

Universidad Nacional Del Nordeste

Facultad de Ciencias Exactas y Naturales y Agrimensura

Licenciatura en Sistemas de Información.

Proyecto Final de Carrera

“Titulo”

Autor

Silva, Pablo Ariel

pabloariel01@hotmail.com

Profesores Orientadores

Lic. Bosco, Juan Francisco

boscojfrancisco@hotmail.com

Expt. Cuzziol, Juan Jose

jcuzziol@hotmail.com

Corrientes – Argentina – Año 2016

Capítulo 1.

Introducción.

Breve estado del arte.

Hoy en día si queremos desarrollar una aplicación web tenemos, necesariamente, que pensar en la web 2.0 [1] donde diseño centrado en el usuario, colaboración e interoperabilidad son las bases, y los sitios antiguos que solo muestran información dejan de ser populares y son reemplazados por sitios donde los usuarios pasan a tomar un papel central, pueden interactuar entre ellos y con las aplicaciones generando nueva información y se cambia el uso de links para mostrar distintos contenidos por sitios dinámicos donde se puede mostrar distintos contenidos sin cargar otra página, permitiendo crear aplicaciones similares a las de escritorio sobre los mismos [2].

El avance de la tecnología, también ha permitido que la capacidad de conectarnos crezca exponencialmente, pasamos de usar computadoras para trabajar y conectarnos a internet, a tener múltiples dispositivos conectados y poder elegir entre ver información, en una computadora, Tablet o teléfono celular, con lo que pasamos de tener que utilizar solo la computadora, por diseñar una aplicación para computadoras y otra para móviles, a llegar a lo que hoy se conoce como responsive design, poder diseñar una sola aplicación, que se adapte a los distintos dispositivos.

Para ser posible la llegada de sitios dinámicos se emplean distintos tipos de lenguajes, por un lado se continuo usando HTML [3], un lenguaje de etiquetado que nos permiten dar una estructura básica y un código para la definición de contenido de nuestro sitio, y se lo complemento con lenguajes de Script [4], como JavaScript, el cual posee una gran variedad de librerías adicionales entre las cuales podemos resaltar a Jquery y Angular, que se pueden añadir para incorporar nuevas funcionalidades, con hojas de estilo en cascada o también conocido como CSS [5] que cumplen la función de organizar la presentación y el aspecto del sitio.

Este conjunto de tecnologías funcionando de forma coordinada para evitar los tediosos tiempos de recarga, permitiendo crear aplicaciones interactivas y fáciles de usar se conoce como AJAX (Asynchronous JavaScript and XML) [6], el cual en sus principios no tuvo mucha popularidad, pero con el tiempo eso cambio, transformándolo en un requisito fundamental para que una aplicación tenga éxito. Para Incorporar estas tecnologías a nuestra aplicación satisfactoriamente, debemos realizar modificaciones a la estructura de la misma y pasar de utilizar solamente la arquitectura cliente servidor a incluir patrones de diseño, como el patrón MVC [2] el cual permite desarrollar una aplicación web de una forma ordenada y con un código modularizado y legible.

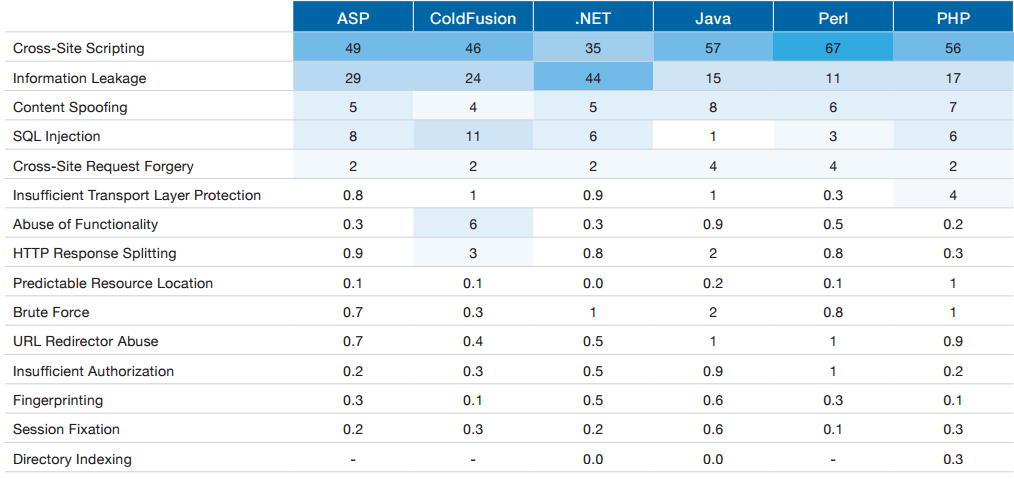
Además hay que tener en cuenta nuevas consideraciones de seguridad, el mismo dinamismo que aporta funcionalidades a las aplicaciones también agrega nuevos puntos de falla que pueden servir para fines maliciosos.

Muchas veces se confía que las aplicaciones web donde se deposita nuestra información es confiable y que los datos están seguros, aunque según una encuesta de Oracle [7] se puede observar que la mayoría de las bases de datos de las organizaciones carecen de seguridad suficiente. Pero esta confianza está empezando a bajar debido al 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 [8].

Actualmente hay un proyecto llamado OWASP [9] que realizo un TOP 10 de los principales riegos de las aplicaciones web, entre los que se encuentran las secuencias de comandos de sitios cruzados como unos de los más explotados.

Las secuencia de comando de sitios cruzados o mejor conocido como Cross Site Scripting (XSS) [10] [11] [12]es uno de los riesgos más usuales de las aplicaciones web, se mantiene entre las primeras 5 amenazas de OWASP desde el 2005 y según Whitehat [13] ocupa el tercer puesto en probabilidad de vulnerabilidad de aplicaciones web, con un 47% y es la principal vulnerabilidad en aplicaciones web programadas en PHP. 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.

Fig. 1 porcentaje de vulenerabilidad por lenguaje[whitehat2014]

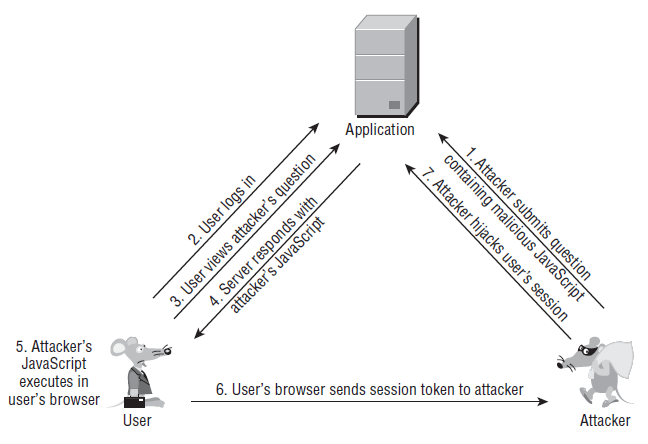


Estos ataques son basados códigos HTML o JavaScript, que utilizan como anfitrión al sitio infectado, y se ejecuta del lado del cliente, por lo que se vuelve difícil detectarlo, permitiendo el robo de Id de usuarios, cookies, contraseñas, números de tarjetas de créditos, control remoto del navegador entre otros [14].

Los ataques XSS se dividen en tres tipos [10]:

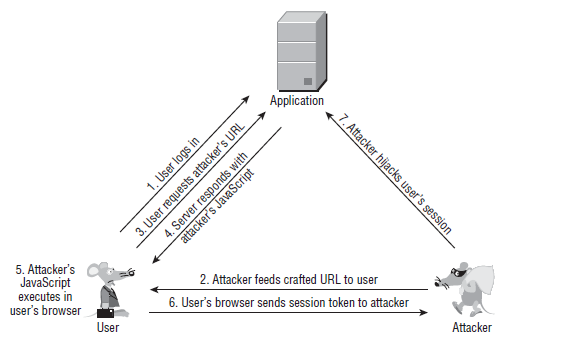
XSS almacenados o persistentes: ocurren cuando un usuario ingresa a un sitio donde se inyecto un código malicioso, y se lo dispone en el sitio en alguno de los campos, (como comentarios en un blog, o de un formulario web), el código se almacena, y se carga con cada visita.

Fig. 2 XSS almacenados [11]



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 o pre visualización de comentarios. Requieren que la víctima acceda a una URL creada especialmente por el atacante y en ella estará el código malicioso [11].

Fig. 3 XSS reflejados [11]

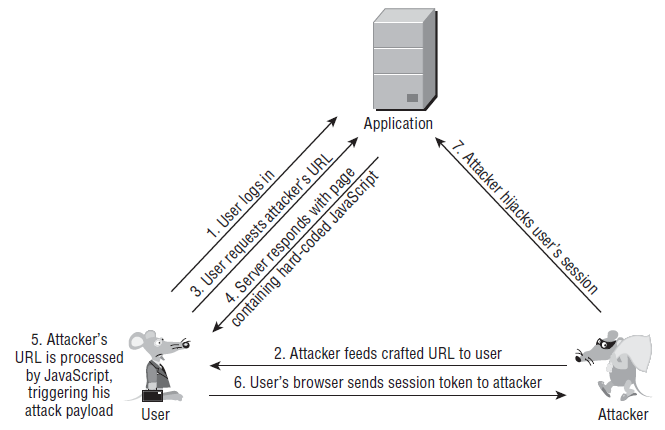


Basadas en el DOM de la aplicación [12], son explotables en sitios donde buscando dinamismo se modifica el DOM del sitio desde la URL. La porción de la URL antes de “&” se utiliza para indicar al navegador a que parte del sitio dirigirse, y la porción adelante del signo queda en el DOM. La diferencia con los XSS reflejados es que estas no son enviadas al navegador, sino que quedan en el DOM. Ej.:

http://victim/promo?product\_id=100&title=Foo#<SCRIPT>alert

('XSS%20Testing')</SCRIPT>

Fig. 4 XSS basados en DOM [11]



Actualmente la mayoría de los navegadores proveen filtros básicos para evitar ataques XSS, pero para ser posible que estén activos todo el tiempo, son muy limitados y ninguno asegura que el sitio sea seguro, por lo que no se debería confiar únicamente que el filtro del navegador web proteja a la aplicación de todos los ataques [15].

En este proyecto se aplican técnicas para mitigar los ataques XSS [10] [14] [16] [11] [12], se controlaran las entradas a las funciones de la aplicación [17] [11] [16], siendo estas una de las vulnerabilidades más comunes en los aplicativos web.

Para realizar el control existen diferentes técnicas, como el uso de librerías, proxy de aplicación, la validación de entradas y el escape de caracteres considerados peligrosos, lo que puede ser hecho del lado del cliente o del servidor, entre otras [16] [15].

Objetivo.

Objetivo general:

Estudiar las técnicas de Seguridad Web orientadas a la prevención de inyecciones de código malicioso y aplicarlas en el desarrollo de un sistema Web para el INICNE.

Objetivos específicos:

* Aplicar medidas de protección contra Cross Site scripting.
* Profundizar el estudio de conceptos de seguridad web.
* Implementar un framework para aumentar la seguridad, crear una aplicación estructurada, y con un código legible y mantenible.
* Diseñar y desarrollar una base de datos.

Fundamentación.

Actualmente el INICNE(Instituto de Ictiología del Nordeste) trabaja con datos de muestreos recolectados de distintas campañas en diferentes localidades, los cuales son anotados en distintas planillas en papel al momento de ser recolectados y una vez finalizada la campaña uno de los investigadores se encarga de transcribirlos a una planilla de Excel la cual esta indexada por un identificador asignado secuencialmente a cada uno de los peces muestreados y posee un alto índice de redundancia de datos, la cual es copiada al final de una planilla histórica, y otra planilla con datos de otras campañas, para luego ser enviados a distintos documentos Excel para ser tratados, formateados y utilizados en tablas dinámicas para generar informes, un trabajo que se realiza en una semana.

Este trabajo podría reducirse utilizando una base de datos relacional, para almacenar la información de una forma más eficiente, tener los datos en un solo lugar y realizar consultas SQL para recuperar los datos deseados, con el formato requerido de una forma más rápida. Como las personas que utilizan los datos desconocen dicho lenguaje y, para garantizar usabilidad y una mayor eficiencia se desarrollo una aplicación web que explote la base de datos, permita mostrar toda la información y generar los informes de una forma más sencilla. La una carga de datos se puede mejorar, quitando los datos duplicados, e introduciendo únicamente los datos necesarios.

Aun cuando los usuarios potenciales de la aplicación carecen conocimientos informáticos, es una aplicación web, y como tal debemos prever de que otros usuarios atenten los aspectos de seguridad.

Según OWASP [9] los riesgos más comunes de las aplicaciones web son las inyecciones SQL [10] y XSS, por lo que el uso de un framework y de librerías especializadas permiten la prevención de estas amenazas. Por otro lado se utiliza un sistema de control de usuarios para garantizar que solo los usuarios deseados puedan acceder a la aplicación y a la vez poder restringir el acceso de determinados sitios de la aplicación.

Capítulo 2.

Metodología.

Capítulo 3.

Herramientas y/o lenguajes de programación.

Capítulo 4.

Resultados.

Capítulo 5.

Conclusiones y futuros trabajos

# Referencias

|  |  |
| --- | --- |
| [1] | . T. O'Reilly, « What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software.,» *Communications & Strategies,* nº 65, pp. 17-37, 2007. |
| [2] | A. MacCaw, JavaScript Web Applications, North, Sebastopol, CA: O'Reilly Media, 2011. |
| [3] | W3C, «W3C: HTML & CSS,» [En línea]. Available: https://www.w3.org/standards/webdesign/htmlcss. |
| [4] | D. Hazaël-Massieux, «W3C,» [En línea]. Available: https://www.w3.org/standards/webdesign/script. |
| [5] | H. W. Lie y B. Bos, «W3C,» 11 4 2008. [En línea]. Available: https://www.w3.org/TR/2008/REC-CSS1-20080411/. |
| [6] | A. T. Holdener III, Ajax: The Definitive Guide, North, Sebastopol, CA: O'Reilly Media, 2008. |
| [7] | J. McKendrick, «Dbs. Security Superhero: IOUG Enterprise Data Security Survey,» Unisphere Research, New Jersey, 2014. |
| [8] | A. Bamrara, «Evaluating Database Security and Cyber Attacks: A Relational Approach,» *Journal of Internet Banking and Commerce,* vol. 20, nº 2, 2015. |
| [9] | OWASP, «OWASP Top Ten Project,» 12 junio 2013. [En línea]. Available: www.owasp.org/images/5/5f/OWASP\_Top\_10\_-\_2013\_Final\_-\_Espa%C3%B1ol.pdf. |
| [10] | E. Couture, «Web Application Injection Vulnerabilities a Web App’s Security Nemesis?,» GIAC (GWAPT) Gold Certification, SANS Institute, Mayo 2013. |
| [11] | D. Stuttard y M. Pinto, The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws, Indianapolis, Indiana: John Wiley & Sons, Inc., 2011. |
| [12] | S. Forgie, J. Grossman, R. Hansen, A. Rager y P. D. Petkov, XSS Attacks: Cross Site Scripting Exploits and Defense., Burlington, MA: Syngress Publishing, Inc., 2007. |
| [13] | WhiteHat Security, « WhiteHat Security Status Report,» 2015. [En línea]. Available: https://info.whitehatsec.com/rs/whitehatsecurity/images/2015-Stats-Report.pdf. |
| [14] | B. B. Guptaa, K. P. Meenaa, S. Guptaa, S. Gangwara y M. Kumara, «Cross-Site Scripting (XSS) Abuse and Defense: Exploitation on Several Testing Bed Environments and Its Defense,» *Journal of Information Privacy and Security,* vol. 11, nº 2, pp. 118-136, 2015. |
| [15] | M. Heiderich, E. A. Vela Nava, G. Heyes y D. Lindsa, Web Application Obfuscation ‘-/WAFs..Evasion..Filters//alert(/Obfuscation/)-’, Burlington, MA: Elsevier, 2011. |
| [16] | P. Brady, «Survive the Deep End: PHP Security,» 26 Octubre 2015. [En línea]. Available: http://phpsecurity.readthedocs.org/en/latest/. |
| [17] | B. Gupta, «Web Application Security – What You Need to Know,» de *Thirty-Third Annual Pacific Northwest Software Quality Conference*, World Trade Center Portland Portland, Oregon. pp 271-280, Octubre 2015. |

Anexos.