Review / Artículo de revisión / Revisão - Tipo 3

Antidefacement - State of art

Christian Camilo Urcuqui López / ccurcuqui@icesi.edu.co

Melisa García Peña / melisa.garcia@correo.icesi.edu.co

José Luis Osorio Quintero / jose.osorio1@correo.icesi.edu.co

Andrés Navarro Cadavid / anavarro@icesi.edu.co

Universidad Icesi, Cali-Colombia

ABSTRACT Internet connects around three billions of users worldwide, a number increasing every day. Thanks to this technology, people, companies and devices perform several tasks, such as information broadcasting through websites. Because of the large volumes of sensitive information and the lack of security in the websites, the number of attacks on these applications has been increasing significantly. Attacks on websites have different purposes, one of these is the introduction of unauthorized modifications (defacement). Defacement is an issue which involves impacts on both, system users and company image, thus, the researchers community has been working on solutions to reduce security risks. This paper presents an introduction to the state of the art about techniques, methodologies and solutions proposed by both, the researchers community and the computer security industry.

KEYWORDS Defacement; Web application; security; vulnerability; Web security; integrity.

Antidefacement - Estado del arte

RESUMEN Internet conecta alrededor de tres mil millones de usuarios en todo el mundo, número que sigue en aumento con el transcurrir de los días. Gracias a esta tecnología, las personas, las compañías y los dispositivos pueden realizar distintas tareas, como por ejemplo difundir información a través de los sitios Web. Debido a los grandes volúmenes de información sensible y al desconocimiento de la seguridad en las páginas Web, el número de ataques a estas aplicaciones ha aumentado significativamente. Los ataques a las páginas web pueden tener distintas finalidades, una de estas es la introducción de modificaciones no autorizadas (defacement), un problema que puede afectar, tanto a los usuarios del sistema, como a la imagen de una empresa. Debido a ello, la comunidad de investigadores viene trabajando en soluciones que permitan reducir los riesgos de seguridad. Este trabajo aporta a ese esfuerzo en la medida en que presenta una introducción al estado del arte de las técnicas, metodologías y soluciones propuestas, tanto por la comunidad de investigadores, como por la industria de seguridad informática.

PALABRAS CLAVE Defacement; aplicación Web; seguridad; vulnerabilidad; seguridad en la Web; integridad.

Antidefacement - Estado da arte

RESUMO Internet conecta cerca de três mil milhões de usuários em todo o mundo, número que continua em aumento com o transcorrer dos dias. Graças a esta tecnologia, as pessoas, as empresas e os dispositivos podem realizar várias tarefas, como por exemplo, difundir informação através dos websites. Devido aos grandes volumes de informação sensível e ao desconhecimento do da segurança nas páginas Web, o número de ataques a estas aplicações tem aumentado significativamente. Os ataques às páginas web podem ter finalidades diferentes, uma de estas é a introdução de modificações não autorizadas (defacement), um problema que pode afetar, tanto aos usuários do sistema, como à imagem de uma empresa. Devido a isso, a comunidade de investigadores vem trabalhando em soluções que irão permitir reduzir os riscos de segurança. Este trabalho contribui para esse esforço na medida em que apresenta uma introdução ao estado da arte das técnicas, metodologias e soluções propostas, tanto pela comunidade de investigadores, como pela indústria de segurança informática.

PALAVRAS-CHAVE Defacement; aplicação Web; segurança; vulnerabilidade; segurança da Web; integridade.

Received / Recepción: October 28, 2016 - Accepted / Aceptación: December 12, 2016 *doi:10.18046/syt.v14i39.2341*

I. Introduction

Internet is a technology that connects around three billion of users worldwide increasing every day (Cerf & Quaynor, 2014). Thanks to this technology, people, companies and devices perform several tasks, such as information broadcasting through websites. Due to the significant advances in technology, the volume of sensitive information and the lack of security expertise in the development of web applications (Dalai & Jena, 2011; Jericho & Munge, 2000), the number of attacks has been increasing seriously, for example WhiteHat Security (2016) presents on its study of websites vulnerabilities, its results demonstrates the number of SQL Injection attacks and Cross Site Scripting percentage increased 50-52% and 49%-55% thorugh the years 2013-2015, respectively. The above ones are some reasons why software development must involve good practices and tools to protect websites from attackers.

The websites contain static or dynamic documents. Hence, the information flow between the server and web browser may be either unidirectional or bidirectional, respectively. Often, attackers try to modify the files within a server, thereby mutating the content of the website or using system resources for warez distribution (Stuttard & Pinto, 2011).

There are different vulnerabilities which a Web Application can be compromised with, among the most representative are: SQL Injection, loss of authentication and session management, Script cross-site scripting (XSS), insecure direct Snaps, incorrect security settings, exposure of sensitive information, no access to control functions, falsification of cross-domain requests (CSRF), using components with known vulnerabilities, unvalidated redirects and forwards (Harper et al., 2015).

Attacks on websites can have different purposes and one of these is the introduction of unauthorized modifications (Website Defacement). Defacement can be classified into two categories: Substitutive Defacement refers to the replacement of legitimate content and Additive Defacement happens when a link pointing to an external URL is placed over the content of the website (Bartoli, Davanzo, & Medvet, 2009). Some of the most recent cases of defacement are: A hacker was able to gain access to Gaana website through a SQL injection vulnerability becoming it as compromised (Kumar, 2015). On the other hand, in 2013 a story about a group of cyber attacks carried out by a *hacktivist* group was published, those movements involved to modify several countries websites as a target (Wei, 2015). In addition, the website of the US Army was modified and discharged (Gross, 2015). Finally,

I. Introducción

Internet es una tecnología que conecta alrededor de tres mil millones de usuarios en todo el mundo, un número que aumenta con el transcurrir de los días (Cerf & Quaynor, 2014). Gracias a esta tecnología las personas, las compañías y los dispositivos pueden realizar distintas tareas, como por ejemplo la difusión de su información a través de los sitios web. Debido a los grandes avances en la tecnología, el volumen de información sensible y la falta de conocimiento en seguridad en el desarrollo de aplicaciones web (Dalai & Jena, 2011; Jericho & Munge, 2000), el número de ataques a los sitios web ha aumentado significativamente, como lo demuestra el estudio de WhiteHat Security (2016) a vulnerabilidades de páginas web, quien indica que durante 2013 y 2015 la cantidad de ataques de tipo *SQL Inyección y Cross Site Scripting* aumentó de 50 a 52% y de 49 a 55%, respectivamente. Las anteriores son algunas razones por las se debe contar con buenas prácticas en el desarrollo del software y con herramientas que permitan proteger de los atacantes a los sitios web.

Las páginas web son sitios que contienen documentos estáticos o dinámicos, dependiendo de lo anterior, el flujo de información entre el servidor y el buscador web será unidireccional o bidireccional, respectivamente. Usualmente, los atacantes buscan modificar los archivos dentro de un servidor, alterando así el contenido del sitio web, o hacer uso de los recursos del sistema para la distribución de *warez* (Stuttard & Pinto, 2011).

Existen distintas vulnerabilidades con las cuales una aplicación web puede verse comprometida, entre las más representativas se encuentran: Inyección SQL, pérdida de autenticación y gestión de sesiones, secuencia de comandos en sitios cruzados [XSS], referencia directas inseguras a objetos, configuración de seguridad incorrecta, exposición de información sensible, ausencia de control de acceso a funciones, falsificación de peticiones en dominios cruzados [CSRF], utilización de componentes con vulnerabilidades conocidas, redirecciones y reenvíos no validados (Harper et al., 2015).

Los ataques a las páginas web pueden tener distintas finalidades, una de estas es la introducción de modificaciones no autorizadas [*website defacement*]. Un *defacement* puede estar clasificado en dos categorías: *substitutive defacements*, que se refiere al reemplazo del contenido legítimo; y *additive defacements*, que sucede cuando se enlaza una URL sobre el contenido de la página web (Bartoli, Davanzo, & Medvet, 2009). Algunos de los recientes casos de *defacement* se encuentran a través de una vulnerabilidad de inyección SQL, un hacker pudo obtener acceso al sitio web de Gaana para así comprometerla (Kumar, 2015). Por otra parte, en 2013 se publicó una noticia sobre un conjunto de ciberataques realizados por un grupo *hacktivista*, cuyas actividades tenían como finalidad alterar las páginas web de los gobiernos de distintos países (Wei, 2015); asimismo, el sitio web del ejército de los EE.UU fue alterado y dado de baja

(Gross, 2015). Los anteriores casos hacen parte de una gran cantidad de ataques que se han presentado durante los últimos años, algunas estadísticas se pueden tomar de la página Zone-H (2016), un sitio web que ha recolectado más de doscientos mil reportes de páginas con *defacement*.

Existen distintas técnicas, metodologías y sistemas para la detección de ataques que generan *defacement* en una página web; como técnicas se encuentran el análisis dinámico, estático y la utilización de la inteligencia artificial. Por otra parte, existen diversas metodologías de trabajo en donde se describen las actividades para el análisis de ataques web (Bartoli, Davanzo, & Medvet, 2010; Zhong, Asakura, Takakura, & Oshima, 2015; Roesch, 1999). Adicionalmente, se han propuesto distintas soluciones como por ejemplo: Tripwire (Kim & Spafford, 1994), contenidos de políticas de seguridad (Stamm, Sterne, & Markham, 2010), IPVmon (IPVTec, 2014), StatusCake (Barnes, 2013), Socuri (2016), y Nagios (Aman, Yamashita, Sasaki, & Kawahara, 2014), entre otras.

Como se ha mencionado, una aplicación web puede estar expuesta a distintas vulnerabilidades, gran parte de ellas se debe a: malas prácticas durante el desarrollo del software, desconocimiento de la seguridad informática, desarrollos a la medida, ataques de día cero y mejora en las habilidades de los cibercriminales, entre otras (Stuttard & Pinto, 2011). En conclusión, es difícil garantizar una solución con 100% de seguridad y es por ello que el objetivo es brindar técnicas o herramientas que permitan reducir los riesgos que puedan generar alguna amenaza al sistema o a los usuarios.

En este artículo se realiza el estudio de algunas de las técnicas, metodologías y herramientas más representativas que dan solución al problema de *defacement* en las páginas web. El trabajo se divide en los siguientes componentes: La sección 2 incluye la información sobre los ataques informáticos más representativos que provocan *defacement* sobre las páginas web; en la sección 3 se exploran algunos estudios sobre las vulnerabilidades del lenguaje PHP; en la sección 4 se abordan los trabajos para la detección y control de los ataques que provocan *defacement* en las páginas web; y finalmente, en la sección 6, se resumen las oportunidades por abarcar y las conclusiones.

II. Ataques informáticos a sitios web para defacement

Existen distintas vulnerabilidades a las cuales un sitio web puede verse expuesto, pero no todas las anteriores pueden desencadenar un problema de *defacement*, a continuación se describen las más importantes según el TOP 10 OWASP 2013.

Inyección SQL

Esta vulnerabilidad se presenta por la falta de control en los parámetros que se pueden ingresar en una página web. Las aplicaciones web que utilizan contenido dinámico obtienen su información a través de peticiones SQL a una base

above scenarios are part of a lot of attacks which have occurred in recent years, some statistics can be taken from the Zone-H page (www.zone-h.org), a website that has collected more than 200 thousand reports of pages with defacement.

There are several techniques, methods and systems for detecting attacks which leads for defacement on a website; such as dynamic and static analysis and the use of artificial intelligence. Furthermore, there are different work methodologies which describe the analysis of web attacks (Bartoli, Davanzo, & Medvet, 2010; Zhong, Asakura, Takakura, & Oshima, 2015; Roesch, 1999). In addition, many solutions have been proposed such as: Tripwire (Kim & Spafford, 1994), content security policy (Stamm, Sterne, & Markham, 2010), IPVmon (IPVTec, 2014), StatusCake (Barnes, 2013), Socuri (2016), and Nagios (Aman, Yamashita, Sasaki, & Kawahara, 2014), among others.

As mentioned above, a web application may exposure to different vulnerabilities which could be given by missing the best programming practices during software development, the lack of computer security, the custom development, zero-day attacks, improvement on cybercriminal skills, among others (Stuttard & Pinto, 2011). In conclusion, it is difficult to guarantee a solution on a hundred percent certainty and that is why the goal is to provide techniques or tools to reduce risks that may cause a threat to the system or users.

This paper describes the study of some of the techniques, methodologies and more representative tools which mean a solution for defacement on websites. The given project is partitioned into the following components: section 2 includes information about the most representative computer attacks that cause defacement on websites; in section 3 studies on vulnerabilities of PHP language are explored; in section 4 development on detection and attacks control generated by defacement on websites are approached; finally section 5 summarizes cover outstanding opportunities and conclusions.

II. Informatics attacks to websites for defacement

There are several vulnerabilities a website can be exposed, but not all of the above ones may generate a defacement issue, indeed the most important of these are described from a TOP 10 OWASP 2013 study as follows.

SQL Injection

It is a vulnerability that is presented by the lack of control over the parameters that can be entered on a website. Dynamic Web applications capture information through SQL requests to a database, this technology may be affected when

an attacker has the possibility to enter code via deliberated parameters (Stuttard & Pinto, 2011). Platforms such as PHP, ASP and JSP are also impacted by this type of vulnerability (Ullrich & Lam, 2008).

Broken of authentication and session management

This vulnerability occurs by the presence of flaws in the authentication mechanism of a web application, which can be compromised through passwords generators, brute force attacks, identity theft, among others (Stuttard & Pinto, 2011).

Script cross-site

XSS vulnerability allows attackers to affect users of the web application through uncontrolled validations. A cybercriminal could take advantage of this vulnerability by commands HTML / JavaScript insertion and access to user data procurement, unauthorized actions, user redirection to malicious sites, among other activities (Stuttard & Pinto, 2011).

Insecure direct object references

Through a minimum control to internal objects of a web application, an attacker could access or manipulate its references as malicious acts.

Security missconfiguration

A bad configuration could affect the safety of users or the web application; hence, it is a good practice to apply the required activities for a good setting of the technologies and their respective updates.

Sensitive data exposure

As its name suggests, it refers to the vulnerability of the web applications by not effectively protect sensitive data.

Missing function level access control

It is important to have a user access control to each function of the web application due to cybercriminals may take advantage of unauthorized privileges and perform requests that might affect users or the system.

Cross-site request forgery

It is a vulnerability that allows cybercriminals to include commands that take advantage of the session and user privileges. The attack pretends to disrupt the operation of the application and induce users to perform seemingly legitimate actions on malicious environments (Stuttard & Pinto, 2011).

Using components with known vulnerabilities

It is important to know the technologies used in the web application, because of any component can be vulnerable and become a risk to the safety of the system.

de datos, las cuales pueden verse afectadas cuando un atacante tiene la posibilidad de introducir código a través de parámetros no controlados (Stuttard & Pinto, 2011). Las plataformas como PHP, ASP y JSP también sufren este tipo de vulnerabilidad (Ullrich & Lam, 2008).

Pérdida de autenticación y gestión de sesiones

Esta vulnerabilidad se presenta cuando hay falencias en el mecanismo de autenticación de una aplicación web, la cual puede ser comprometida a través generadores de contraseñas, ataques de fuerza bruta y suplantación de identidad, entre otros medios (Stuttard & Pinto, 2011).

Secuencia de comandos en sitios cruzados

Una vulnerabilidad de XSS permite a los atacantes afectar a los usuarios de la aplicación web a través de validaciones no controladas. Un cibercriminal podría explotar esta vulnerabilidad a través de la inserción de comandos de HTML/JavaScript y obtener acceso a los datos de los usuarios, realizar acciones no autorizadas en su nombre y redireccionar al usuario a sitios maliciosos, entre otras actividades (Stuttard & Pinto, 2011).

Referencias directas inseguras a objetos

A través de un bajo control a los objetos internos de una aplicación web, un atacante podría acceder o manipular sus referencias para actos maliciosos.

Configuración de seguridad incorrecta

Una mala configuración podría afectar la seguridad de los usuarios o la aplicación web, por lo que es una buena práctica aplicar las actividades necesarias para una buena configuración de las tecnologías empleadas con sus respectivas actualizaciones.

Exposición de información sensible

Como su nombre lo indica, es la vulnerabilidad a la que se ven expuestas las aplicaciones web al no proteger efectivamente los datos sensibles.

Ausencia de control de acceso a funciones

Es importante controlar los accesos de los usuarios a cada función de la aplicación web, debido a que los cibercriminales podrían aprovecharse de privilegios no autorizados y realizar peticiones que podrían afectar a los usuarios o al sistema.

Falsificación de peticiones en dominios cruzados

Es una vulnerabilidad que permite a los cibercriminales introducir comandos que se aprovechan de la sesión y los privilegios de los usuarios, el ataque consiste en alterar el funcionamiento de la aplicación e inducir a los usuarios a realizar acciones aparentemente legítimas sobre ambientes maliciosos (Stuttard & Pinto, 2011).

Utilización de componentes con vulnerabilidades conocidas

Es importante conocer las tecnologías empleadas en la aplicación web, lo anterior se debe a que cualquier componente puede ser vulnerable y representar un riesgo para la seguridad del sistema.

Redirecciones y reenvíos no validados

Es una vulnerabilidad a la que se ve expuesta la aplicación web, que es aprovechada por los cibercriminales para redireccionar a las víctimas a sitios con contenidos maliciosos.

III. Vulnerabilidades en PHP

Existen diversos lenguajes de programación para el desarrollo de páginas web, en esta sección se abarcará el lenguaje PHP debido a que es una de las tecnologías más utilizadas (Futoransky, Gutesman, & Waissbein, 2007). PHP es un lenguaje de programación del lado del servidor que se ha extendido por todo el mundo empleando servidores como parte de la plataforma Linux-Apache-MySQL-PHP [LAMP]. Sus vulnerabilidades son muy comunes (Son& Shmatikov, 2011) debido a que es uno de los lenguajes más usados; según el *Top-ten languages of public open source projects* (Amanatidis & Chatzigeorgiou, 2016) su porcentaje de uso en los servicios de *SourceForce* y *GitHub* es de 11% y 9%, respectivamente; en un estudio publicado en el blog de la IEEE Spectrum, Diakopoulos y Cass (2015), ubican a PHP en la cuarta posición, con una popularidad que asciende, aproximadamente, a 1.3 millones de direcciones IP y 18 millones de dominios; además, se encuentra instalado en el 50% de servidores Apache (Nguyen-Tuong, Guarnieri, Greene, Shirley, & Evans, 2005; Gupta & Gupta, 2015).

PHP tiene una *suite* que construye y hace operaciones especiales para el desarrollo web, dentro de las cuales se mencionan: *Natural integration with SQL, Dynamic types and implicit casting to and from strings, Variable scoping and the environment* (Xie & Aiken, 2006). Sin embargo, lo anterior trae consigo algunas fallas con respecto a la seguridad. Existen dos grandes ataques que se presentan en este lenguaje: los ataques de inyección, que tienen como finalidad construir solicitudes al servidor que ocasionan un estado corrupto, o revelar información confidencial; y los ataques de salida, que son intentos de envío de solicitudes a los servidores Web para generar respuestas que producen ambientes maliciosos a los clientes (Nguyen-Tuong et al., 2005; Eshete, Villafiorita, & Weldemariam, 2011).

Hoy en día, las bases de datos son componentes esenciales de cualquier aplicación web debido a que les permiten tener una amplia variedad de contenido dinámico. Puesto que se puede almacenar información secreta en una base de datos, debería considerarse la protección de estas. Es primordial aclarar que PHP no posee un mecanismo autónomo que permita proteger una base de datos, por tal motivo se dan unas breves recomendaciones para la seguridad de estos almacenamientos (Jingling & Rulin, 2015):

- Diseño de la bases de datos. Se recomienda, primero crear los distintos roles de usuarios para que la aplicación tenga permisos muy limitados, una vez creados se puede aplicar el proceso de administración de perfiles que permita asignar, actualizar y borrar los privilegios de cada usuario, con el fin de que un

Unvalidated redirects and forwards

It is a vulnerability to the web application, which is used by cybercriminals in order to redirect victims to malicious content sites.

III. PHP Vulnerabilities

There are several programming languages for the development of websites. This section will study the PHP language, one of the most adopted technologies (Futoransky, Gutesman, & Waissbein, 2007). PHP is a server-side programming language has been spreading worldwide using servers as part of the LAMP (Linux-Apache-MySQL-PHP) platform. Vulnerabilities are very common (Son& Shmatikov, 2011) due to it is one of the most used languages. According to the top-ten languages of public open source projects (Amanatidis & Chatzigeorgiou, 2016), its percentage of use in the services of SourceForce and GitHub is 11% and 9%, respectively. A study published in the IEEE Spectrum blog (Diakopoulos & Cass, 2015), PHP ranks fourth with a popularity of approximately 1,3 million IP addresses, 18 million domains and installed on 50% of apache servers (Nguyen-Tuong, Guarnieri, Greene, Shirley, & Evans, 2005; Gupta & Gupta, 2015).

PHP has a suite to build and make special operations to a web development, such as: Natural integration with SQL, Dynamic types and implicit casting to and from strings, Variable scoping and the environment (Xie & Aiken, 2006). Nevertheless, the above presents several failures with respect to security. There are two major attacks of this language: injection attacks –that pretends to make requests to the server that cause a corrupt state or reveal confidential information–, and outbound attacks –attempts to send requests to web servers that generates responses of malicious environments to customers– (Nguyen-Tuong et al., 2005; Eshete, Villafiorita, & Weldemariam, 2011).

Nowadays, databases are essential components of any web application due to his wide variety of dynamic content. Because of the possibility to collect secret information, database protection should be considered. PHP does not have an autonomous mechanism that allows a database protection; therefore brief recommendations are given for the security of these storages (Jingling & Rulin, 2015):

- Database design: First, different roles of users should be created in order to the application has limited permissions; then, it is posible to elaborate the profile management process that allows to assign, update and delete the privileges of each user, in order that an intruder can not access the database using the application

credentials and that just do what the application allows to do it. Second, it should not implement all business logic in the web application, it must be created in the database schema using views or rules.

- Connection to a database: It is recommended to establish connections over SSL or over SSH, it allows encrypting the connection between the clients and the server, hence it will be more difficult for an attacker to obtain information from the databases.
- Encrypted storage model: SSL and SSH are tools that do not protect the information belong to a database. It is recommended, first, to create an own encryption package and use it in PHP scripts. It lets to ensure the data before inserting and obtaining them.

Open Web Application Security Project (OWASP) (2016) highlights the issue of SQL injection and answers the following questions: ¿How do you realize that a web page is vulnerable? ¿How can it be protected? (Vanderaj, 2016), and suggest:

- to use the mysql_query() function or to validate that the user can not insert SQL statements malicious for the website;
- migrate to PHP 5.1 in order to employ the most recents methods which reduce the attacks generated.validate correctly the datatype, the datatype size, and the datatype syntax;
- not use names for dynamic tables; and
- always to choose the positive validation.

PHP-Sensor is a model presented in a publishment for the discovery of a rape attack workflow and Cross-Site Scripting [XSS] type attacks (Gupta & Gupta, 2015). Authors declare that a rape attack workflow occurs when there is an inconsistency during the logical sequence of the user and system exchanges before a transaction ends; PHP-Sensor has a process that is divided into two phases (extraction and recognition) for the detection of these attacks through an extraction of axioms sets (three types of axioms). On the other hand, for the detection of XSS worms in the client-side interface, PHP-Sensor uses only the recognition phase. The detection process uses HTTP request/response information, for attacks in the workflow, each axiom is compared with the data obtained to detect some inconsistency. Moreover, the model was evaluated in both phases against numerous open source PHP applications, the offline phase presented good results in the development of axioms with a minimum of false positi-

intruso no pueda obtener acceso a la base de datos utilizando las credenciales de la aplicación y que sólo haga lo que la aplicación le permite hacer; segundo, no implementar toda la lógica de negocio en la aplicación web, si no crear en el esquema de la base de datos utilizando vistas o reglas.

- Conexión a una base de datos. Se recomienda establecer las conexiones sobre SSL o bien sobre SSH, esto permite encriptar la conexión entre los clientes y el servidor, así será más difícil para un atacante obtener información de la bases de datos.
- Modelo de almacenamiento encriptado. SSL y SSH son herramientas que no protegen la información que se encuentra en una base de datos, se recomienda entonces, crear primero un paquete de encriptación propio y utilizarlo en los scripts de PHP, pues así se podrían asegurar mejor los datos antes de insertarlos y obtenerlos.

Open Web Application Security Project [OWASP] (2016) resalta el tema de inyección por SQL y responde a las preguntas: ¿cómo darse cuenta que una página web es vulnerable? y ¿cómo puede protegerse? (Vanderaj, 2016), y recomienda:

- emplear la función mysql_query() o validar que los campos de ingreso de información por parte del usuario no contengan sentencias SQL que interactúen con la base de datos;
- migrar a PHP 5.1 con la finalidad de usar los métodos más recientes que reducen el número de ataques inyectados;
- validar el tipo de dato de forma correcta; el tamaño del tipo de dato; la sintaxis del tipo de dato;
- no usar nombres de tablas dinámicas; y
- siempre preferir la validación positiva

PHP-Sensor es un modelo presentado en una publicación para el descubrimiento del flujo de trabajo de un ataque de violación y de ataques de tipo Cross-Site Scripting [XSS] (Gupta & Gupta, 2015). Los autores enuncian que un flujo de trabajo de un ataque de violación se presenta cuando hay una inconsistencia durante la secuencia lógica de la serie de intercambios del usuario y el sistema antes de terminar una transacción; PHP-Sensor tiene un proceso que se divide en dos fases (extracción y reconocimiento) para la detección de estos ataques a través de una extracción de conjuntos de axiomas (tres tipos de axiomas). Por otra parte, para la detección de gusanos XSS en la interfaz del lado del cliente, PHP-Sensor utiliza solamente la fase de reconocimiento. El proceso de detección emplea la información de los HTTP *request/response*, para los ataques en el flujo de trabajo se compara cada axioma con los datos obtenidos para detectar alguna inconsistencia. El modelo fue evaluado en ambas fases contra numerosas aplicaciones PHP de código abierto: la fase *offline* presentó muy

buenos resultados en el desarrollo de los axiomas con un mínimo de falsos positivos; y en la fase real PHP-Sensor se detectaron los catorce ataques controlados. Sin embargo, por más robusto que sea este modelo, se presentan algunos inconvenientes:

- PHP-Sensor no se puede usar en las aplicaciones web que han sido desarrolladas bajo plataformas como AJAX; y
- PHP-Sensor ha restringido la capacidad de controlar las restricciones más difíciles en la base de datos.

Jingling y Rulin (2015) proponen un *framework* que integra el análisis estático y dinámico para detección de vulnerabilidades de seguridad en aplicaciones desarrolladas en PHP. En el análisis estático se utilizó la herramienta HHVM para que el código fuente PHP fuera representado en un Árbol Abstracto Sintáctico [AST] para el análisis de los datos. El proceso de análisis de los datos se divide en tres pasos: la primera actividad consiste en convertir el código PHP en una estructura AST, la segunda fase se encarga de construir un CFG [*Control Flow Graph*] del AST, y la tercera tarea de detectar las vulnerabilidades, recuperar las variables y los parámetros sobre el CFG. Adicionalmente, cada variable ($USERNAME, $_GET, $_SERVER, entre otros) y parámetro evaluado fue representado sobre una estructura de datos compuesta por tres categorías –tabla de símbolos, cadenas y objetos–, y a su vez, cada estructura puede dividirse en tres categorías –NoRef, RefToIndex y RefToStorage–. Por otra parte, en el análisis dinámico se utiliza el *fuzz testing* a partir de tres actividades: la primera obtiene las URL, posteriormente se abstrae el parámetro y el valor de cada URL y finalmente se emplea el *fussing testing* sobre el parámetro y el valor obtenidos. Adicionalmente, la evaluación del experimento incluyó 110 programas PHP, la colección incluye programas con vulnerabilidades y otros que integran parches para la vulnerabilidad. Finalmente, el estudio utilizó TipaskV2.5, YxcmsV1.2.6, EspcsmV6.2 y otros CMS, y como resultado obtuvo una cobertura del código de más del 92%, y de un 72% para las URL abstraídas.

IV. Trabajos para el control del web defacement

Existen dos enfoques para la detección de ataques a las aplicaciones web (Zhong et al., 2015): una técnica está *basada en firmas*, y su objetivo es detectar los ataques a partir de una base de datos que contiene distintas características de los datos transmitidos de una comunicación maliciosa; otras en la *detección de anomalías*, y está compuesta por dos fases, la primera se encarga de la construcción de un perfil de la página web a partir de distintas características extraídas de las peticiones legítimas HTTP, y la segunda de monitorear y detectar si hay alguna anomalía en las peticiones respecto del perfil creado. Por otra parte, existen distintas técnicas para el análisis en la seguridad de la computación (Urcuqui & Navarro, 2016), entre estas se encuentran:

ves, and in the actual PHP-Sensor phase the 14 controlled attacks were detected. Finally, however robust this model may be, the paper describes some drawbacks:

- PHP-Sensor cannot be applied on web applications had developed under platforms such as AJAX; and
- PHP-Sensor has limited the capacity to control the most difficult restrictions on the database.

Jingling y Rulin (2015) propose a framework that integrates static and dynamic analysis for the detection of security vulnerabilities in PHP applications. HHVM tool was used on the static analysis so that the PHP source code was represented in an abstract syntactic tree (AST) for the data analysis. The process of data analysis is divided into three steps: the first activity consists to convert the PHP code into an AST structure, the second phase is responsible to transform an AST into CFG (Control Flow Graph), and the third task is the responsible for vulnerabilities detection, retrieving the variables and parameters on the CFG. In addition, each variable ($ USERNAME, $ _GET, $ _SERVER, among others) and parameter evaluated were represented on a data structure composed of three categories (table of symbols, chains and objects). Also, each structure can be divided into three categories: NoRef, RefToIndex, and RefToStorage. Otherwise, into dynamic analysis the fuzzy testing was applied in the next activities: the process evaluates every URL in order to get parameters, values and to apply the fussing testing. Moreover, the experiment evaluation included 110 PHP programs; the collection includes programs with vulnerabilities and others with vulnerability patches. Finally, the study used TipaskV2.5, YxcmsV1.2.6, EspcsmV6.2 and other CMS; as a result, code coverage of over 92% and 72% were obtained for abstracted URLs.

IV. Web defacement management development

There are two approaches to the web applications attacks detection (Zhong et al., 2015): a technique is based on signatures in order to detect attacks from a database that contains several characteristics of the data transmitted from a malicious communication; the other is the technique of anomalies detection, which is composed of two phases: the first one is responsible to build a web page profile using different features extracted of the legimitate HTTP requests, and the second one is responsible to monitor and detect any anomaly into the requests with the profile previously created. Nevertheless, there are different techniques for the analysis on computer security (Urcuqui & Navarro, 2016), among these are:

- static analysis, a technique used for anomalies detection in a source code, data or binary files without system execution (Batyuk et al., 2011);
- dynamic analysis, a method that performs an analysis of the system execution behavior; the technique allows a network traffic evaluation, sockets opening, user interface, among others (Muñoz & Villalba, 2012); and
- artifical intelligence, an area that provides a set of techniques to obtain an approximate solution to complex problems.

Roesch (1999) explains and evaluates the utility of Snort. Snort is a sniffer that allows the intruders detection in the system network from the setting of a traffic capture rules. Each rule is configured as a key that represents services not available on the system network. In the course of traffic capture, Snort detects intrusions from the comparison of packets transmitted on the network against configured rules and the tool generates an alert that is transmitted to the system administrator. Finally, the types of rules included by Snort are: P2P, backdoor, denial of service attacks, web attacks, viruses, among others (Caswell, Beale, & Baker, 2007).

PSigene is an intrusion detection system [IDS] that automatically collects and generates signatures of benign and malicious requests (Howard, Gutierrez, Arshad, Bagchi, & Qi, 2014). The study is focused on SQLi attack. Its signature development process is divided into four activities: the first one consists on downloading information from several cybersecurity portals (such as Security Focus, Exploit Database, PacketStorm Security and Open Source Vulnerability Database). Besides, a total of 30,000 examples of SQLi attacks and 240,000 examples of benign HTTP traffic were obtained following the collection of pages. The second activity consists in the classification of the information through a clustering algorithm, hence 159 categories were created: Reserved SQL words, SQLi signatures of Bro and Snort intrusion detection systems, and SQLi documents of reference. Moreover, a binary array of 30,000 by 159 was created, where its values represent whether or not each category can be found on each reading of the SQLi attacks. On the other hand, pSigene was implemented in conjunction with the firms of Bro, Snort, ModSec and other signature generation algorithms against three datasets (traffic of a university network, results of the SQLmap, Arachni and Vega application) to evaluate the performance of the firms. Therefore, the results show that for the SQLmap dataset the tool performs better than Snort and Bro, with 86.23% for

- el análisis estático, técnica para detección de anomalías en el código fuente, datos o archivos binarios sin ejecución del sistema (Batyuk et al., 2011);
- el análisis dinámico, método que realiza un análisis del comportamiento del sistema en ejecución y permite evaluar el tráfico de la red, la apertura de sockets y la interfaz de usuario, entre otros (Muñoz & Villalba, 2012); y
- la inteligencia artificial, área que provee un conjunto de técnicas que permiten dar una solución aproximada a problemas complejos.

Por su parte, Roesch (1999) explica y evalúa la utilidad de Snort, un *sniffer* que permite la detección de intrusos en la red de un sistema a partir de la configuración de unas reglas de captura de tráfico. Cada regla está configurada como una clave que representa a los servicios que no están disponibles en la red del sistema. Durante la captura del tráfico, Snort detecta las intrusiones a partir de la comparación de los paquetes transmitidos en la red contra las reglas configuradas y genera una alerta que es transmitida al administrador del sistema. Los tipos de regla que incluye Snort son: P2P, puertas traseras, ataques de denegación de servicio, ataques web y virus, entre otros (Caswell, Beale, & Baker, 2007).

PSigene es un sistema para la detección de intrusos [IDS] que recopila y genera automáticamente firmas generales de las peticiones benignas y maliciosas (Howard, Gutierrez, Arshad, Bagchi, & Qi, 2014). El estudio se enfocó en los ataques SQLi e inició con la descarga de la información de distintos portales de ciberseguridad –Security Focus, Exploit Database, PacketStorm Security y Open Source Vulnerability Database, entre otros– y obtención de un total de 30.000 ejemplos de ataques SQLi y 240.000 ejemplos de tráfico benigno HTTP. Continuó con la categorización de la información a través de un algoritmo de *clustering*, para lo cual se crearon 159 categorías –palabras reservadas de SQL, firmas de SQLi de los sistemas de detección de intrusos Bro y Snort, y documentos de referencias de SQLi– y se creó una matriz binaria de 30.000 por 159, cuyos valores representan si cada categoría se encuentra o no sobre cada lectura de los ataques SQLi. En la evaluación se implementó pSigene en conjunto con las firmas de Bro, Snort. ModSec y otros algoritmos de generación de firmas, contra tres *datasets* –tráfico de una red universitaria, resultados de la aplicación de SQLmap, Arachni y Vega–, para evaluar el desempeño de las firmas. Los resultados muestran que para el *dataset* de SQLmap la herramienta tiene un desempeño mayor que Snort y Bro, con un 86.23% para nueve firmas y 82.72%, pero es más bajo ModSecurity (96.07%); los resultados son muy similares a los obtenidos contra el *dataset* de Arachni.

Bartoli et al., (2010) proponen un *framework* –Goldrake– para detección de *defacement* en páginas web de alto contenido dinámico. En *Goldrake* se implementa la técnica de detección por anomalías, como parte de su valor agregado

el *framework* puede ser integrado a un servicio de monitoreo sin requerir alguna instalación sobre la infraestructura de la página web. Las fases de entrenamiento y monitoreo cuentan con una cantidad de 45 sensores encargados de extraer las características de cada URL. Los sensores se encuentran organizados en cinco grupos: en el primer conjunto se encuentran los sensores de cardinalidad, encargados de generar datos numéricos de los objetos, por ejemplo, el número de líneas; en el segundo conjunto se encuentran los sensores encargados de computar la frecuencia relativa de cada elemento en su respectiva clase, por ejemplo, cada carácter del conjunto ASCII y cada etiqueta HTML; el tercer grupo de sensores cuenta el número de veces que cada elemento recurrente no aparece en una lectura; en la cuarta categoría cada sensor construye un árbol de recursos HTML/XML; y el quinto conjunto de sensores tienen como finalidad la búsqueda de atributos y la generación de alertas por cada lectura. Por otra parte, se desarrolló un prototipo para el experimento, la evaluación contó con un conjunto de 300 recursos dinámicos y el monitoreo se realizó cada seis horas durante cuatro meses contra un conjunto de 900 ejemplos de páginas con *defacement*. Los resultados de la evaluación demuestran que el prototipo tuvo un buen desempeño en la tasa de falsos negativos y falsos positivos.

Davanzo, Medvet, y Bartoli (2011), por su parte, evalúan distintas técnicas para la detección de anomalías a partir de aplicación de los algoritmos de *machine learning* y los perfiles generados de las páginas web. El trabajo hace mención al *framework* (Wei, 2015) para la generación de las características, los algoritmos utilizados fueron: *kth nearest*, *Local Outlier Factor, hotelling´s T-Square, parzen windows, Support Vector Machines, y Domain knowledge aggregator*. Durante el experimento se utilizaron dos *datasets*: el primer conjunto de datos estuvo compuesto por capturas de 300 páginas web de alto contenido dinámico y 320 lecturas de casos reales de *defacement* obtenidos de Zone-h; y el segundo haciendo referencia a un resultado de un trabajo previo que contiene 125 capturas malignas. El desempeño fue evaluado en términos de falsos positivos y falsos negativos, se realizó una prueba sobre los conjuntos con todas las características y otra con una selección/filtrado de las características. Los resultados del experimento fueron los siguientes: para el segundo *dataset* y todas las características el mejor algoritmo fue *DomainKnowledge*¸ para el anterior *dataset* y con la reducción de las características el mejor algoritmo fue SVM, y para el conjunto más grande sin infiltrado la mejor solución fue SVM. Finalmente, para distintas configuraciones el *Domain Knowledge* fue el más rápido en los tiempos de respuesta

Zhong et al., (2015) exponen un método que emplea una estructura abstracta de parámetros creada a partir de la captura de las peticiones HTTP. En el estudio se utilizó una fase de entrenamiento que cuenta con tres pasos: transformación, filtrado y determinación del tipo de per-

9 signatures and 82.72%, although it is under ModSecurity (96.07%); the results are highly similar to those obtained against the Arachni dataset.

Bartoli et al., (2010) propose a framework –Goldrake– to detect defacement in websites with high dynamic content. In Goldrake anomalies detection is implemented and as part of his added value the framework can be integrated to a monitoring service without any instalation on the website infraestructure. Training and monitoring phases have 45 sensors responsible for obtaining the characteristics of each URL. Sensors are joined in 5 groups. First, there are the cardinality sensors which are responsible for generating the objects numerical data (such as the number of lines). In the second set, there are sensors responsible for computing the relative frecuency of each elements in its respective class (such as each caracter of the ASCII set and each HTML tag), the third goup of sensor counts the time that each recurrent element does not appear in a reading, in the fourth one each sensor create a HTML/XML resources tree, and the fifth set of sensors are aimed to find attributes and to generate alerts for each reading. On the other hand, a prototype was developed for experiment development, the examination set 300 dynamic resources and a monitoring each 6 hours for 4 months against a set of 900 examples of websites with defacement. As a result, a well performance of the prototype was obtained through falses negatives and falses positives rates.

Continuing with the project done by the mentioned researchers, Davanzo, Medvet and Bartoli (2011) evaluate several techniques for the anomalies detection through the machine learning algorithms applications and the profiles of websites. This study makes mention to framework (Wei, 2015) which is responsible for the characteristics and algorithms generations, such as kth nearest, Local Outlier Factor, hotelling´s T-Square, Parzen Windows, Support Vector Machines and Domain Knowledge Aggregator. During the experiment datasets were used, the first set of data was composed of 300 websites with high dynamic content and 320 readings of real cases of defacement obtained from Zone-H, and the second one refers to a result of a previous study that contains 125 malicious catches. Performance was assessed in terms of false positives and false negatives. A test was performed on the sets with all the characteristics and another with a selection/filtering of the characteristics. Additionally, the experiment generated these results: for the second dataset the best algorithm was Domain Knowledge Aggregator, for the previous dataset and with the reduction of the characteristics the best algorithm was Support Vector Machines, and for the

larger set without filtering the best solution was Support Vector Machines. Finally, for several configurations the Domain Knowledge Aggregator was the fastest in response times.

Zhong et al., (2015) expose a method that employs an abstract parameter structure created from the capture of HTTP requests. The study used a training phase composed of three steps: transformation, filtering and determination of the profile type. Transformation is responsible for obtaining each parameter of HTTP requests and to change them into a sequence of characters class. Each class is expresed by a regular expression and its categorization depends on its type (alphabets, numbers, symbols, URI, Zip Code, and email addresses). The second stage is responsible for assigning a frecuency to each of the clases for a later filtering. Subsequently, each class is assigned to a profile based on a set of association rules. Nonetheless, the method has a last stage which is responsible about the detection. The given process has the same activities for parameters extraction and mutation on requests for characters sequences classes, however, it differs since it evaluates each parameter similarity with its related profile in terms to stand whether it is either a right or bad request. Therefore, the results agains the Kruegel and Kim methods were evaluated in order to study the technique development, reaching a good result over the false positives rate.

Mohaisen (2015) proposes a system in which machine learning algorithms are used for the identification of malicious pages and the prediction of the classes of a website vulnerabilities. This study integrates two activities: the first one applies an SVM classification algorithm based on a stocasthic descendent gradient, which is responsible for malignant and benign pages classification. Training was performed from a binary matrix composed of 41 columns by 20 thousands rows. Each binary data is a label that represents a malicious or benign object obtained from a process of characteristics extraction of a dataset of 10 thousand benign URLs and 10 thousand malicious URLs. Consequently, in the second stage a classifier was used for 12 types of vulnerabilities, each algorithm was trained with malicious objects that represent either one or several vulnerabilities. Finally, a result is a classification algorithm with a 93% performance in injection and back door detections in the server; excluding the samples of the dataset injection it was obtained a precision of 96% and 91% in the detection of malicious objects.

Medvet, Fillon, y Bartoli (2007) developed a study in which the genetic exploration is explored [GP] in the detection of the web defacement. The analysis process is base on anomalies detection technique, its training phase consists on fil. Transformación es la etapa que se encarga de obtener cada parámetro de las peticiones HTTP y de transformarlos a una clase de secuencia de caracteres, cada clase se encuentra expresada por una expresión regular y su categorización depende de su tipo (alfabetos, números, símbolos, URI, código ZIP y direcciones de correo electrónico). La segunda etapa se encarga de asignar una frecuencia a cada una de las clases para un posterior filtrado. Posteriormente, a partir de unas reglas de asociación cada clase es asignada a un perfil. El método cuenta con una última fase que se encarga de la detección, el proceso cuenta con las mismas actividades de extracción y transformación de los parámetros de las peticiones a clases de secuencia de caracteres, pero con la diferencia de que se evalúa la similitud del valor de cada parámetro con su perfil y así se determina si la petición es benigna o maligna. Finalmente, para el estudio del desempeño de la técnica se evaluaron los resultados contra los métodos de Kruegel y Kim, obteniendo un buen resultado en la tasa de falsos positivos.

Mohaisen (2015) propone un sistema en el que se emplean algoritmos de aprendizaje de máquina para la identificación de páginas maliciosas y la predicción de las clases de vulnerabilidades con las que se ve comprometida una página web. El estudio integra dos actividades: en la primera se utiliza un algoritmo de clasificación de SVM basado en un gradiente descendiente estocástico, que se encarga de la clasificación de páginas malignas y benignas, el entrenamiento se realizó a partir de una matriz binaria compuesta de 41 columnas y 20 mil filas, donde cada dato binario es una etiqueta que representa a un objeto malicioso o benigno obtenido de un proceso de extracción de características de un *dataset* de diez mil URL benignas y diez mil URL maliciosas; en la segunda actividad se utilizó un clasificador para doce tipos de vulnerabilidades, cada algoritmo fue entrenado con objetos maliciosos que representan, ya sea una o varias vulnerabilidades. El estudio tiene como resultado un algoritmo de clasificación con un desempeño del 93% en la detección inyecciones y puertas traseras en el servidor, excluyendo las muestras de inyección del *dataset* y obtuvo una precisión de 96% y 91% en la detección de objetos maliciosos.

Medvet, Fillon, y Bartoli (2007) realizaron un estudio en el cual se explora la programación genética [GP] en la detección del web defacement. Para empezar, el proceso de análisis se basa de la técnica de detección de anomalías, su fase de entrenamiento consiste en la lectura de varios documentos descargados de un conjunto de URL, sobre cada documento se emplea una función de transformación con 53 sensores encargados de tomar distintas características de cada documento –elementos HTML, enlaces externos, entre otros– y generar un vector numérico; juntando los resultados de cada vector se crea una secuencia de aprendizaje que será empleada por la GP en la selección, reproducción y la función *fitness*. Adicionalmente, la función *fitness* incluye como variables la tasa de falsos positivos y

falsos negativos, y la secuencia de aprendizaje. El experimento contó con un *dataset* que contiene información de aproximadamente 720 lecturas de 15 páginas de contenido dinámico, cada lectura se realizó con una frecuencia de seis horas durante un mes. Por otra parte, en la fase de monitoreo se revisó cada documento y se le asignó un valor que indica si la página es benigna o maligna. Sus resultados confirman que los algoritmos generados por la GP cuentan con un buen desempeño en la detección de anomalías, con una baja tasa de falsos positivos y falsos negativos.

DICE es un sistema para detección automática de sitios web con *defacement* (Tanaka, Kai, Tamura, & Sasaki, 2011). El sistema incorpora un método que permite detectar scripts (inyecciones SQL) no autorizados. El método utiliza la teoría de cuantificación tipo II del profesor Chikio Hayashi para el análisis y la discriminación de los datos a través de los siguientes parámetros: alteración del título, múltiples inserciones de scripts, nombres de scripts prejuiciados, un conjunto de URL no autorizadas o prejuiciadas, y una lista de estados en un script. El sistema fue evaluado a través de su implementación en un servidor proxy de libre acceso Squid, se utilizaron 121 URL de contenido malicioso, 229 URL benignas, y de cada tipo se seleccionó aleatoriamente 60 para el ajuste de los parámetros del experimento. El estudio obtuvo una tasa de detección del 84% y un 16% para falsos positivos. Finalmente, los autores proponen la exploración de nuevas características para prevenir otros tipos ataques además de las inyecciones SQL.

Tripwire es una herramienta de libre acceso para ambientes UNIX que facilita la seguridad e integridad de la información del directorio de archivos (Kim & Spafford, 1994). La solución cuenta con la capacidad de notificar a los administradores si se han encontrado cambios en los ficheros y tomar acciones correctivas en un tiempo significativo. Tripwire emplea el método de firmas; en la técnica se evalúa cada uno de los archivos para abstraer unos valores (tamaño, fecha de la última modificación y propietario) y se genera una firma que será almacenada en una lista de comprobación; el sistema revisa periódicamente los archivos contra la lista para verificar si hay alguna inconsistencia. Por otra parte, Fujimura y Jin (2007) abarcan la implementación de un sistema para interpolación de archivos, en su trabajo se realizó una modificación sobre Tripwire, cuenta como valor agregado que la solución tiene la capacidad de ser utilizada, tanto por los administradores, como por los usuarios ordinarios del sistema.

Para abordar el problema del crecimiento de las vulnerabilidades de las aplicaciones web, Stamm et al., (2015) proponen un esquema de restricciones de contenido llamada *Content Security Policy* [CSP]. Las restricciones de contenido son un mecanismo que permite a los desarrolladores definir cómo el contenido interactuará sobre las páginas web. CSP es un conjunto de reglas que tiene como finalidad

reading several documents downloaded from a set of URLs, on each document a transformation function is used with 53 sensors responsible for taking several characteristics of each document (HTML elements, external links, among others) and to generate a numeric vector. Combining the results of each vector a learning sequence is created, which is used by GP though the selection, reproduction and fitness function. Additionally, the fitness function includes as variables both the false positivies rate and the false negatives rate, and also the learning sequence. The experiment had a dataset that contains information from approximately 720 readings of 15 pages of dynamic content. Each reading was performed with a frecuency of 6 hours during a month. On the other hand, in monitoring phase each document is reviewed and assigned a value that represents whether the page is benign or malicious. Finally, the results confirm that the algorithms generated by the GP have a good performance in the detection of anomalies, with a low rate of both false positives and false negatives.

DICE is a system for automatic detection of websites with defacement (Tanaka, Kai, Tamura, & Sasaki, 2011). The system incorporates a method to detect unauthorized scripts (SQL injections). Its method uses the quantification theory type II of Professor Chikio Hayashi for the analysis and discrimination of the data through the following parameters: alteration of the title, multiple insertions of scripts, a set of unauthorized URLs or and a list of states in a script. The system was evaluated through its implementation in a free access proxy server "Squid", 121 URLs malign and 229 beningn URLs were used, and of each type 60 were selected randomly for the adjustment of the parameters belong to the experiment. Subsequently, it was obtained a detection rate of 84% and 16% for false positives. Finally, the authors propose the exploration of new features to prevent other attacks besides the SQL injections.

Tripwire is a free access tool for UNIX environments that facilitates the security and integrity of the file directory information (Kim & Spafford, 1994). The solution has the ability to notify administrators if it has found changes in the files in order to take corrective actions at a significant time. Tripwire uses the signature method; each of the files is evaluated to abstract some values (size, date of the last modification and owner) and then a signature is generated, which will be stored in a check list. Subsequently, the system periodically checks the files against the list to look for inconsistencies. On the other hand, Fujimura and Jin (2007) include the implementation of a system for files interpolation. In the project a

modification on Tripwire was made and as value added the solution has the capacity to be used by both administrators and system ordinary users.

In order to analyze the vulnerability growth of web applications, Stamm et al., (2015) propose a content restriction scheme called Content Security Policy [CSP] as a solution. Content restrictions are a mechanism that allows developers to define how the content can interact over websites. CSPs are a set of rules whose pretends to monitor, detect and alert when an HTTP request presents vulnerabilities on the web page. A prototype was developed in both the client and the server for the CSP implementation, also a modification was made on the Firefox web browser and the CSP restrictions on the server were added.

IPVmon is a commercial solution that allows for finding anomalies into the websites (IPVmon, 2014). By way of a series of sensors, the system detects malicious activities like defacement attacks, DDoS attacks, malicious software injection and theft of domains. When IPVmon detects some inconsistency, it generates a notification via SMS message or email.

StatusCake provides a monitoring service to websites and as part of its services is the generation of alerts about low performance and attacks (Barnes, 2013).

Nagios is a monitoring system that allows companies to identify and solve problems that affect the IT infrastructure (Aman et al., 2014). The solution is integrated by several projects, such as commercial and open source projects. On the other hand, the system has a monitor function in order to detect any anomalies; if the function detects some anomaly then the system sends some alerts to the system administrators.

Shahriar and Zulkernine (2009) propose a set of 11 operators to test XSS vulnerabilities, five of them are elaborated to JavaScript and the remaining six are set to PHP code. Operators are part of a mutation technique for XSS Vulnerability testing (XSSV). It was developed a prototype called MUTEC (Mutation-based Testing of Cross Site Scripting Vulnerabilities) in order to evaluate the technique. MUTEC is an automatic tool that generates XSSV mutants to the analysis process with the following operators: ADES adds calls to the escape function, RESC removes calls to the escape function, RWWE replaces calls to the write function with calls to the eval function, RIHA replaces the innerHTML property with the text node, MARF modifies the arguments of the calls to the replace function, AHSC adds calls to the htmlspecialchars function, RHSC removes calls to the htmlspecialchars function, AHEN adds Calls to the htmlentities function, monitorear, detectar y alertar si en una petición HTTP se presenta alguna vulnerabilidad sobre la página web. Para la implementación de los CSP se desarrolló un prototipo, tanto en el cliente, como en el servidor, para lo que se realizó una modificación sobre el navegador web Firefox y se agregaron las restricciones CSP sobre el servidor.

IPVmon es una solución comercial que permite una búsqueda de anomalías a los sitios web (IPVmon, 2014). A través de una serie de sensores el sistema detecta actividades maliciosas, como ataques de *defacement*, ataques de DDoS, inyección de software malicioso y robo de dominios, si detecta alguna inconsistencia, provoca una notificación a través de un mensaje SMS o in correo electrónico.

StatusCake provee un servicio de monitoreo a páginas web e incluye, como parte de sus servicios, la generación de alertas sobre el bajo desempeño y los ataques (Barnes, 2013).

Nagios es un sistema de monitoreo que permite identificar y resolver problemas que afectan a la infraestructura de TI (Aman et al., 2014). La solución se encuentra integrada por varios proyectos, tanto comerciales, como de código abierto. Por otra parte, el sistema permite el monitoreo y la generación de alertas en los casos en que se llegase a detectar alguna modificación o inserción maliciosa a una página web.

Shahriar y Zulkernine (2009) proponen un conjunto de once operadores para *testear* vulnerabilidades de XSS, cinco de ellos son para JavaScript y los seis restantes para código PHP. Los operadores hacen parte de una técnica de mutación para pruebas de vulnerabilidades de XSS (XSSV). Para la evaluación de la técnica se desarrolló un prototipo llamado MUTEC (*mutation-based testing of cross site scripting vulnerabilities*), una herramienta automática que genera mutantes de XSSV para el proceso de análisis y hace uso de los siguientes operadores: ADES, que agrega llamadas a la función *escape*; RESC, que remueve las llamadas a la función *escape*; RWWE, que reemplaza las llamadas a la función *write* con las llamadas a la función *eval*; RIHA, que reemplaza la propiedad *innerHTML* con el nodo de texto; MARF, que modifica los argumentos de las llamadas a la función *replace*; AHSC, que añade llamadas a la función *htmlspecialchars*; RHSC, que elimina las llamadas a la función *htmlspecialchars*; AHEN, que adiciona llamadas a la función *htmlentities*; RHEN, que remueve las llamadas a la función *htmlentities*; MALT, que modifica el parámetro tag en las llamadas a las funciones *striptags*; y RSTT, que elimina las llamadas a la función *striptags*. En el experimento se evaluaron cinco aplicaciones de libre acceso que fueron reportadas con XSSV, como resultado encontraron que los operadores son efectivos para un adecuado *testeo*.

Hollander (2000) propone unas capas para la protección de los ataques que pueden generar un web *defacement*. Estas constan de:

- On-the-spot prevention: el ataque puede ser identificado durante la solicitud de un servicio a través de una rutina de intercepción transparente, como por ejemplo durante las llamadas del sistema o la invocación de una API. A través de la rutina se podría determinar la transferencia de solicitudes autorizadas para el sistema.
- Administrator (root) resistant: como parte de las buenas prácticas en la seguridad de los servidores web, es recomendable restringir los privilegios a las cuentas de los administradores.
- Application access control: un programa predefinido con un único acceso a la modificación de la página web.
- OS- level protection: se recomienda que la solución este en la capacidad de identificar y prevenir la explotación a las vulnerabilidades de ataques al sistema operativo.
- HTTP attack protection: debido a que la gran mayoría de ataques a las páginas web son a través de protocolo HTTP, se recomienda escáner cada petición para detección de solicitudes maliciosas.
- Web server resources protection: es necesario proteger los recursos más importantes del sistema, entre estos se pueden incluir: ejecutables, archivos de configuración, archivos de información y procesos del servidor web.

Borgolte, Kruegel y Vigna (2015) introducen un sistema para detección de *defacement* llamado Meerkat. La propuesta utiliza una detección de anomalías a partir de distintas características obtenidas de una captura de la representación visual de la página web; con la información obtenida se creó un perfil para el monitoreo y detección de cambios no legítimos sobre la página web. Fue una motivación para el estudio el empleo de técnicas visuales para la evaluación de anomalías debido a que los métodos basados en análisis de código fuente o contenido alteran usualmente archivos JavaScript y Cascading Style Sheets [CSS] para la generación del *defacement*, estos cambios usualmente no contienen mucho contenido textual a diferencia de las imágenes con texto (Haq et al., 2015; Sommer & Paxson, 2010; Barreno, Nelson, Joseph, & Tygar, 2010). Meerkat emplea en su clasificación una red neuronal de aprendizaje profundo, en el proceso de aprendizaje se utilizaron capturas de pantalla de 160x160 y 1600x900 pixeles a partir de una estrategia de extracción de páginas web maliciosas y benignas. El aprendizaje fue de tipo no supervisado con *autoencoders*, se integró un *dataset* de 925.817 casos de *defacement* de Zone-H y 255.490 páginas web legítimas. Una vez entrenada la red neuronal, los resultados de la detección de contenidos con *defacement* fueron los siguientes: la tasa de verdaderos positivos fue entre 97.422% y 98.816%; la tasa de falsos positivos entre 0.547% y 1.528%; y la detección Bayesiana entre 98.583% y 99.845%. Es importante resaltar que este sistema solo necesitó de un pequeño espacio de la pantalla

RHEN removes the calls to the htmlentities function, MALT modifies the tag parameter in the calls to the striptags function and RSTT removes the calls to the striptags function. In order to the experiment, five free access applications reported with XSSV were evaluated. Finally, as a result, it was found that operators are effective for proper testing.

Hollander (2000) proposes layers for the attacks protection that can generate web defacement. These are described below:

- On-the-spot prevention: The attack can be identified during the request of a service through transparent interception routine, for instance it is generated during system calls or an API invocation. Also, through the routine it could be determined the transfer of authorized requests for the system.
- Administrator (root) resistant: As part of the best practices in web server security, it is advisable to restrict privileges to administrators' accounts.
- Application access control: A predefined program with a single access to the modification of the website.
- OS-level protection: It is recommended that the solution may to identify and prevent exploitation to vulnerabilities of attacks on the operating system.
- HTTP attack protection: Owing to the vast majority of attacks on websites have taken place through HTTP protocol, it is recommended to scan every request for detection of malicious requests.
- Web server resources protection: It is necessary to protect the most important resources of the system, these may include: executables, configuration files, information files and Web server processes.

Borgolte, Kruegel and Vigna (2015) introduce a defacement detection system called Meerkat. The proposal uses anomalies detection with several characteristics from the capture of the website visual representation. Taking this into account, a profile was created to monitor and detection of non legitimate changes into the web page. The use of visual techniques (Haq et al., 2015; Sommer & Paxson, 2010; Barreno, Nelson, Joseph, & Tygar, 2010) were considered as a motivation for the evaluation of anomalies, due to methods based on source or content analysis usually alter JavaScript and Cascading Style Sheets [CSS] files for the generation of defacement, the study explains that these changes usually do not contain much textual content unlike the images with text. Meerkat uses a neural network of deep learning in its classification. From a strategy of extraction of mali-

cious and benign websites were used screen captures of 160 x 160 and 1600 x 900 pixels during the process of learning. It was of not monitored type with autoencoders, a dataset of 925,817 cases of Zone-H defacement and 255,490 legitimate websites was integrated. On the other hand, once the neural network was trained the results of the detection of contents with defacement were as follows: the true positive rate was between 97.422% - 98.816%, a false positive rate between 0.547% - 1.528%, and the Bayesian detection was between 98.583% - 99.845%. Finally, it is important to emphasize that this system only requires a small space of the screen to do the defacement detection and with the purpose of avoiding evasion of the system Meerkat employs different techniques, such as desertion and set-up.

Lui and Cinquini (2012) patent a mechanism for the protection of websites against defacement. Its solution proposes an association of the website content for a single a user with a digital signature. A Content Delivery Network (CDN) is a network superimposed of computers that allows origins (owners) of the websites to hide from the cybercriminals. In architecture, the CDN cache server uses an algorithm to validate the website's content against the digital signature (which can be internal or external) before the content is delivered to the end user, offering a mechanism that allows the companies to prevent unauthorized content distribution. The validation algorithm uses a part of the web content to construct a digital signature in order to store it in a repository. Finally, once an inconsistency between the content and the signatures is detected, the system proceeds to generate alerts to the administrator or to the persons responsible for security.

Kim, Lee, Park, & Kim (2006) stand both, a remote mechanism for detection of defacement on dynamic content websites and a threshold adjustment method for the reduction of false positives. The detection mechanism uses one vector for each website through a 2-gram frequency index, which allows it to compare the similarity between the current threshold and the previous capture. Thresholds were adjusted with two methods to reduce the number of false positives. In the first method the weighted average of the threshold was calculated with two daily catches, and in the second one, it was applied a calculation over the difference of the threshold between the current and the old catch. There were used around 185 operating websites for experiments purposes from Korea and two system executions (one for the threshold with adjustment method and other with a static threshold). Finally, results show how the system with static threshold describes an increase on false positives in contrast to the one which applies threshold adjustments.

para hacer la detección del *defacement*, para evitar la evasión del sistema Meerkat, él emplea diversas técnicas, tales como la deserción y la puesta a punto.

Lui y Cinquini (2012) patentaron un mecanismo para la protección de las páginas web contra el *defacement*. Su solución propone la asociación del contenido de la página web para un usuario contra una firma digital. Un *Content Delivery Network* [CDN] es una red superpuesta de computadoras que permite esconder el origen (propietario) del contenido de una página web a los cibercriminales. En la arquitectura, el servidor cache de CDN utiliza un algoritmo que permite validar el contenido de una página web contra la firma digital (que puede ser interna o externa) antes de que el contenido sea entregado a los usuarios finales, ofreciendo así un mecanismo que permite evitar la difusión de contenido no autorizado. El algoritmo de validación utiliza una porción del contenido de la web para construir una firma digital que será luego almacenada a un repositorio. Finalmente, una vez detectada una inconsistencia entre el contenido y las firmas, el sistema procede a generar alertas al administrador o a las personas encargadas de la seguridad.

Kim, Lee, Park, y Kim, (2006) proponen un mecanismo remoto para detección de *defacement* en páginas web de contenido dinámico y un método de ajuste de umbral para la reducción de falsos positivos. El mecanismo de detección utilizó un vector por cada página web a través de un índice de frecuencia de 2-*gram* que permite comparar la similitud entre el actual umbral y el de una captura previa. Los umbrales fueron ajustados con dos métodos para reducir la cantidad de falsos positivos, en el primer método se calculó la media pondera del umbral con dos capturas diarias, y en la segunda se aplicó a cada proceso de detección un cálculo de la diferencia del umbral entre la captura actual y la captura antigua. Para el experimento se utilizaron 185 páginas web de operación en Corea y dos ejecuciones del sistema, una para el umbral con método de ajuste, la otra con un umbral estático. Sus resultados muestran cómo el sistema con umbral estático presenta una mayor cantidad de falsos positivos a diferencia del que implementa ajustes en umbral.

Shani (2008) propuso un sistema y un método para la identificación, prevención y gestión de los sitios que han sido afectados por un *defacement*. El sistema permite la identificación de las partes que sufren el *defacement* mediante la detección de anomalías sobre el código fuente. El proceso de monitoreo y detección utiliza las siguientes actividades: codificación, almacenamiento, comparación del código fuente de la página web en la base de datos –información de código fuente y ejemplos de *defacement*–, identificación de los parámetros del ataque, cálculo de la probabilidad del *defacement* y decisión sobre el estado de la página web. Una vez que se identifica una página en estas condiciones, el sistema puede permitir la eliminación del *defacement* y su sustitución por la última versión de la página web.

Por último, Alhamed y Alsuhaibany (2013) describen un sistema de manejo de incidentes de *defacement* en páginas web a partir de la detección de anomalías en la información y las capturas de pantalla de las páginas web. La solución utiliza múltiples factores para la determinación del *web defacement*, entre ellas: la comparación de las capturas de pantalla, la relación entre los textos de la página contra una lista de sitios web con *defacement*, la determinación de los sitios ilegítimos a través de las referencias por URL, y la comparación de las propiedades de los archivos del sitio web. Adicionalmente, para cada uno de los factores mencionados se calcula una puntuación de probabilidad que indica si la página presenta un *defacement*; si el sitio se encuentra alterado, el sistema cuenta con la posibilidad de realizar la restauración del contenido y el envío de alertas.

V. Conclusiones

Los estudios demuestran que es importante conocer sobre la seguridad de la computación durante el desarrollo de las soluciones tecnológicas, debido que este sería un medio que podría mitigar los ataques generados por los cibercriminales.

A través de la investigación del estado del arte se encontraron diversas propuestas que permiten reducir y controlar los ataques que provocan el *web defacement*. Gran parte de los trabajos se han enfocado a dar solución al problema a páginas de alto contenido dinámico debido al aumento de este tipo de sistemas. Por otra parte, un camino prometedor es la aplicación de las técnicas de inteligencia artificial, identificación de patrones, el *Big Data*, y el análisis sobre el tráfico de la red que permita identificar el ataque y su tipo antes de generarse el *defacement* sobre la página web.

Los trabajos enfocados en *machine learning* sugieren la importancia de la veracidad de los *datasets*, su cantidad y variabilidad; entre los estudios se pueden encontrar aplicaciones bajo el enfoque de la detección de anomalías. Adicionalmente, con el análisis estático o dinámico se pueden extraer las características que servirán para el entrenamiento del algoritmo que predecirá si se ha presentado un ataque a la aplicación web; los estudios proponen distintas características, como por ejemplo: capturas de pantalla, utilización del código fuente e información del tráfico de la red, entre otras.

Se pueden encontrar soluciones que proponen instalación sobre la infraestructura y otras que ofrecen un modelo como servicio externo, ambas propuestas tienen sus ventajas y desventajas, pero la tendencia es hacía un modelo como servicio de monitoreo, en el cual se le ofrece un medio externo de administración al *webmaster* y la generación de alertas por detección de anomalías.

Agradecimientos

Muchas gracias a COLCIENCIAS y a la Universidad ICESI por sus invaluables contribuciones, a Jose Ignacio Claros y John Edinson Martinez por sus comentarios, a Oscar Mondragon y Andrés Mera por las contribuciones realizadas durante el proyecto, y a todas las personas que han aportado de su tiempo para la realización de este artículo. S&T

On the other hand, Shani (2008) proposed a system and a method for the identification, prevention and management of sites that have been affected by defacement. The system identifies the parts that suffer the defacement by detecting anomalies on the source code. The monitoring and detection process uses the following activities: coding, storage, source code of the web page in the database (source code information and defacement examples) comparison, parameters of the attack identification, probability of the defacement calculation and decision on the state of the website. Once a page is identified under these conditions, the system can remove the defacement and then it is replacement by the latest version of the website.

Finally, Alhamed and Alsuhaibany (2013) describe an incident management system to control the defacement in websites from the detection of anomalies in the information and screenshots of websites. The solution uses multiple factors for the determination of the web defacement, among of them are: comparison of the screen captures, relation between the texts of the page against a list of websites with defacement, determine the illegitimate sites through references by URL and comparison of the website files properties. Additionally, for each of the factors mentioned above, a probability score is calculated to indicate if the page has defacement. If the site is altered, the system will perform content restoration and send alerts.

V. Conclusions

Studies demonstrate that is important to know about computer security during the technologies development, due to it could be a way to mitigate the attacks generated by cybercriminals.

Following the state of the art research several proposals was founded, which leads to reduce and control the web defacement attacks. Most of the researchs have been focused on solving high dynamic content pages, due to the increase of these systems. Nonetheless, a potential way is the artificial intelligence techniques application, patterns identification, the Big Data, and the analysis on the network traffic that allows identifying the attack and its type before the defacement on the website.

Researches focused on machine learning suggest the importance of the datasets veracity, quantity and variability; besides applications under the approach of the detection of anomalies can be found in the studies. In addition, static or dynamic analysis can extract the characteristics that will be used for the algorithm training which leads to predict if has

occurred an attack to the web application; the studies propose several characteristics, such as screen captures, source code usage, and network traffic information, among others.

Finally, it is possible to find solutions that propose installation on the infrastructure and others that may offer a model as an external service, both proposals have their advantages and disadvantages, however the trend is to have a model as a monitoring service, which offers a webmaster external administration and alerts generation by detection of anomalies.

Acknowledgments

In gratitude to COLCIENCIAS and Universidad ICESI for their invaluable contributions, Jose Ignacio and John Edinson Martinez for their comments, Oscar Mondragon and Andrés Mera for their contributions developed in this project, and finally thanks to all the people who have actively participated during this research development. ST

References / Referencias

Alhamed, M., & Alsuhaibany, O. M. (2013). *U.S. Patent No. 8,549,637.* Washington, DC: U.S. Patent and Trademark Office.

Aman, H., Yamashita, A., Sasaki, T., & Kawahara, M. (2014, August). Multistage growth model for code change events in open source software development: An example using development of Nagios. In: *Software Engineering and Advanced Applications (SEAA), 2014 40th EUROMICRO Conference on* (pp. 207-212). IEEE.

Amanatidis, T., & Chatzigeorgiou, A. (2016). Studying the evolution of PHP web applications. *Information and Software Technology,* 72, 48-67.

Barnes, J. (2013, February 18). *Free real user monitoring* [StatusCake]. Retrieved from: https://www.statuscake.com/free-real-user-monitoring/#

Barreno, M., Nelson, B., Joseph, A. D., & Tygar, J. D. (2010). The security of machine learning. *Machine Learning, 81*(2), 121-148.

Bartoli, A., Davanzo, G., & Medvet, E. (2009). The reaction time to web site defacements. *IEEE Internet Computing, 13*(4), 52-58.

Bartoli, A., Davanzo, G., & Medvet, E. (2010). A framework for large-scale detection of website defacements. *ACM Transactions on Internet Technology* (TOIT), 10(3), 10.

Batyuk, L., Herpich, M., Camtepe, S. A., Raddatz, K., Schmidt, A. D., & Albayrak, S. (2011, October). Using static analysis for automatic assessment and mitigation of unwanted and malicious activities within Android applications. In: *Malicious and Unwanted Software (MALWARE), 2011 6th International Conference on* (pp. 66-72). IEEE.

Borgolte, K., Kruegel, C., & Vigna, G. (2015). Meerkat: Detecting website defacements through image-based object recognition. In: *24th USENIX Security Symposium (USENIX Security 15)* (pp. 595-610).

Caswell, B., Beale, J., & Baker, A. (2007). *Snort intrusion detection and prevention toolkit.* Syngress.

Cerf, V. G., & Quaynor, N. (2014). The Internet of Everyone. *Internet Computing, IEEE, 18*(3), 96-96.

Dalai, A. K., & Jena, S. K. (2011, February). Evaluation of web application security risks and secure design patterns. In: *Proceedings of the 2011 International Conference on Communication, Computing & Security* (pp. 565-568). ACM.

Davanzo, G., Medvet, E., & Bartoli, A. (2011). Anomaly detection techniques for a web defacement monitoring service. *Expert Systems with Applications, 38*(10), 12521-12530.

Diakopoulos, N., & Cass, S. (2015). *Interactive: The top programming languages 2015. IEEE Spectrum, online, July, 20.* Retrieved from: http://spectrum.ieee.org/static/interactive-the-top-programming-languages-2015

Eshete, B., Villafiorita, A., & Weldemariam, K. (2011, July). Malicious website detection: Effectiveness and efficiency issues. In: *SysSec Workshop (SysSec),* 2011 First (pp. 123-126). IEEE.

Fujimura, N., & Mei, J. (2007, October). Implementation of file interpolation detection system. In: *Proceedings of the 35th annual ACM SIGUCCS fall conference* (pp. 118-121). ACM.

Futoransky, A., Gutesman, E., & Waissbein, A. (2007). A dynamic technique for enhancing the security and privacy of web applications. In: *Proc. Black Hat USA.*

Gross, G. (2015, June). *US Army website defaced, then brought down.* Retrieved from: http://www.pcworld.com/article/2932936/us-army-website-defaced-then-brought-down.html

Gupta, S., & Gupta, B. B. (2015, May). PHP-sensor: a prototype method to discover workflow violation and XSS vulnerabilities in PHP web applications. In: *Proceedings of the 12th ACM International Conference on Computing Frontiers* (p. 59). ACM.

Haq, N. F., Onik, A. R., Hridoy, M. A. K., Rafni, M., Shah, F. M., & Farid, D. M. (2015). Application of Machine Learning Approaches in Intrusion Detection System: A Survey. *IJARAI- International Journal of Advanced Research in Artificial Intelligence, 4*(3), 9-18.

Harper, A., Harris, S., Ness, J., Eagle, C., Lenkey, G., & Williams, T. (2015). *Gray hat hacking the ethical hackers handbook.* McGraw-Hill Osborne Media.

Hollander, Y. (2000). Prevent web site defacement. *Internet Security Advisor, 3*(6), 22.

Howard, G. M., Gutierrez, C. N., Arshad, F. A., Bagchi, S., & Qi, Y. (2014, June). pSigene: Webcrawling to Generalize SQL Injection Signatures. In: *Dependable Systems and Networks (DSN), 2014 44th Annual IEEE/IFIP International Conference on* (pp. 45-56). IEEE.

IPVTec (2014). *What´s IPVmon?* Retrieved from: http://www.ipvtec.com/whats-ipvmon/

Jericho & Munge. (2000). *Hard-core web defacement statistics trends and analysis.* In: Black Hat USA 2000. Retrieved from: https://www.blackhat.com/html/bh-usa-00/bh-usa-00-speakers.html#JerichoPunkis

Jingling, Z., & Rulin, G. (2015, July). A New Framework of Security Vulnerabilities Detection in PHP Web Application. In: *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2015 9th International Conference on* (pp. 271-276). IEEE.

Kim, G. H., & Spafford, E. H. (1994, November). The design and implementation of tripwire: A file system integrity checker. In: *Proceedings of the 2nd ACM Conference on Computer and Communications Security* (pp. 18-29). ACM.

Kim, W., Lee, J., Park, E., & Kim, S. (2006). Advanced mechanism for reducing false alarm rate in web page defacement detection. In: *The 7th International Workshop on Information Security Applications.*

Kumar, M. (2015, May). *Gaana.com Hacked, 10 Million User´s Details Exposed.* Retrieved from: http://thehackernews.com/2015/05/gaanacom-hacked-10-million-users.html

Lui, Z. & Cinquini, M. J. (2012). *Web content defacement protection system* [U.S. Patent No. 8,145,908]. Washington, DC: U.S. Patent and Trademark Office.

Medvet, E., Fillon, C., & Bartoli, A. (2007, August). Detection of web defacements by means of genetic programming. In: *Information Assurance and Security, 2007. IAS 2007. Third International Symposium on* (pp. 227-234). IEEE.

Mohaisen, A. (2015, November). Towards automatic and lightweight detection and classification of malicious web contents. In: *Hot Topics in Web Systems and Technologies (HotWeb), 2015 Third IEEE Workshop on* (pp. 67-72). IEEE.

Muñoz, F. R., & Villalba, L. G. (2012). Preproceso de formularios para el análisis de seguridad de las aplicaciones web. *Actas de la XII Reunión Española sobre Criptología y Seguridad de la Información (RECSI 2012), Donostia-San Sebastián, España.*

Nguyen-Tuong, A., Guarnieri, S., Greene, D., Shirley, J., & Evans, D. (2005, May). Automatically hardening web applications using precise tainting. In: *IFIP International Information Security Conference* (pp. 295-307). Springer.

Open Web Application Security Project [OWASP]. (2013). OWASP Top 10 - 2013 *The ten most critical web application security risks.* Retrieved from: https://www.owasp.org/index.php/Top_10_2013-Top_10

Open Web Application Security Project [OWASP]. (2016). *PHP Top 5.* Retrieved from: https://www.owasp.org/index.php/PHP_Top_5

Roesch, M. (1999, November). Snort: Lightweight intrusion detection for networks. In *LISA, 99*(1), 229-238.

Shahriar, H., & Zulkernine, M. (2009, May). Mutec: Mutation-based testing of cross site scripting. *In Proceedings of the 2009 ICSE Workshop on Software Engineering for Secure Systems* (pp. 47-53). IEEE Computer Society.

Shani, O. (2008). *System and method for identification, prevention and management of web-sites defacement attacks* [Patent Application No. 12/531,728]. . Washington, DC: U.S. Patent and Trademark Office.

Socuri [Web site] (2016). Retrieved from: https://sucuri.net/?clickid=QszQyrVcJ2HBV35ytHQRK1hvUkSUeXwqU0SxXQ0

Sommer, R., & Paxson, V. (2010, May). Outside the closed world: On using machine learning for network intrusion detection. In: *2010 IEEE symposium on security and privacy* (pp. 305-316). IEEE.

Son, S., & Shmatikov, V. (2011, June). SAFERPHP: Finding semantic vulnerabilities in PHP applications. In: *Proceedings of the ACM SIGPLAN 6th Workshop on Programming Languages and Analysis for Security* (p. 8). ACM.

Stamm, S., Sterne, B., & Markham, G. (2010, April). Reining in the web with content security policy. In: *Proceedings of the 19th international conference on World Wide Web* (pp. 921-930). ACM.

Stuttard, D. & Pinto, M. (2011). *The web application hacker's handbook: finding and exploiting security flaws.* Hoboken, NJ: John Wiley & Sons.

Tanaka, T., Kai, T., Tamura, Y., & Sasaki, R. (2011, October). Development and evaluation of defaced sites automatic detection system DICE. In: *Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2011 Seventh International Conference on* (pp. 196-201). IEEE.

Ullrich, J. B., & Lam, J. (2008). Defacing websites via SQL injection. *Network Security, 2008*(1), 9-10.

Urcuqui, C., & Navarro, A. (2016, April). Machine learning classifiers for android malware analysis. In: *Communications and Computing (COLCOM), 2016 IEEE Colombian Conference on* (pp. 1-6). IEEE.

Vanderaj. (2016). *The open web application security project, PHP Top 5*. Retrieved from: https://www.owasp.org/index.php/PHP_Top_5#P3:_SQL_Injection

Wei, W. (2015, November). Rise in website defacement attacks by hackers around the world. Retrieved from: http://thehackernews.com/2013/11/rise-in-website-defacement-attacks-by.html

WhiteHat Security. (2016). *Web applications security statistics report 2016.* Retrieved from: https://www.whitehatsec.com/info/website-stats-report-2016-wp/

Xie, Y. & Aiken, A. (2006, July). Static detection of security vulnerabilities in scripting languages. In: *USENIX Security, 6,* 179-192.

Zhong, Y., Asakura, H., Takakura, H., & Oshima, Y. (2015, July). Detecting malicious inputs of web application parameters using character class sequences. *In Computer Software and Applications Conference (COMPSAC), 2015 IEEE 39th Annual* (Vol. 2, pp. 525-532). IEEE.

Zone-H [website]. (2016). Retrieved from: http://www.zone-h.org

CURRICULUM VITAE

Christian Camilo Urcuqui Systems Engineer (emphasis in Management and Computing) and Master in Informatics and Telecommunications from Universidad Icesi (Cali-Colombia). Member of Informatics and Telecommunications research group [i2t]. His areas of interest include: artificial intelligence, machine learning and security applied to informatics / Ingeniero de Sistemas con énfasis en Administración e Informática y Máster en Informática y Telecomunicaciones de la Universidad Icesi (Cali-Colombia). Miembro del grupo de investigación en Informática y Telecomunicaciones [i2t] de la Universidad Icesi. Sus áreas de interés incluyen: inteligencia artificial, aprendizaje de máquina, y seguridad informática.

Melisa García Peña Systems Engineering student at the Universidad Icesi (Cali-Colombia); she participates of Informatics and Telecommunications (i2t) research group activities / Estudiante de Ingeniería de Sistemas de la Universidad Icesi (Cali-Colombia), colabora con el grupo de investigación en Informática y Telecomunicaciones (i2t).

José Luis Osorio Quintero Systems Engineering student at the Universidad Icesi (Cali-Colombia). He participates of Informatics and Telecommunications (i2t) research group activities / Estudiante de Ingeniería de Sistemas de la Universidad Icesi (Cali-Colombia), colabora con el grupo de investigación en Informática y Telecomunicaciones (i2t).

Andrés Navarro Cadavid Full professor and Director of i2t (Informatics and Telecommunications research group) at the Universidad Icesi (Cali, Colombia). Electronics Engineer and Master in Technology Management (Universidad Pontificia Bolivariana de Medellín (Colombia), and Ph.D. in Telecommunications (Universidad Politécnica de Valencia, España). His main areas of interest are: spectrum management, radio propagation and m-health / Profesor titular y Director del Grupo de Investigación en Informática y Telecomunicaciones (i2T) de la Universidad Icesi de Cali (Colombia). Es Ingeniero Electrónico y Magister en Gestión de la Tecnología de la Universidad Pontificia Bolivariana de Medellín (Colombia) y Doctor Ingeniero en Telecomunicaciones de la Universidad Politécnica de Valencia (España). Sus áreas de interés son: la gestión del espectro radioeléctrico, la radio-propagación y las soluciones móviles aplicadas a salud.