generó el reporte, el cual encontró 6 posibles inyecciones de código.

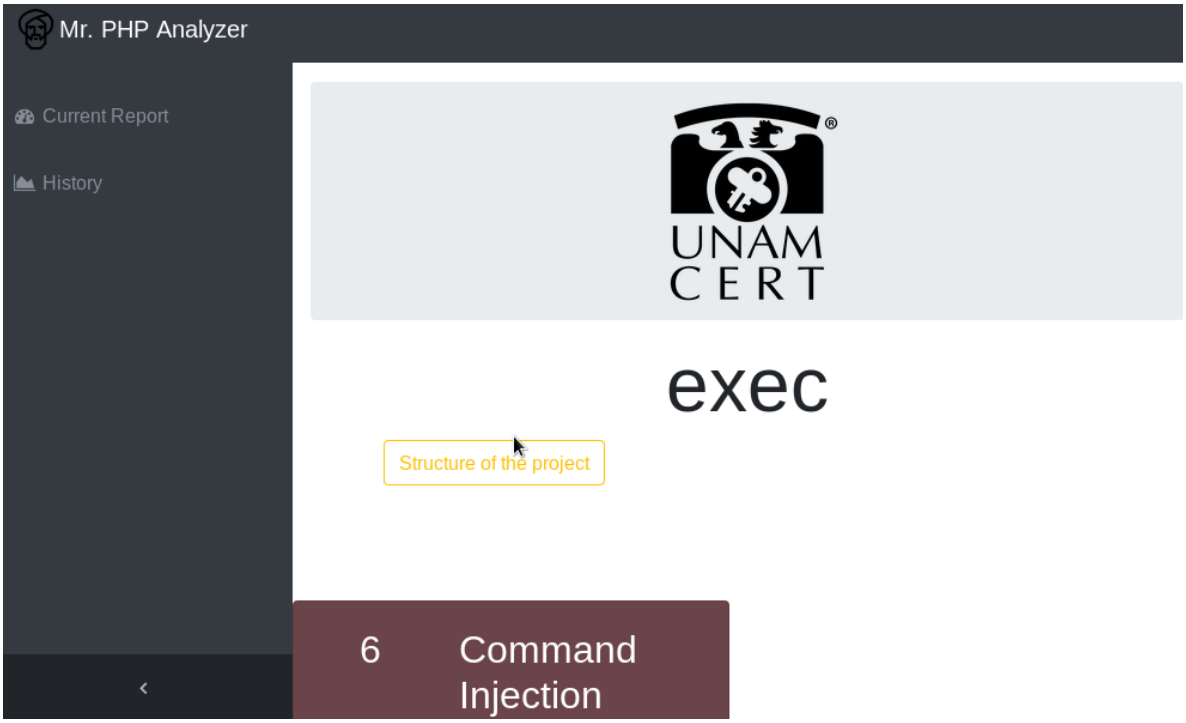


Imagen 3.29. Generación del reporte, Inyección de comandos.

Se muestra la función donde se puede hacer la inyección.

Table of Vulnerabilities

Show 10 entries Search:

Line Number	Path	Affected strings
10	/var/www/html/DVWA-master /vulnerabilities/exec/source /low.php	$cmd = shell_exec('ping ' . $target);
14	/var/www/html/DVWA-master /vulnerabilities/exec/source /low.php	$cmd = shell_exec('ping -c 4 ' . $target);

Imagen 3.30. Posibles variables afectadas.

Se observa que no se está sanitizado la variable *"$cmd"*

```
 GNU nano 2.9.5                                                  low.php

<?php

if( isset( $_POST[ 'Submit' ]  ) ) {
        // Get input
        $target = $_REQUEST[ 'ip' ];

        // Determine OS and execute the ping command.
        if( stristr( php_uname( 's' ), 'Windows NT' ) ) {
                // Windows
                $cmd = shell_exec( 'ping  ' . $target );
        }
        else {
                // *nix
                $cmd = shell_exec( 'ping  -c 4 ' . $target );
        }

        // Feedback for the end user
        $html .= "<pre>{$cmd}</pre>";
}

?>
```

Imagen 3.31. Variables no sanitizadas.

Se comprueba lo que hace la aplicación. Siendo que funciona para hacer ping a algún dispositivo.

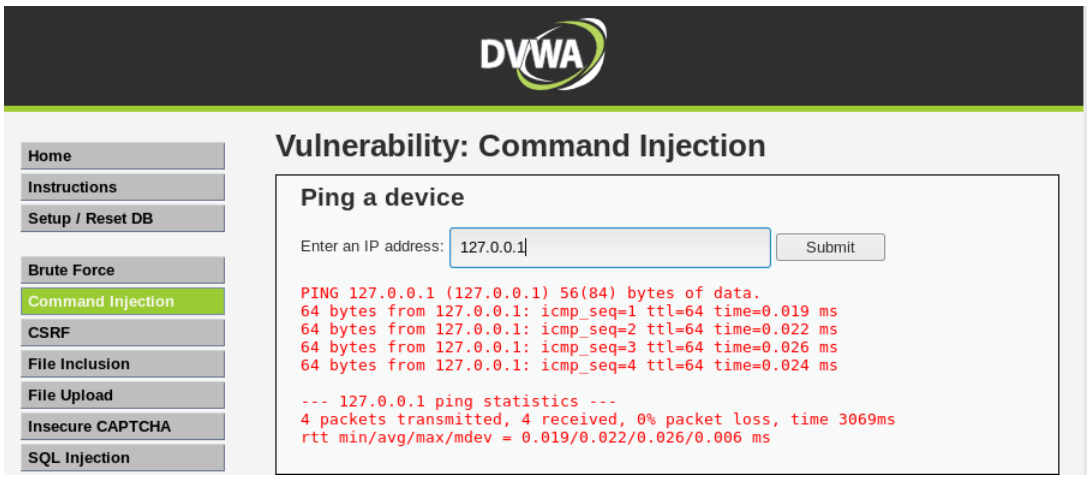


Imagen 3.32. Comprobación de funcionalidad de la aplicación.

Al introducir el siguiente comando, no manda ningún mensaje de error.

Imagen 3.33. Intento de inyección de código.

La aplicación al estar en un ambiente Linux, la concatenación de comandos se hace con ";" (punto y coma). Después de introducir una dirección IP, se muestra el comando a ejecutarse, el cual mostrara el archivo donde se almacenan todos los usuarios del sistema operativo.

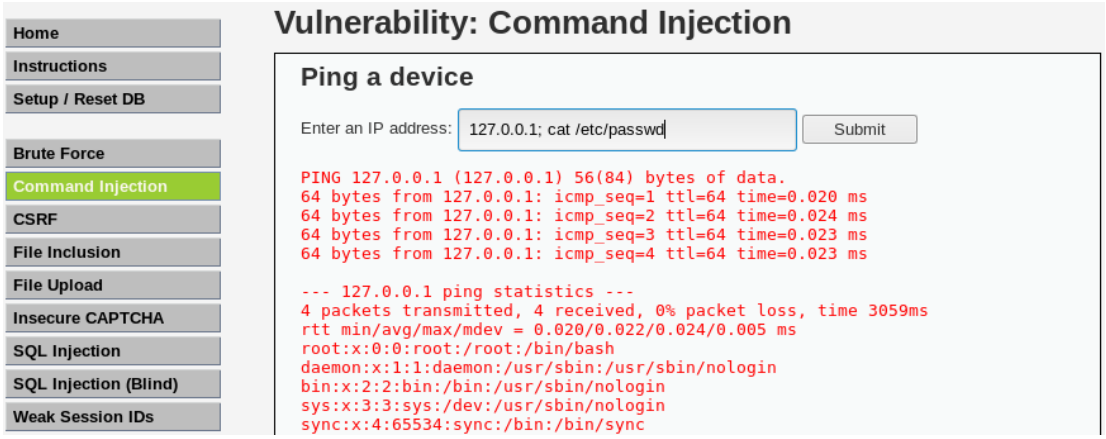


Imagen 3.34. Inyección de código.

3.2.6.5 Detección LFI & RFI

Al correr la herramienta se generó el reporte, el cual encontró 6 posibles LFI.

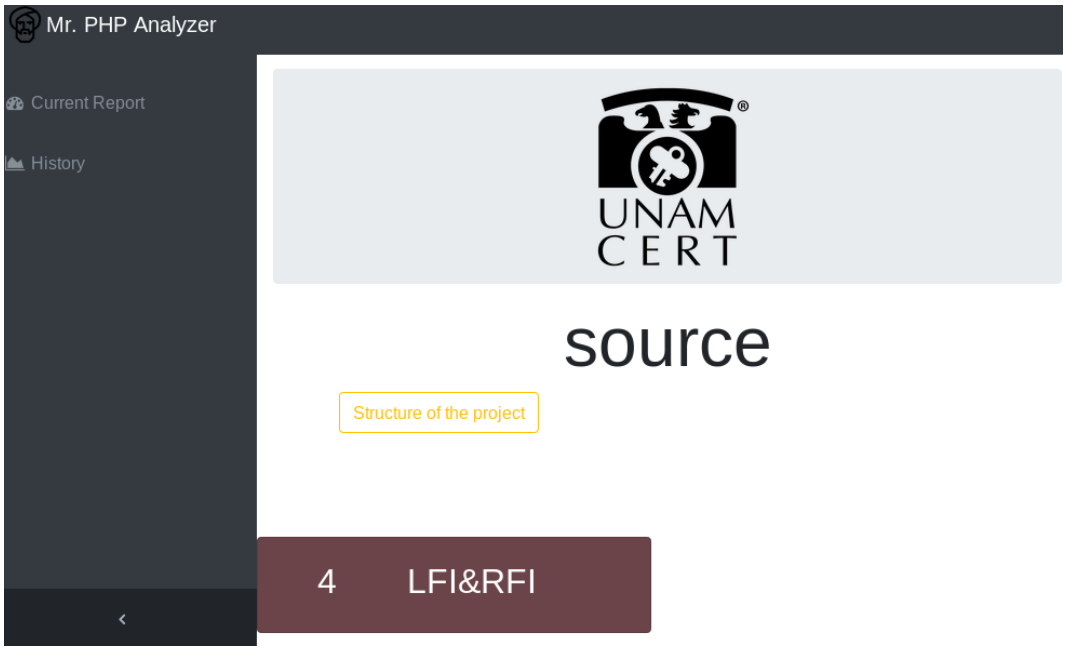


Imagen 3.35. Generación del reporte, LFI

Se muestra las variables que pueden ser explotadas al estar mal sanitizadas.

Table of Vulnerabilities

Show 10 entries Search:

Line Number	Path	Affected strings
4	/var/www/html/DVWA-master /vulnerabilities/fi/source /medium.php	$file = $_GET['page'];
4	/var/www/html/DVWA-master /vulnerabilities/fi/source/low.php	$file = $_GET['page'];

Imagen 3.36. Posibles variables afectadas.

Se verifica en el código que la variable no está sanitizada, por lo que se puede explotar la vulnerabilidad.

```
 GNU nano 2.9.5                  source/low.php

<?php

// The page we wish to display
$file = $_GET[ 'page' ];

?>
```

Imagen 3.37. Variable no sanitizada.

En la URL se observa la variable *page*, en la cual se puede hacer el LFI, consultando el archivo donde se almacenan todos los usuarios.

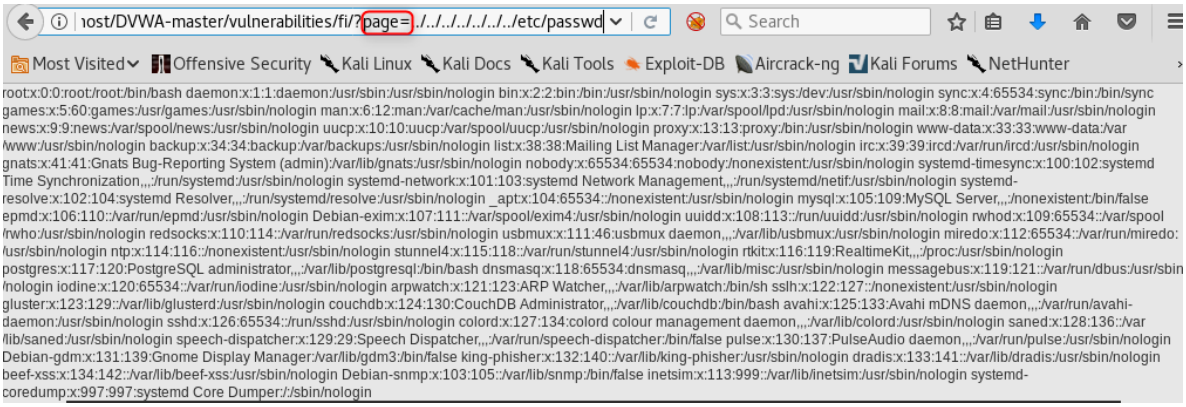


Imagen 3.38. Ejecución del LFI.

3.2.6.6 Detección de generación débil de cookies de sesión

Al correr la herramienta se generó el reporte, el cual encontró 4 posibles generaciones débiles de cookies de sesión.

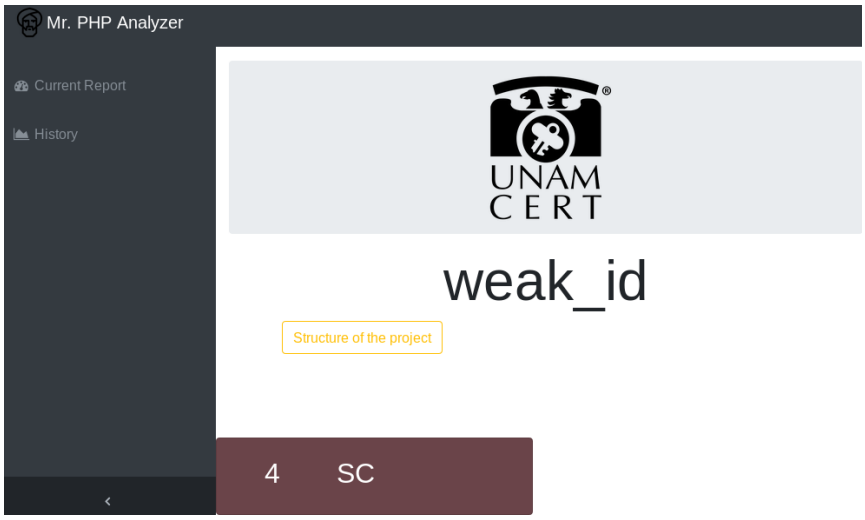


Imagen 3.39. Generación del reporte. Cookies de sesión.

Se observan distintas formas en la que es declarada la generación de la cookie.

Table of Vulnerabilities

Show 10 entries Search:

Line Number	Path	Affected strings
11	/var/www/html/DVWA-master /vulnerabilities/weak_id/source /high.php	setcookie("dvwaSession", $cookie_value, time()+3600, "/vulnerabilities/weak_id/", $_SERVER['HTTP_HOST'], false, false);
7	/var/www/html/DVWA-master /vulnerabilities/weak_id/source /impossible.php	setcookie("dvwaSession", $cookie_value, time()+3600, "/vulnerabilities/weak_id/", $_SERVER['HTTP_HOST'], true, true);
11	/var/www/html/DVWA-master /vulnerabilities/weak_id/source /low.php	setcookie("dvwaSession", $cookie_value);
7	/var/www/html/DVWA-master /vulnerabilities/weak_id/source /medium.php	setcookie("dvwaSession", $cookie_value);

Imagen 3.40. Declaración de las cookies.

En la aplicación al estar dando clic en el botón *Generate*, se estarán generando cookies.

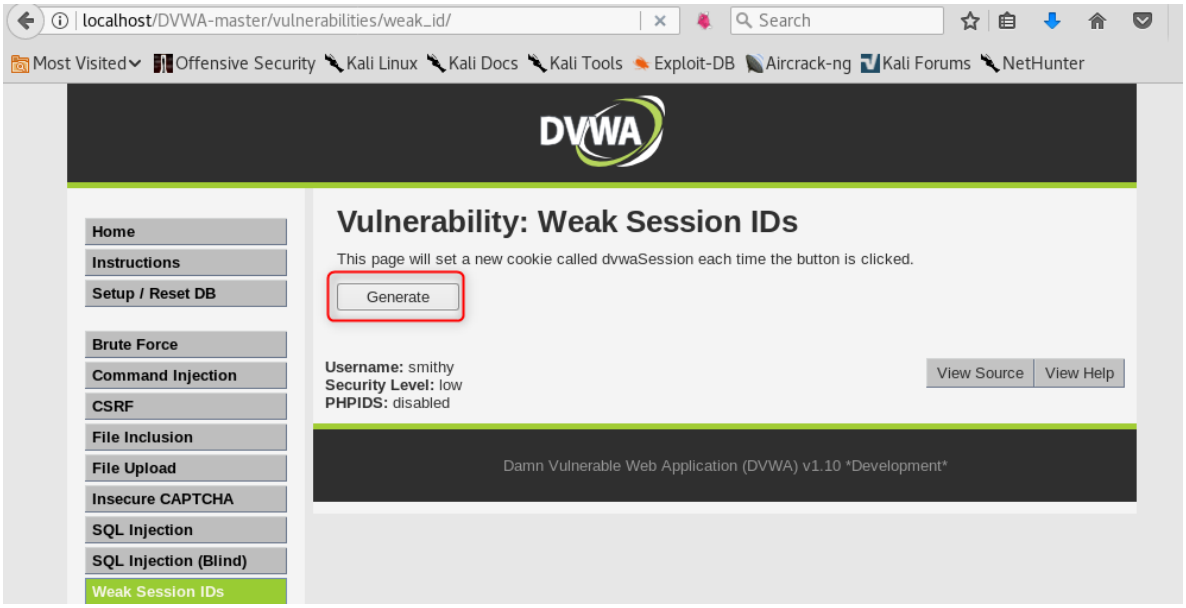


Imagen 3.41. Aplicación para generar cookies.

Se comprueba que la declaración de la cookie es débil al no contener más argumentos que ayuden a robustecer la generación de esta.

Imagen 3.42. Débil declaración de generación de cookies.

Mediante BurpSuite[7] fue posible obtener la petición al dar clic en el botón, observando que la cookie no es lo suficientemente aleatoria.

```
POST /DVWA-master/vulnerabilities/weak_id/ HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost/DVWA-master/vulnerabilities/weak_id/
Cookie: dvwaSession=2; security=low; PHPSESSID=su5knit15ubu9d049cm27as5uo;
all_RyDwsSBXVzZXJzGPa8s-MBDA-clickdesk_referrer=http%3A//localhost/store/tienda/php/consultapedido1.php;
all_RyDwsSBXVzZXJzGPa8s-MBDA-site_visit_time=1532738055794;
all_RyDwsSBXVzZXJzGPa8s-MBDA-visit_count=%7B%22http%3A//*%22%3A4%2C%22website_count%22%3A4%7D
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 0
```

Imagen 3.43. Comprobación de la débil generación de cookies.

3.2.7 Reporte CSV

Como lo mencionaba anteriormente, por cada ejecución se genera un reporte CSV, el cual cuenta con cuatro divisiones, id de la vulnerabilidad, nombre de la vulnerabilidad, archivo de donde se encontró la vulnerabilidad, línea donde se encuentra la vulnerabilidad y si es el caso una alternativa de la función vulnerable o un link para saber cómo tratar esta. En la siguiente imagen muestro un ejemplo de un reporte. Ver Anexo B.

```
ID  NAME                                    FILE                                    LINE                    ALTERNATIVE
2,xss,/var/www/html/DVWA-master/vulnerabilities/fi/source/low.php,$file = $_GET[ 'page' ];,http://wonko.com/post/html-escaping
```

Imagen 3.44. Reporte CSV.

3.2.8 Resultados

El analizador estático de vulnerabilidades de código PHP, ayuda a agilizar la identificación de posibles vulnerabilidades, que estén en el aplicativo, no importando el número de archivos que se tengan, para así, que el desarrollador valide los hallazgos y se corrijan de manera oportuna. A través de los reportes, podrán llevar un histórico de sus análisis para saber qué tipos de vulnerabilidades son las que persisten y en consecuencia tomar acciones para mitigarlas.

Como trabajo a futuro planeo hacer correcciones a las expresiones regulares, tanto de las vulnerabilidades como al código fuente, de tal forma que reduzca los falsos positivos[8], haciendo la detección más confiable.

[7] BurpSuite es una herramienta utilizada principalmente para las auditorías de seguridad en aplicaciones web, una de sus funciones que puede ser utilizada es en forma de proxy.

CONCLUSIONES

Mi estancia en la Facultad de Ingeniería marcó un antes y un después en mi vida, al adquirir una gran diversidad de conocimientos en cuanto a administración de proyectos, redes, sistemas operativos y programación. Estos conocimientos, más los aprendidos en el Plan de Becarios de Seguridad Informática me permiten desempeñar mis actividades laborales, además de que me ayudaron a la realización de los proyectos en los que participé.

No sólo desarrollé habilidades técnicas durante la Facultad, considero que me formó como una persona con ética, en cuanto a las decisiones y actividades que cómo ingeniero desempeñaré, es decir, las repercusiones que pudieran llegar a tener en los códigos de ética en la práctica profesional de la ingeniería.

En el desarrollo de la Auditoría Interna, logré notificar puntos que fueron de ayuda para la CSI/UNAM-CERT, en cuanto al cumplimiento del estándar ISO/IEC 27001:2013 y así poder estar preparados para la auditoría externa.

Este proyecto es el primero que realizo que tenga repercusión a la CSI y por ende a la UNAM, ya que, al realizar dicho proyecto, la CSI mantiene un SGSI mejor implementado y por ende no hay grandes impactos que se puedan ver afectados para la auditoría externa.

Para el proyecto referente al analizador estático de vulnerabilidades de código PHP, cumplí con los objetivos planteados para el proyecto, al lograr la correcta detección de las vulnerabilidades y al generar los reportes en formato CSV y HTML para que dichas vulnerabilidades puedan ser analizadas.

Este proyecto contribuye a todo aquel desarrollador que requiera realizar una revisión de seguridad a su aplicación web, con el objetivo de encontrar posibles vulnerabilidades de forma rápida y hacer las correcciones necesarias.

Parece que un número considerable de desarrolladores no comprende las implicaciones de seguridad que las aplicaciones web requieren, lo que demuestra la falta de capacitación en ciberseguridad. Esta situación crea frustración, que, a veces terminan implementando soluciones totalmente inseguras pero fáciles de implementar. Entre los ejemplos de estas correcciones rápidas incluyen el uso de funciones obsoletas, la desactivación de la protección de falsificación de solicitudes entre sitios, la confianza en todos los certificados en la verificación HTTPS o el uso de protocolos de comunicación obsoletos. Estas malas prácticas de codificación, si se usan en el código de producción puede haber grandes consecuencias como las pruebas que realicé.

---

[8] Falso positivo, sistema o herramienta que genera una falsa alarma. (Gómez, 2018)

El desarrollador irá aprendiendo la importancia de no utilizar funciones obsoletas, sanitizar variables y por ende validar las entradas de usuarios. Asimismo, ayuda a crear una cultura de seguridad a empresas y desarrolladores que servirá para establecer las bases de la protección que esté involucrada en la aplicación web. "Una cultura de la seguridad no se impone, sino que se construye a partir de los compromisos individuales de cada uno de los integrantes de una organización. Y en una cultura de la prevención se combinan: el liderazgo y el apoyo de la alta dirección, el compromiso de los supervisores, y la participación de los trabajadores." (Grupo sancor seguros, 2018)

Ingeniería en Computación es un área en donde el conocimiento va cambiando constantemente, por ende, es de suma importancia el mantenerse actualizado, ya sea por certificaciones, posgrados o en el mismo Plan de Becarios de Seguridad Informática, ya que, como instructor, debo dar información actualizada a los estudiantes. Así que, si creía que ya había terminado de estudiar, estaba pensando mal, tendré que seguir, para mantenerme al día sobre los avances que surgen en cuanto a lo que me rodea.

Referencias electrónicas

(15 de noviembre de 2011). Obtenido de udo: http://udo.mx/sgc/admin/estatico/ISO%2019011-2011.pdf

(10 de junio de 2015). Obtenido de OWASP: https://www.owasp.org/index.php/Path_Traversal

Acunetix. (2016). *Acunetix*. Obtenido de https://d3eaqdewfg2crq.cloudfront.net/resources/acunetix-web-application-vulnerability-report-2016.pdf

Acunetix. (20 de abril de 2017). Obtenido de https://www.acunetix.com/blog/articles/local-file-inclusion-lfi/

Acunetix. (14 de abril de 2017). Obtenido de https://www.acunetix.com/blog/articles/remote-file-inclusion-rfi/

All About Lean. (19 de abril de 2016). Obtenido de https://www.allaboutlean.com/pdca/

Cámara de diputados. (19 de enero de 2018). Obtenido de http://www.diputados.gob.mx/LeyesBiblio/pdf/208_190118.pdf

CIFRHS. (2017). Obtenido de http://cifrhs.salud.gob.mx/2017_enarm/2017_enarm_convocatoria.pdf

Cruz, L. C. (s.f.). Obtenido de http://fcasua.contad.unam.mx/apuntes/interiores/docs/98/6/bases_datos.pdf

CSI/UNAM-CERT. (2017). *adminUNAM 2017*. Recuperado el 13 de 03 de 2018, de https://adminunam.seguridad.unam.mx/

CSI/UNAM-CERT. (2018). *Misión y Visión*. Recuperado el 18 de marzo de 2018, de https://www.cert.unam.mx/mision-y-vision

DGTIC. (30 de noviembre de 2015). Obtenido de https://www.tic.unam.mx/organigrama.html

Guía Scrum. (julio de 2013). Obtenido de https://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-es.pdf

Hamila, A. (30 de enero de 2018). *Medium*. Obtenido de https://medium.com/@abderrahman.hamila/what-sanitize-mean-and-why-sanitize-in-code-data-5c68c9f76164

kanban tool. (2018). Obtenido de https://kanbantool.com/es/metodologia-kanban

OWASP. (12 de junio de 2011). Obtenido de https://www.owasp.org/index.php/Session_Prediction

OWASP. (12 de diciembre de 2013). Obtenido de https://www.owasp.org/index.php/Code_Injection

Owasp. (29 de septiembre de 2017). Obtenido de https://www.owasp.org/index.php/Static_Code_Analysis

OWASP. (05 de 31 de 2018). Obtenido de https://www.owasp.org/index.php/Command_Injection

OWASP. (06 de mayo de 2018). Obtenido de https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

QuincyLarson. (03 de noviembre de 2016). *freeCodeCamp*. Obtenido de https://medium.freecodecamp.org/what-programming-language-should-i-learn-first-%CA%87d%C4%B1%C9%B9%C9%94s%C9%90%CA%8C%C9%90%C9%BE-%C9%B9%C7%9D%CA%8Dsu%C9%90-19a33b0a467d

Scrum Manager. (23 de marzo de 2014). Obtenido de https://www.scrummanager.net/bok/index.php?title=Epic

shodan. (2018). Obtenido de https://www.shodan.io/

TechTarget. (abril de 2012). *SearchSecurity*. Obtenido de https://searchsecurity.techtarget.com/definition/role-based-access-control-RBAC

Temprado, M. O. (30 de septiembre de 2015). *Kaleido consultoría*. Obtenido de https://www.kaleidoconsultoria.com/

UNAM. (2018). Recuperado el 22 de marzo de 2018, de DGTIC: http://www.tic.unam.mx/pdfs/AcuerdoTIC.pdf

W3SCHOOLS. (2018). Obtenido de https://www.w3schools.com/sql/sql_injection.asp

ANEXO A. SCRUM

Anexo que contiene el desarrollo de las actividades dentro de la metodología Scrum.

Épica

Desarrollar una herramienta que realice análisis estático de código fuente en aplicaciones desarrolladas en PHP, buscando funciones inseguras y malas prácticas de programación. Como resultado del análisis, mostrar en un reporte en formato HTML los hallazgos detectados, con la finalidad de mostrar ante directivos de una forma clara y visual lo que la aplicación carece.

Historias de usuario

1. Como desarrollador de aplicaciones en el lenguaje de programación PHP requiero analizar de manera estática, aplicaciones para detectar vulnerabilidades.
2. Como desarrollador de aplicaciones en el lenguaje de programación PHP requiero un reporte técnico en HTML de las vulnerabilidades detectadas.
3. Como desarrollador de aplicaciones en el lenguaje de programación PHP requiero instalación sencilla de la herramienta y fácil uso.
4. Como desarrollador de aplicaciones en el lenguaje de programación PHP requiero sea multiplataforma para hacerlo compatible y se implemente en diferentes desarrollos.
5. Como desarrollador de aplicaciones en el lenguaje de programación PHP requiero que analice de manera rápida la aplicación.
6. Como desarrollador de aplicaciones en el lenguaje de programación PHP requiero poder analizar aplicaciones con grandes cantidades de línea de código.

Lista de tareas

PRIMER SEMANA

PRIMER DÍA

1. Investigar el funcionamiento de las vulnerabilidades para PHP. (4 h)
2. Investigar las funciones obsoletas para PHP (versión mínima 5). (4 h)

SEGUNDO DÍA

3. Realizar expresiones regulares para la detección de las vulnerabilidades basadas en SQLi para PHP. (4 h)

4. Realizar expresiones regulares para la detección de las vulnerabilidades basadas en LFI para PHP. (4 h)

TERCER DÍA

5. Realizar expresiones regulares para la detección de las vulnerabilidades basadas en RFI para PHP. (4 h)
6. Realizar expresiones regulares para la detección de las vulnerabilidades basadas en Path Traversal para PHP. (4 h)

CUARTO DÍA

7. Realizar expresiones regulares para la detección de las vulnerabilidades basadas en envío de información sensible de manera insegura para PHP. (4 h)
8. Realizar expresiones regulares para la detección de las vulnerabilidades basadas en manejo inadecuado de cookies de sesión para PHP. (4 h)

QUINTO DÍA

9. Realizar expresiones regulares para la detección de las vulnerabilidades basadas en inyección de comandos para PHP. (4 h)
10. Realizar expresiones regulares para la detección de las vulnerabilidades basadas en inyección de código fuente para PHP. (4 h)

SEGUNDA SEMANA

SEXTO DÍA

11. Realizar expresiones regulares para la detección de funciones obsoletas en PHP. (5 h)
12. Diseñar la base de datos donde se almacenarán los reportes y las líneas donde se detectó la vulnerabilidad. (1 h)
13. Programar el código para la creación de la base de datos. (2 h)

SÉPTIMO DÍA

14. Programar el módulo para la detección de funciones obsoletas. (4 h)
15. Programar el módulo para la detección de LFI Y RFI. (4 h)

OCTAVO DÍA

16. Programar el módulo para la detección de SQLi. (4 h)
17. Programar el módulo para la detección de path traversal. (4 h)

NOVENO DÍA

18. Programar el módulo para la detección de Cross Site Scripting. (4 h)
19. Programar el módulo para la detección de inyección de comandos y de código. (4 h)

DÉCIMO DÍA

20. Programar el módulo para la detección de envío de información sensible de manera insegura y manejo inadecuado de cookies de sesión (4 h)
21. Desarrollar el módulo para la generación de reportes CSV. (4 h)

DÉCIMO PRIMER DÍA

22. Desarrollar el módulo para la generación de reportes web de las vulnerabilidades detectadas. (8 h)

DÉCIMO SEGUNDO DÍA

23. Realizar pruebas de funcionalidad con la aplicación de tienda al módulo de carrito de compras. (2 h)
24. Realizar pruebas de funcionalidad con la aplicación de tienda al módulo de inicio de sesión. (2 h)
25. Realizar pruebas de funcionalidad con la aplicación de gestión de inscripciones programado en PHP. (2 h)
26. Realizar pruebas de funcionalidad con la aplicación de nómina programado en PHP. (2 h)

DÉCIMO TERCER DÍA

27. Generar la documentación necesaria para el uso de la herramienta. (3 h)

ANEXO B. REPORTE CSV

Anexo que contiene un ejemplo de la generación de un reporte en formato CSV.

ID	NAME	FILE	LINE	ALTERNATIVE
2	xss	/var/www/html/DVWA-master/vulnerabilities/fi/source/high.php	$file = $_GET['page'];	http://wonko.com/post/html-escaping
5	lfi&rfi	/var/www/html/DVWA-master /vulnerabilities/ fi/source/impossible.php	$file = $_GET['page'];	https://www.netsparker.com/blog/web-security/remote-file-inclusion-vulnerability/
2	xss	/var/www/html/DVWA-master/ vulnerabilities/ fi/source/low.php	$file = $_GET['page'];	http://wonko.com/post/html-escaping
5	lfi&rfi	/var/www/html/DVWA-master/ vulnerabilities/ fi/source/medium.php	$file = $_GET['page'];	https://www.netsparker.com/blog/web-security/remote-file-inclusion-vulnerability/

Tabla B.1. Reporte CSV.

ANEXO C. DICCIONARIO DE LA BASE DE DATOS

Anexo que contiene el contenido de cada tabla de la base de datos.

Tabla: functions		
Descripción: tabla que contiene las expresiones regulares para detectar las funciones obsoletas		
Campo	Tipo	Descripción
function_id	INTEGER	id de la función
function	TEXT	nombre de la función vulnerable
expression	TEXT	expresión vulnerable
alternative	TEXT	expresión a sustituir
description	TEXT	información de la función obsoleta
versionob	INTEGER	versión obsoleta

Tabla C.1. Funciones obsoletas.

Tabla: names_vul		
Descripción:tabla que contiene el nombre de las vulnerabilidades.		
Campo	Tipo	Descripción
name_vul_id	INTEGER	id del nombre de la función
name	TEXT	nombre de la vulnerabilidad
description	TEXT	Descripción de la vulnerabilidad

Tabla C.2. Nombre vulnerabilidades.

Tabla: recommendations		
Descripción: tabla que contiene enlaces que serán de ayuda para consultar sobre cómo mitigar las vulnerabilidades.		
Campo	Tipo	Descripción
recommendation_id	INTEGER	id de la recomendación
liga	TEXT	enlace de la recomendación
description	TEXT	enlace ligado a la vulnerabilidad

Tabla C.3. Recomendaciones.

Tabla: reports		
Descripción: tabla que contiene el nombre del proyecto analizado, así como la fecha de creación del reporte		
Campo	Tipo	Descripción
report_id	INTEGER	id de la función
name	TEXT	nombre de la aplicación web
date_hour	DATETIME	hora de ejecución
version	TEXT	versión PHP

Tabla C.4. Contenido de los reportes.

Tabla: reports_vul_func		
Descripción: tabla que contiene la línea vulnerable, archivo, id de la vulnerabilidad e id del reporte.		
Campo	Tipo	Descripción
re_vul_func_id	INTEGER	id re_vul_func
line	TEXT	línea vulnerable
file	TEXT	archivo vulnerable
functionob	INTEGER	id de la función obsoleta
report	INTEGER	id del reporte
vulnerability	INTEGER	id de la vulnereabilidad

Tabla C.5. Concentración del reporte y vulnerabilidades.

Tabla: versions		
Descripción: tabla que contiene las expresiones regulares para detectar la versión de PHP.		
Campo	Tipo	Descripción
version_id	INTEGER	id de la función
version	TEXT	versión de PHP
expression	TEXT	función de PHP
description	TEXT	descripción de la función

Tabla C.6. Detección de versiones de PHP.

Tabla: vulnerabilities		
Descripción: tabla que contiene las expresiones regulares para detectar las vulnerabilidades.		
Campo	Tipo	Descripción
vulnerability_id	INTEGER	id de la función
name	INTEGER	id nombre de vulnerabilidad
expression	TEXT	expresión vulnerable
description	TEXT	descripción de la vulnerabilidad
recommendation	INTEGER	id del enlace

Tabla C.7. Vulnerabilidades.

GLOSARIO

Término	Definición
Auditoria	Proceso sistemático, independiente y documentado para obtener evidencias de la auditoría y evaluarlas de manera objetiva con el fin de determinar la extensión en que se cumplen los criterios de auditoría.
CSI	Coordinación de Seguridad de la Información
CERT	Computer Emergency Response Team/Equipo de Respuesta a Incidentes
UNAM	Universidad Nacional Autónoma de México
DGTIC	Dirección General de Cómputo y de Tecnologías de Información y Comunicación
Épica	Historia de usuario que por su gran tamaño, el equipo descompone en historias con un tamaño más adecuado para ser gestionada con los principios y técnicas ágiles: estimación y seguimiento cercano
Falso positivo	Sistema o herramienta que genera una falsa alarma
Historia de usuario	Descripción de una funcionalidad que debe incorporar un sistema de software, y cuya implementación aporta valor al cliente.
ISMS/SGSI	Information Security Management System/Sistema de Gestión de Seguridad de la Información.
Incidente	Serie de eventos de seguridad de la información indeseados o inesperados, que tienen una probabilidad significativa de comprometer las operaciones de negocio y de amenazar la seguridad de la información.
ISO 27001	Especifica los requerimientos para establecer, implementar y mantener ISMS.

ISO 19011	Directrices para la auditoria de sistemas de gestión.
Scrum	Metodología de Desarrollo Ágil.
Sprint	Lapso, durante el cual se crea un "Hecho", utilizable y como parte del proceso empírico. Producto potencialmente liberable.
XSS	Este tipo de vulnerabilidad abarca cualquier ataque que permita ejecutar código de "scripting", como VBScript o javascript, en el contexto de otro dominio. Estos errores se pueden encontrar en cualquier aplicación HTML, no se limita a sitios web, ya que puede haber aplicaciones locales vulnerables a XSS, o incluso el navegador en sí.
SQLi	Inserción o "inyección" de una consulta SQL a través de los datos de entrada del cliente a la aplicación.
Generación insegura de cookies de sesión	Se centra en la predicción de valores de ID de sesión que permiten a un atacante eludir el esquema de autenticación de una aplicación. Al analizar y comprender el proceso de generación de ID de sesión, un atacante puede predecir un valor de ID de sesión válido y obtener acceso a la aplicación.
Local File Inclusion	Es el proceso de incluir archivos, que ya están presentes localmente en el servidor, a través de la explotación de los procedimientos de inclusión vulnerables implementados en la aplicación.
Remote File Inclusion	Es el proceso de incluir archivos remotos a través de la explotación de los procedimientos de inclusión vulnerables implementados en la aplicación.
Path Traversal	Consiste en acceder a los archivos y directorios que se almacenan fuera de la carpeta raíz web. Al manipular variables que hacen referencia a archivos con secuencias "punto-punto-barra (../)" y sus variaciones o mediante el uso de rutas de archivos absolutas.
Product Owner	Es quien toma las decisiones del cliente.
Inyección de comandos	Se refiere a la capacidad de un usuario, que controla la entrada de comandos (bien a través de un terminal de Unix/Linux o del

	interfaz de comando de Windows), para ejecutar instrucciones que puedan comprometer la integridad del sistema.
Inyección de código fuente	Está debida a la inclusión de la función include() la cual permite el enlace de archivos situados en otros servidores, mediante los cuales se puede ejecutar código PHP en el servidor. Se utilizan las funciones include, include_once, require, require_once las cuales son utilizadas para incluir en una página web otras páginas por tanto el atacante podrá obtener una Shell en el servidor de la víctima y ejecutar un archivo. Para que se pueda ejecutar dicho archivo debe tener una extensión diferente a ".php" ya que con esta extensión, el archivo se ejecutaría en el servidor del atacante y no en el de la víctima, así un archivo ".txt", ".gif", etc. Serían algunos de los más adecuados.
Shell	Interfaz con un sistema operativo, se pueden dar órdenes y mandar datos para que el sistema realice las tareas que se especifican.
CSI	Coordinación de Seguridad de la Información
UNAM	Universidad Nacional Autónoma de México
DGTIC	Dirección General de Cómputo y de Tecnologías de Información y Comunicación