Estado del arte (mejorar, muy extenso, concentrarse en las soluciones, no el problema)

~~En la actualidad las aplicaciones web se volvieron muy populares facilitando el acceso a información sin importar el lugar donde se encuentre, pero así como trae facilidades, también aparecen nuevos inconvenientes~~.

Muchas veces confiamos que las aplicaciones web donde depositamos nuestra información es confiable y que nuestros datos están seguros, pero no siempre es así, según una encuesta de Oracle [3] podemos observar que la mayoría de las bases de datos de las organizaciones carecen de seguridad suficiente, y por el número de ataques a sitios web dejando expuesta sensible a manos de los atacantes, se está aumentando la presión por mejorar la seguridad de las bases de datos [Bamrara 2015]

~~Por otro lado,~~ los aplicativos web también tienen sus inconvenientes. El proyecto abierto de seguridad en aplicaciones Web (OWASP por sus siglas en inglés) ~~es una comunidad abierta dedicada a facultar a las organizaciones a desarrollar, adquirir y mantener aplicaciones que pueden ser confiables~~ [2]. Dentro de sus proyectos podemos encontrar un TOP 10 de riesgos más comunes en aplicaciones web, entre los que se encuentran:

A1: Inyección~~: Las fallas de inyección, tales como SQL, OS, y LDAP, ocurren cuando datos no confiables son enviados a un intérprete como parte de un comando o consulta. Los datos hostiles del atacante pueden engañar al interprete en ejecutar comandos no intencionados o acceder datos no autorizados [OWASP, 2013]. Estos ataques son ejecutados del lado del servidor. La principal forma de inyección son las inyecciones SQL, también conocidas como SQLI y son muy fáciles de explotar a que el atacante envía ataques con cadenas simples de texto a través de una fuente no confiable, como puede ser un formulario web, los cuales explotan la sintaxis del intérprete a vulnerar. Casi cualquier fuente de datos puede ser un vector de inyección. Las entradas en pasan a formar una consulta dinámica que resulta en explotar una falla de seguridad. Las técnicas se utilizan para:~~

~~-obtener acceso a datos sin autorización~~

~~-modificar datos llevando a perdida de información o integridad~~

~~-cambiar autorizaciones a datos almacenados~~

~~La forma más simple es agregando comillas simples en el campo de login de usuario, siguiendo el ejemplo de [Couture, 2013], en el formulario de inicio de sesión, deberíamos escribir nuestro nombre de usuario y contraseña, para con eso crear una consulta SQL como SELECT id FROM usuarios WHERE nombreusuario = ‘Pablo’ AND password = ‘1234’, pero modificando esas entradas, podemos escribir como nombre de usuario ‘OR 1=1 --/, entonces a la hora de consultar nuestra base de datos, nuestra query pasaría a ser SELECT id FROM usuarios WHERE nombreusuario = ‘‘OR 1=1 --/ AND password=’’, en donde el motor de bases de datos tomaría todo lo siguiente a --/ como comentario, ignorándolo, dejando la consulta SELECT id FROM usuarios WHERE nombreusuario = ‘‘OR 1=1, a lo que va a devolver con todos los nombres de usuarios que sean null o devolver el nombre de usuario en caso de que 1=1, cosa es que siempre es verdadero, retornando los sombres de todos los usuarios. [Gupta, 2015].~~

A3 – Cross Site Scripting (XSS) ~~o también conocido como secuencia de comando de sitios cruzados es una de los riesgos más usuales de las aplicaciones web, se mantiene entre las primeras 5 amenazas de OWASP desde el 2005~~ y hoy en día ocupa el tercer puesto en probabilidad de vulnerabilidad de aplicaciones web, con un 47% en Whitehat Security Status Report 2015. Los ataques XSS engañan al sitio web para que responda a un código ejecutable inyectado por el atacante, el cual se ejecuta en el navegador del cliente.

~~Estos ataques utilizan códigos HTML o JavaScript, que utilizan como anfitrión al sitio infectado, y el ataque se ejecuta del lado del cliente, por lo que se vuelve difícil detectarlo, permitiendo el robo de Id de usuarios, cookies, passwords, números de tarjetas de créditos, control remoto del navegador entre otros [Gupta &Gupta 2015].~~

Los ataques XSS se dividen principalmente en dos tipos:

XSS almacenados~~: ocurren cuando un usuario ingresa a un sitio donde se inyecto un código malicioso, el código se crea y se lo dispone en el sitio en alguno de los campos, (como comentarios en un blog, o de un formulario web) sin pasar por filtros de seguridad, el código se almacena, y se vuelve a cargar con cada visita. En otras palabras, el código queda en el sitio.~~

~~Los~~ XSS indirectos o reflejados ~~ocurren en las aplicaciones web donde las entradas no confiables, son incluidas inmediatamente en la salida, y reflejadas del servidor web directamente en el navegador. Ocurren principalmente en mensajes de error, peticiones a motores de búsqueda pre visualización de comentarios. Requieren que la víctima acceda a una URL creada especialmente por el atacante y en ella estará el código malicioso, cuando la víctima acceda al enlace, el navegador ejecutara el código~~ [Brady 2015]. ~~Un ejemplo de sitios vulnerables son los que tienen una página dinámica para mostrar errores, y envían el mensaje de error como un parámetro Ej.:~~ [~~http://mdsec.net/error/5/Error.ashx?message=Sorry%2c+an+error+occurred~~](http://mdsec.net/error/5/Error.ashx?message=Sorry%2c+an+error+occurred), si no se aplica ningún tipo de filtro o saneo de entradas, el sitio será vulnerable a este tipo de ataques. [Stuttard, Pinto 2011]

~~Podemos concluir de que estas~~ La vulnerabilidad se deben principalmente a errores de programación, muchas veces por no seguir las buenas prácticas y otras por ahorrar tiempo de programación o testeo [1], que se pueden evitar en su gran medida con validación de campos y parametrización de consultas SQL [Couture 2013]. Otra medida de combatir las inyecciones SQL es usar un modelo de privilegios de usuarios de la base de datos propuesto en [7], lo que también beneficia a la seguridad general de la base de datos.

Los mecanismos de defensa básicos de las aplicaciones web son [8]:

-Controlar el acceso a la información de la aplicación y prevenir el acceso no autorizado a la misma

-Controlar las entradas a las funciones de la aplicación para prevenir que estas causen un comportamiento indebido en la misma

-Controlar ataques para asegurarse que la aplicación siga funcionando correctamente después de resultar objetivo, pueda aplicar medidas defensivas u ofensivas

-Habilitar una sección para el administrador, para que pueda monitorear actividades y configurar funcionalidades.

Siendo el control de acceso la primera línea de defensa para nuestra aplicación, pero vamos a hacer un enfoque principalmente en el control de entradas, siendo este uno de los puntos más vulnerables de las aplicaciones web.

Para verificar que nuestra aplicación es segura se pueden usar herramientas como sqlmap y arachni [9] para detectar vulnerabilidades de SQLI y XSS respectivamente, ambos escáner gratuitos open source, como también Web Goat, una aplicación de OWASP para detectar y explotar vulnerabilidades, donde también podemos probar la inyección de scripts codificados [4], lo cual se usa mucho hoy en día como forma de evadir los controles regulares [5].

Objetivo general:

~~Diseñar y desarrollar un sistema web multiusuario aplicando técnicas de seguridad de base de datos e inyección de código malicioso~~

Estudiar las técnicas de Seguridad Web y aplicarlas en el desarrollo de un sistema Web para el INECDE

Objetivos particulares:

(reducir, mas generales)

-Aplicar medidas de protección de bases de datos contra inyecciones SQL

-aplicar medidas de frontend y backend para proteger la integridad de la base de datos

-aplicar medidas para mitigar cross site scripting

-aplicar configuraciones del servidor web para evitar posibles brechas de seguridad

-aplicar modelo de privilegios de base de datos para evitar acciones indebidas

-aplicar medidas de forensia para poder analizar corregir posibles problemas de errores humanos en la base de datos

-Implementar un framework para aumentar la seguridad

- Diseñar y desarrollar una base de datos compartida y realizar la migración desde planillas Excel

Fundamentación(explicar mas el porque de la aplicacion)

La idea central es realizar una aplicación web multiusuario con orientación web para el INICNE (instituto de ictiología), que actualmente utiliza planillas Excel individuales como medio de resguardo de la información tanto para su archivado como para su análisis. Para su desarrollo se hará uso de metodologías agiles, más específicamente una variante de FDD [10], el cual en la carrera no se lo lleva a la práctica.

Con una aplicación web multiusuario se crearían nuevas posibilidades como compartir información entre los distintos investigadores del instituto, la posibilidad de poder diferenciar datos de distintas campañas, realizar búsquedas más específicas, obtener datos de forma ordenada y la capacidad de generar informes.

Junto a estas ventajas también aparecen inconvenientes, la seguridad de los datos se vuelve más vulnerable ante el incremento de usuarios y la disposición de estos mismos en la web.

Hoy en día la mayoría de las personas disponen mucha información, muchas veces personal, sensible o confidencial en distintas aplicaciones de internet sin ser consciente de los distintos riesgos que esto conlleva. Los distintos ataques informáticos contra aplicaciones basadas en web ocurren prácticamente todos los días, haciendo uso de vulnerabilidades o fallas de seguridad de las aplicaciones, muchas veces por uso de software desactualizado, falta de testeo o mala programación [1].

En esta tesis se propone investigar y aplicar medidas defensivas y preventivas para proteger los datos y la aplicación, empezando por la protección de la aplicación contra accesos indebidos, siguiendo con la administración de la base de datos para mitigar posibles intentos de operaciones indebidas finalmente realizando acciones para mitigar intentos de inyección maliciosa de código, tanto inyecciones SQL como cross site scripting mediante el uso de un framework llamado Codeigniter para asegurar la estructura de las aplicaciones web, y hacer uso de sus beneficios frente al uso de PHP puro, como se implementa en la carrera.

METODOLOGIA(a revisar, puede ser xp o incremental)

La metodología a usar será una variante de FDD o desarrollo dirigido por características [10], a pesar de no ser un proyecto muy grande, va a ser conveniente porque es uno de los que mejor se adapta a un sistema donde la seguridad es un factor crítico.

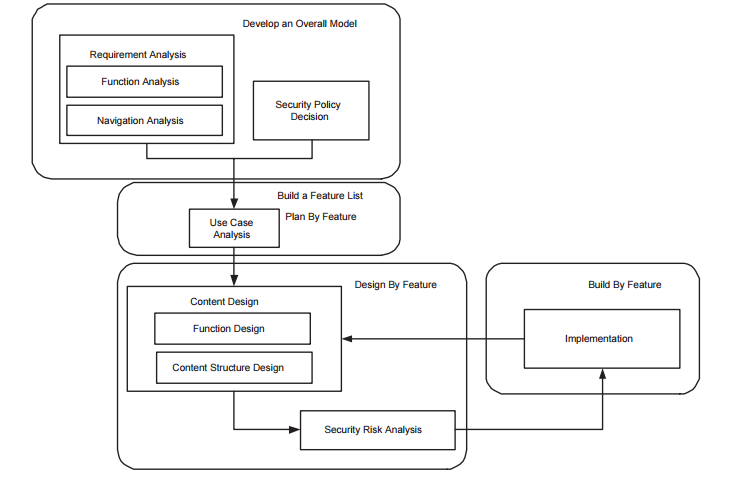
Define 5 procesos:

– Desarrollar el modelo global: en esta etapa se busca determinar las necesidades y expectativas de los clientes, se van a captar los requerimientos y se creara una política de seguridad

– Construir una lista de características: en esta etapa se usaran diagramas de caso de uso para representar las distintas características a implementar

– Planificar

– Diseñar por característica: en la fase de diseño más importante, por un lado se consideran las implementaciones de las características, se crea un diseño estructural, donde se especifican organizan los contenidos relacionados a cada característica, y el diseño funcional, donde se especifica como el usuario va a interactuar con las características. Por otro lado se hace un análisis de riesgos de seguridad, el cual se evalúa con cada iteración, es decir, por cada característica agregada, se vuelve a analizar los riesgos de seguridad

– Construir por característica. En esta etapa se transforma el modelo concreto de una característica en una reléase, que es una versión ejecutable, que cumple con las políticas de seguridad 

Ámbito de trabajo:

El trabajo se desarrollara principalmente de forma independiente, acudiendo con los futuros usuarios de la aplicación para obtener información e ir comprobando las versiones de la aplicación, y acudiendo; y acudiendo a la FaCENA para consultas al profesor orientador

Recursos:

Como recursos informáticos hará uso de Ubuntu como sistema operativo host, en el cual se programara con HTML, PHP y JavaScript con el uso del framework Codeigniter ejecutados sobre un servidor Apache y abasteciéndose de datos de Mysql. Para poder realizar esto se hará uso de las documentaciones de los lenguajes frameworks, como también uso de bibliografía adicional para profundizar más sobre la temática de la seguridad [8 y 6].

Desarrollos propuestos:

Se propone diseñar y desarrollar una aplicación web que sea capaz de almacenar y explotar grandes volúmenes de datos muestrales, para lo cual se configurara un servidor Apache, no solo para que pueda hospedar la aplicación sino para que pueda actuar como una medida más de seguridad, evitando accesos indebidos y registrando distintos tipos de sucesos. Por otra parte se deberá diseñar e implementar una base de datos teniendo en cuenta aspectos de seguridad como los distintos tipos de usuarios, y los permisos que deberá tener cada uno. La misma se deberá crear en base a datos tabulados en planillas Excel, y luego migrar los mismos hacia la nueva base de datos.

Una vez desarrollada la aplicación, se propone probar que tan segura es contra el tipo de inyecciones mencionado con herramientas como SQLmap y Web Goat

Resultados esperados:

Se espera poder aprender sobre los distintos riesgos de seguridad de aplicaciones web, centrado en las inyecciones SQL y cross site scripting, ya que para poder defendernos debemos conocer cuáles son los posibles ataques.

También espera diseñar una aplicación que pueda servir no solo para los objetivos propuestos, sino también como una guía para futuros alumnos que quieran profundizar más en la programación web, ya sea con el uso de Frameworks para php, como también en realizar aplicaciones más seguras.

Referencias:

[1] L. Salgado, M. Ron, F. Solis “ANÁLISIS DE RIESGOS DE LAS APLICACIONES WEB DE LA SUPERINTENDENCIA DE BANCOS Y SEGUROS, UTILIZANDO LAS RECOMENDACIONES TOP TEN DE OWASP.” Universidad de las Fuerzas Armadas ESPE. Mar-2014

[2]OWASP top 10 https://www.owasp.org/images/5/5f/OWASP\_Top\_10\_-\_2013\_Final\_-\_Espa%C3%B1ol.pdf

[3] Joseph McKendrick oracle” Dbs security superhero” 2014 IOUG ENTERPRISE DATA SECURITY SURVEY October 2014

[4]Hansen, R. XSS cheat sheet for filter evasion. Retrieved from https://www.owasp.org/index.php/XSS\_Filter\_Evasion\_ Cheat\_Sheet

[5] B. B. Guptaa\*, S. Guptaa, S. Gangwara, M. Kumara & P. K. Meenaa” Cross-Site Scripting (XSS) Abuse and Defense: Exploitation on Several Testing Bed Environments and Its Defense” Journal of Information Privacy and Security, Volume 11, Issue 2, 2015, Special Issue: Secured Communication in Wireless and Wired Networks

[6]P. Brady”Survive the Deep End: PHP Security”

[7]S. Amirmohammad, M. Zamani, and A.A. Manaf. “SQL Injection is Still Alive: A Study on SQL Injection Signature Evasion Techniques," In Informatics and Creative Multimedia (ICICM), 2013 International Conference, pp. 265-268. IEEE, 2013.

[8] Stuttard, Dafydd and Pinto, Marcus, the Web Application Hacker's Handbook: Finding and Exploiting Security Flaws, 2011

[9] Erik Couture “Web Application Injection Vulnerabilities a Web App’s Security Nemesis?” SANS Institute. Mayo 2013

[10] X. Ge, R. F. Paige, F. A.C. Polack, H. Chivers, P. J. Brooke ”Agile development of secure web applications” ICWE '06 Proceedings of the 6th international conference on Web engineering pp 305-312, New York, NY, USA ©2006

[11]B. Gupta “Web Application Security – What You Need to Know” THIRTY-THIRD ANNUAL PACIFIC NORTHWEST SOFTWARE QUALITY CONFERENCE. World Trade Center Portland Portland, Oregon. October 2015

[12]Open Web Application Security Project (OWASP). OWASP Web Goat project. Bel Air, MD: OWASP. Retrieved June 2, 2015 from https://www.owasp.org/index.php/Category:OWASP\_WebGoat\_Project

[13]L. Conklin, G. Robinson OWASP “code review guide” extraído <https://www.owasp.org/images/7/78/OWASP_AlphaRelease_CodeReviewGuide2.0.pdf>

[14]  S. Fogie, J.Grossman, R. Hansen, A. Rager, P. D. Petkov  “XSS Attacks: Cross Site Scripting Exploits and Defense” Syngress; 1 edicion. (May 15, 2007)

[15] A. Bamrara” Evaluating Database Security and Cyber Attacks: A Relational Approach” Journal of Internet Banking and Commerce, August 2015, vol. 20, no. 2

[16] en Whitehat Security Status Report 2015