Cómo controlar 40 páginas web en 7 minutos por Georgios Konstantopoulos

Esta oportunidad es para traerles en castellano el magnífico relato de Georgios Konstantopoulos, empleado de la empresa Honda en Alemania, ingenireor y ahora estudiante y entusiasta de las tecnologías informáticas. Como cosa extraña el artículo está en idioma inglés y no en alemán, ya sabemos que el inglés ahora es como el latín del Imperio Romano: una lengua franca, nos guste o no. La cuestión es que hablamos castellano y hay que extender el conocimiento y así completamos el círculo: por eso es que no está escrito en alemán (ni tampoco en griego, idioma natal del sr. Konstantopoulos). **En este punto ya queda completamente claro que no es de nuestra autoría el siguiente artículo, que es una traducción aderezeda *así como un merecido reconocimiento a las habilidades y obra del sr. Georgios ?¡a su salud!?***

Introducción.

El diccionario de la Real Academia Española, lamentablemente, definen a la palabra "*hacker*" como pirata informático, asociándolo con lo que comúnmente conocemos como copia ilegal de software privativo. Debemos hacer hincapié en que una cosa es "**to hack**" y otra cosa muy diferente es "**to crack**": la primera busca conseguir las vulnerabilidades y de paso aprender con el intento, la otra es lo que dice la RAE: piratear, tomar por la fuerza lo que sea útil *¡e incluso hundir el barco con todo y tripulación!* Así la Reina de Inglaterra le ponga el título de "*Sir*", **pirata por los siglos de los siglos se queda**:

https://www.youtube.com/watch?v=4TI-VAa5UFI

Por eso aclaramos que la tarea de un **"hacker"** es la búsqueda del conocimiento en beneficio de la sociedad y la de un **"cracker"** pues es la del beneficio propio *¡menuda diferencia!*

Os dejamos entonces, traducido al castellano llano, el buen artículo del día de hoy que versa sobre seguridad en ordenadores:

«Cómo "jaquié" 40 sitios web en 7 minutos» por Georgios Konstantopoulos.

El verano pasado comenzé a aprender acerca de la seguridad en la información y el "*hacking*". En el último año he participado en varios juegos de guerra, capturas de banderas y simulaciones de penetraciones o sus intentos y de manera continua he ido mejorando mis habilidades de "hacking": aprendiendo cosas nuevas acerca de "*como hacer que las computadoras se desvien del*

comportamiento esperado".

Abreviando la larga historia, mi experiencia siempre estuvo limitada a entornos simulados, y debido a que me considero un "*hacker*" de sombrero blanco -mejor conocido como uno de los tipos buenos- (*sic*, es una redundancia, N. del T.) nunca metí mis narices en los asuntos de otros -literalmente hablando-.

Hasta ahora. Esta será una historia detallada acerca de como "*jaquié*" un servidor el cual albergaba 40 (este es un número exacto) de sitios web y mis hallazgos.

> Nota: Se necesitan, como prerequisitos, conocimientos previos en ciencias de la computación para hilvanar la partes técnicas de este artículo.

Tiempo de sacar las grandes armas.

Un amigo me escribió un mensaje de que una vulnerabilidad de guiones de sitios cruzados "XSS" fue encontrada en su sitio web y él quería que yo emitiera una segunda opinión. Este es un paso importante, así que me incliné por preguntarle formalmente y de manera expresa si tenía su permiso y consentimiento de realizarle una prueba completa a su aplicación web y sobre el servidor que lo alojaba. **La respuesta fue positiva.**

> Para el resto del artículo yo me referiré siempre al dominio de mi amigo como "http://www.example.com" ("ejemplo.com" N.del T.)

El primer movimiento es siempre enumerar y encontrar la mayor cantidad de información que puedas acerca de tu "enemigo," siempre tratando de alarmarlos lo menos posible.

En este punto activamos nuestro cronómetro y comenzamos a explorar:

¡Es un montó de puertos abiertos! Por simple observación de la apertura de los puertos 21 (FTP) y los puertos 139 y 445 podemos imaginar que es usado para almacenamiento y compartición de ficheros, además de ser un servidor web (puertos 8 y 443 y los sustitutos 8080 y 8081).

Conócete a tí mismo y a tu "enemigo".

Conócete a tí mismo.
Conoce a tu enemigo.
En cientos de batallas,
cientos de victorias.
Sun Tzu, "El Arte de la Guerra".

Hacer una exploración a los puertos por vía UDP más allá de los mil primeros es una opción a considerar si no es suficiente la información recabada. Los únicos puertos donde estaba permitido interactuar (y sin credenciales) eran los puertos 80 y 443.

Sin desperdiciar tiempo en esto último, lanzo el programa "gobuster**"** para listar algún archivo interesante en el servidor web mientras yo indagaba por cuenta propia de manera manual.

Resulta que la vía /admin es una herramienta administrativa la cual permite autenticar a los usuarios para modificar parámetros en el servidor web. Son requeridas las credenciales para entrar y ya que no tengo ni un nombre de usuario y mucho menos una contraseña, voy al siguiente paso (soy aguafiestas: **"gobuster"** no encontró nada de valor alguno).

Tercer minuto.

> Hasta ahora 3 minutos en esto. Nada útil, aún.

Navegando por el sitio web observamos que nos invita para hacer conexión. No hay problema, creamos una cuenta con un correo electrónico falso en la página web dummy e-mail, hacemos click en el mensaje de confirmación e ingresamos luego de pocos segundos.

El sitio web nos da la bienvenida y nos pregunta si deseamos ir a nuestro perfil de usuario y actualizar nuestra imagen de usuario. Que amables.

Viendo que el sitio web parece de código personalizado, me inclino por probar una técnica conocida como vulnerabilidad de subida irrestricta de fichero. En mi terminal ejecuto lo siguiente y lo guardo en un fichero PHP:

```
echo "" > vulnerabilidad.php
```

Intento subir la "imagen" y ***¡canté bingo!*** La rutina de subida de archivos permitió cargar al fichero **vulnerabilidad.php** . Desde luego que no hay imagen en miniatura , pero eso significa que mi archivo subió a alguna parte.

imagen

Aquí pudieramos esperar que la rutina subidora hiciera una suerte de proceso al fichero cargado, revisar la extensión y reemplazarla con las extensiones permitidas tales como **.jpg** o **.jpeg** para evitar la ejecución remota por parte de un atacante que suba código malicioso, tal como el nuestro, precisamente.

La gente se preocupa por la seguridad, después de todo.

¿Cierto? ¿CIERTO? ***¿CIERTO?***

```
Haciendo click derecho con el ratón y copiando el enlace devuelve algo pa
recido a esto:  http://www.ejemplo.com/admin/ftp/objects/XXXXXXXXXXX.php
```

Así que tenemos un shell funcional y operativo pasandole el siguiente parámetro en la barra de direcciones del navegador web:

```
http :// www.ejemplo. com/ admin/ ftp/objects/XXXXXXXXXXX. php ? cmd = w
hoami
```

Notando que el servidor web corre guiones de **perl** (¿es en serio, **perl**?) pues tomamos un **shell** que reversa de nuestra lista favorita de trucos y ajustamos la dirección IP y puerto de doble vía con un **shell** de bajos privilegios (lo siento, no capturamos imágenes acerca del proceso)

Cinco minutos y ya con acceso limitado.

> ~Más o menos 5 minutos en el intento y ya tenemos un **shell** con privilegios bajos.

Para mi gran sorpresa, el servidor no estaba alojando únicamente un solo sitio web **sino 40 diferentes**. Tristemente no tengo tomas de pantalla de cada uno de los detalles pero la salida o lista de ficheros era algo como esto:

```
$ ls /var/www


access.log sitio1/ sitio2/ sitio3/ {... y la lista sigue y sigue}
```

Ahora entienden el punto. Sorprendentemente tengo derechos de lectura en **TODOS** los sitios web disponibles, lo cual significa que yo pudiera leer todo el código fuente, el que genera todo el código HTML. Me limito a mí mismo al código fuente del sitio de mi amigo, **ejemplo.como**

De manera notable, dentro del directorio cgi-admin/pages, todos los guiones en lenguaje **perl** se estaban conectando a la base de datos **MySQL** como usuario raíz "**root"**. Las credenciales para la base de datos estaban allí, en texto puro y "plano". Pongamos por caso que las credenciales eran **usuario_raiz** y **sometido42**

Ya confiado, el servidor estaba ejecutando MariaDB y tuve que solventar una falla antes de poder tener acceso a la base de datos. Luego ejecuté esto:

```
mysql -u usuario_raiz -p -h localhost base_datos_victima  Password: somet
ido42
```

Ya tenía acceso a la base de datos con derechos de "root", usuario raíz.

Siete minutos y 35 bases de datos.

> Después de 7 minutos, tenemos completo acceso de lectura y escritura al contenido de ¡35 bases de datos!

Aquí tengo la obligación moral de detenerme y revelar mis hallazgos hasta ahora. **El daño potencial es realmente grande.**

¿Qué pudiera hacer un atacante?

1. Vaciar el contenido de todas las bases de datos, tal como lo describen en este enlace, deviniendo en que la información de 35 empresas sea filtrado al dominio público.
2. Borrar el contenido de todas las bases de datos, eliminar por completo los datos de 35 compañías.

3. Dejar una "puerta trasera" para acceso fijo como "usuario apache" en una tarea programada, en caso de querer volver entrar al sitio.

Aquí debo hacer notar que el proceso MySQL estaba corriendo como usuario raíz así que imaginé que debería tratar con el comando **\! whoami** con la esperanza de obtener credenciales de usaurio raíz. Desafortunadamente aún estaba como usuario "apache"

Cronómetro detenido.

> Tiempo de tomar un descanso. Detener el cronómetro.

¿Qué más podría ir mal?

Después de revelar mis hallazgos, me concedo el permiso de indagar en profundidad.

Antes de buscar más vías para escalar mis privilegios a usuario raíz y tener así la capacidad de realizar un daño masivo, estuve buscando qué otros archivos interesantes había y que pudiera leer con mi usuario limitado.

De repente recordé acerca de los puertos **Samba** abiertos. Eso significa que debería haber algún directorio en algún lado que estaba siendo compartido en el sistema entre todos los usuarios. Después de un pequeño listado aparece el directorio **/home/samba/secured** (me disculpo por censurar masivamente el resultado).

Dentro de todos esos directorios hay archivos para cada usuario de la **compañía de alojamiento web**. Eso incluye todo tipo de datos sensibles, entre ellos:

- Archivos en formato **.psd** y **.ai** (los diseñadores gráficos saben de la importancia de mantener esto en privado, es su trabajo y sus técnicas, después de todo).
- Archivos de la aplicación **SQlite** y sus "galletitas".
- Facturas.
- Libros electrónicos *pirateados* (me rio entre dientes cuando veo esto).
- **Ceredenciales para sus propias redes inalámbricas de área local.**

¿Qué pudiera hacer un atacante?

1. Acampar fuera de las oficinas de la compañía, conectarse a su red de área local de manera inalámbrica y realizar todo tipo de ataques divertidos en ella.
2. Descargar toda la información sensible que mencionamos anteriormente y exponerla al dominio público.

> Me tomó cierto tiempo ir a través de todos las carpetas y darme cuenta de cuán serio es el asunto. Tomo un descanso más.

Se acaba el juego, el pitido final.

Después de estar conectado como *usuario apache* decido que es tiempo de ir por el premio gordo: obtener acceso como *usuario raíz.* Les refiero esta popular lista de trucos para escalar privilegios y comienzo por enumerar todo el sistema en búsqueda de archivos interesantes.

Debido a mi investigación hasta el momento, ya había pasado por la mayoría de estas técnicas y no parecía poder encontrar algo que aumentara mi posición.

Ahí es cuando esto me golpeó por sorpresa. En los desafíos de *captura de bandera* que estoy acostumbrado a jugar, el sistema operativo generalmente está "parchado" y es un servicio el que, intencionalmente mal configurado, finalmente otorga el privilegio de *usuario raíz* que tanto buscaba. En el mundo real sin embargo, las personas no "parchean".

¿Qué tipo de GNU/Linux está ejecutando el servidor?

```
$ cat /etc/issue  CentOS Linux release 7.2.1511 (Core)
```

¿Cuál versión de *kernel*?

```
sh-4.2$ uname -a  linux webserver 3.10.0-327.el7.x86_64 #1 SMP Thu Nov 19
 22:10:57 UTC 2015 x86_64 x86_64 x86_64 GNU/Linux
```

¡Eso parece una anticuada versión del *kernel!*

Encontré un weblog que me indica como probar si el *kernel* era vulnerable con el guion allí publicado y luego ejecuto lo siguiente:

```
sh-4.2$ ls  ls  cowroot  cowroot.c  rh-cve-2016-5195_1.sh  sh-4.2$ ./cowr
oot  ./cowroot  id  uid=0(root) gid=48(apache) groups=48(apache),5003(isp
apps),5004(ispconfig)  whoami  root
```

Juego finalizado.

De manera inmediata redacté un correo eletrónico develando los detalles y el potencial impacto de cada paso descrito arriba y que abarcó toda la noche. ¡Guao!

¿Qué pudiera hacer un atacante?

1. Leer o modificar TODOS los archivos en el servidor.
2. Dejar una puerta oculta fija y abierta (de hecho como *usuario apache*).
3. Instalar y potencialmente difundir software malicioso en la red de área local del servidor.
4. Instalar software de encriptado y secuestro de datos (tomando las bases de datos y todos los datos de la empresa de alojamiento web como "rehén" no es poca cosa).
5. Usar el servidor como minería de criptomonedas.
6. Usar el servidor como substituto de contenido *(proxy).*
7. Usar el servidor como "puente" de redes.
8. Usar el servidor como un soldado en un ejército de atacantes digitales.
9. ... (use su imaginación aquí).
10. Aplicar el comando **rm -rf/** (ni jugando, borraría absolutamente todo).

Al siguiente día fui contactado por mi amigo (quien a su vez estaba en contacto con la compañía que opera el servidor) y me informó que el error en el subidor de archivos estaba solucionado.

Conclusión.

En resumen encontramos:

1. Una aplicación con una vulnerabilidad de acceso irrestricto a subida de archivos que condujo a una interfaz de bajo privilegio.
2. Credenciales a una base de datos MySQL, lo que condujo al acceso de escritura y lectura de 35 bases de datos,
3. Cantidades de archivos con información crítica.
4. Finalmente un *kernel* no actualizado con una vulnerabilidad para acceder con derechos de *usuario raíz.*

Sugerencias para mitigar.

- Vamos a comenzar por lo que nos dio el pie inicial. Debido a que el sitio completo está escrito en lenguaje **perl** -y yo no manejo ese lenguaje de programación- pues de verdad que no puedo sugerir nada en ese aspecto.
- Una sugerencia pudiera ser no usar **perl** en 2017 pero eso sólo es mi opinión, siéntanse libres de probarme lo contrario.

- En cuanto se refiere al sistema de archivos, recomiendo gran cuidado en asignar los apropiados permisos de usuario de acuerdo al principio de mínimo privilegio. De esa manera aún si un usuario de bajos privilegio como *un usuario apache* obtuviera acceso, no sería capaz de leer ningún archivo sensible.
- Correr todos los sitios web en un un solo servidor es una mala idea, no estoy seguro si una solución como **Docker** pudiera resolver el problema.
- El tener la misma credencial para todas las bases de datos de seguro que tampoco es una buena idea.
- Finalmente **MANTENER TODO ACTUALIZADO**, en el caso de CentOS con **su -c 'yum update'**y en el caso de Debian y sus derivados **sudo apt-get upgrade**

Nuestra conversación por Twitter con el autor del artículo.

Este artículo va mucho más allá de la pintoresca historia, ya que nosotros desarrollamos aplicaciones, entonces le sugerimos una solución a la vulnerabilidad de subida irrestricta de archivos (aparte de difundir el artículo) y obtuvimos esta respuesta:

https://twitter.com/gakonst/status/933725615546564608

https://twitter.com/ks7000/status/934098443307552768

https://twitter.com/gakonst/status/934101705494540288

https://twitter.com/ks7000/status/934139879713370113

https://twitter.com/ks7000/status/934144411918888962

https://twitter.com/ks7000/status/934153201103921152

De especial significancia y de la cual aprendimos muchísimo, el siguiente enlace propuesto por el señor sr. Konstantopoulos.

Cuando difundimos el trabajo del señor Konstantopolous mencionamos en el mensaje al sr. Wardog, celebérrimo BOFH al cual seguimos y nos honra muchísimo nos halla hecho "retuit", todo un honor:

https://twitter.com/ks7000/status/934083447743827973

.