

VULNERABILIDAD DE FIJACIÓN DE SESIONES EN APLICACIONES WEB BY SH4V



INTRODUCCIÓN:

Bueno, el motivo que me ha llevado a escribir este white-paper, es que apenas hay información sobre vulnerabilidades en la fijación de sesiones (o session fixation). Leí sobre esto hace poco en un libro pero la información que venía era excasa así que me dispuse a recopilar información y elaborar este documento. Gran parte de lo aquí expuesto, está basado en el pdf publicado en www.acros.si/paper/session_fixation.pdf en el año 2002.

Muchas aplicaciones web utilizan sesiones para poder dar un servicio personalizado al usuario, restringir el acceso a zonas privadas, mostrar elementos html determinados, estilos por css personales... vamos, que las sesiones están presentes en el día a día de la World Wide Web. Por defecto, el protocolo HTTP es un protocolo sin estados y por lo tanto es incapaz de mantener estados en usuarios aunque se haya realizado previamente una solicitud hacia el mismo. Para solventar este problema, tanto los servidores, como las aplicaciones web, incorporan mecanismos de administración de sesiones. De esta forma, el servidor o la aplicación web, envía un Identificador de Sesión al cliente y el cliente envía dicho identificador para decirle al servidor que efectivamente, es él el que está visitando la página. El servidor lo que hace al comprobar que efectivamente existe ese usuario, es proporcionar al cliente aquellos privilegios que tiene en relación con el identificador.

Encontramos así tres métodos para mantener sesiones en entornos web:

- 🚩 ***Argumentos URL.***
- 🚩 ***Formularios ocultos.***
- 🚩 ***Cookies.***

De estos tres, el más conveniente tanto por seguridad como por flexibilidad es (por lo menos a mi modo de ver) el uso de cookies. A menudo, las sesiones con Identificador de Sesión, no son únicamente patrones de identificación pero sí autenticadores. Por ejemplo, un usuario puede loguearse con su login y password o por certificados digitales y a partir de ese momento, su cookie va a obtener una identificación que va a certificar que ése y no otro es el usuario. Si la aplicación no hace comprobaciones de IP, esto puede ser un blanco fácil para un atacante, que haciéndose con el ID de sesión, podrá acceder a la página desde la propia cuenta del usuario y por ende, con sus mismos privilegios. Esto puede llevarse a cabo de muchas maneras aunque la más conocida es por XSS y recibe el nombre de *Session Hijacking*. Llegados a este punto, nos encontramos con tres famosos tipos de ataque (expongo también su solución):

- 🚩 ***Intercepción de la sesión o del login/pass:*** Ya sea por XSS, sniffing... para evitarlo, nada como encriptar el tráfico. Si estás en una VPN o el mecanismo de autenticación se hace mediante HTTPS, puedes respirar un poco más tranquilo.

- 🚩 **Predicción de la sesión o del login/pass:** Cuando hablo de predicción del login/pass hablo de los típicos “root:root, admin:password, admin:1234, login:pass” o poner para el campo contraseña el mismo que para el login. Por ejemplo si sabemos que el administrador de la página web se llama Pablo, podemos probar a poner Pablo:Pablo o Pablo:Motos si el chico es asiduo al Hormiguero y tiene sentido del humor (para los que no sepan, Pablo Motos es el presentador de un programa de televisión emitido en España llamado el Hormiguero). ¡No sabéis la cantidad de páginas que me he encontrado con contraseñas tan fácilmente predecibles!

Cuando hablaba de predicción de la sesión, hablaba de una generación de ID débil. Esto suele pasar cuando el administrador del sistema quiere utilizar su método casero de generación de Identificadores de Sesión y lo hace de una forma tan chapucera como esta:

```
1435265724
1435265728
1435265732
1435265736
1435265740
1435265744
1435265748
1435265752
```

Una vez hemos sacado el patrón, si el servidor no se preocupa por las direcciones IP o no utiliza ningún mecanismo adicional de seguridad, podemos hacer un grinding de ID por fuerza bruta para hacernos con sesiones de otros usuarios. Los campos ID de una cookie, a menudo suelen contener varios apartados como por ejemplo “Ipservidor:login:tiempo” que son codificadas. Si el mecanismo de encriptación es hexadecimal o base64, nos será muy fácil sacar la password. Si el mecanismo de encriptación es MD5, SHA-1, SHA-2 o DES, entonces ya la cosa cambia. Si nos percatamos de que el ID de sesión no es ningún cifrado conocido, seguramente se trate de un método casero y si el administrador no ha puesto mucho interés el aspecto de la seguridad, nos será realmente fácil hacernos con el patrón de generación de ID's. Para solventar este problema utilizad una buena política de asignación de contraseñas.

- 🚩 **Ataque por fuerza bruta o diccionario:** Esto no tiene mucho que ver con sesiones pero quiero recalcarlo para concienciar al lector de que de nada sirve tener el sistema más seguro del mundo si la política de asignación de contraseñas es débil y da cabida a uno de estos ataques que revelarían al atacante las credenciales de los usuarios. Un ejemplo vendría a ser un ataque por fuerza bruta a un formulario web. Para solucionarlo, podríamos poner un límite de intentos en la aplicación o cualquier cosa que se nos ocurra (usad la

imaginación). Imaginemos la página:

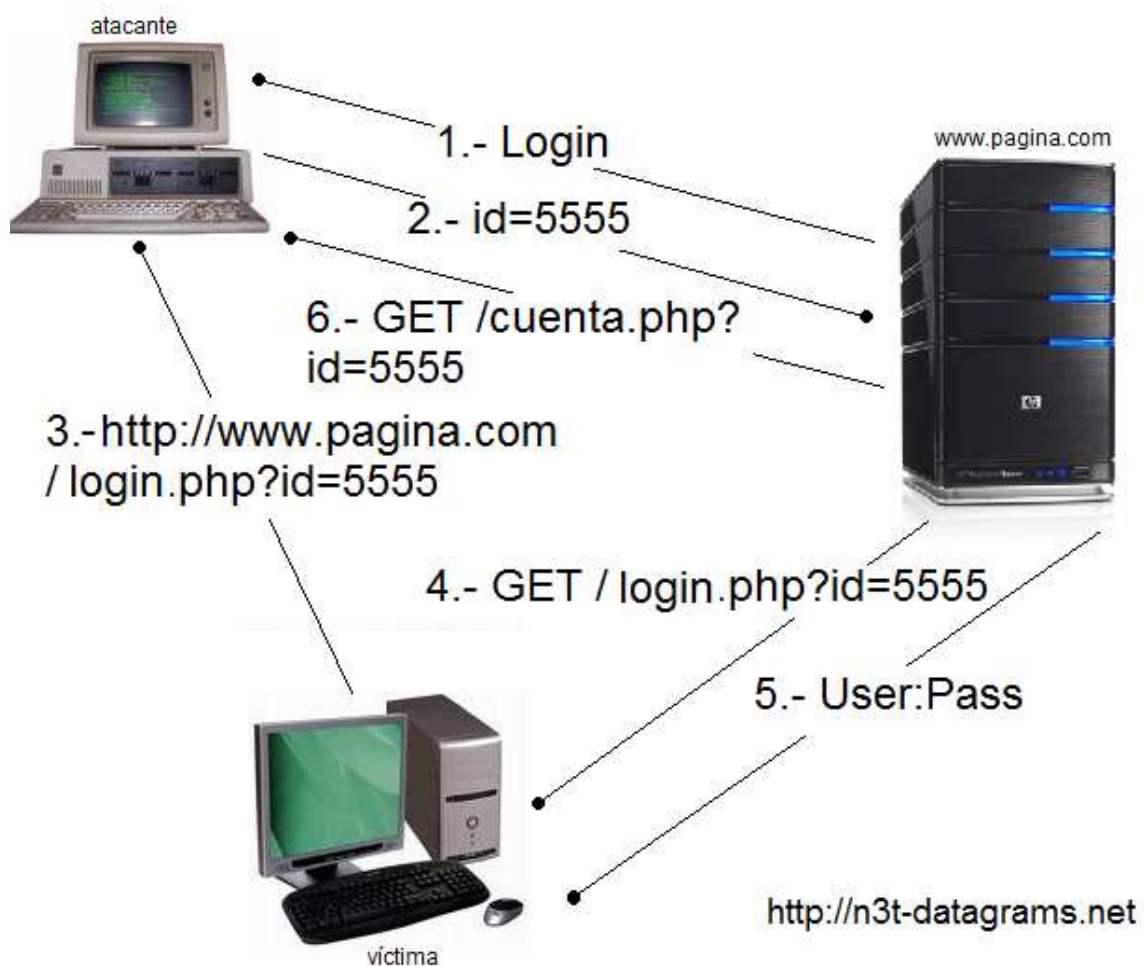
www.pagina.com/index.php?login&name=<user>&pass=<pass>

Si fallamos, nos mostrará una página distinta a si acertamos. Desde este punto de partida, podemos elaborarnos un script que ataque a los formularios o también podemos utilizar Hydra de THC:

```
$ hydra -L login.txt -P pass.txt www.pagina.com https-post-form "/index.php?login&name=^USER^&pass=^PASS^"
```

EL ATAQUE:

Para ver el ataque, lo mejor es utilizar un ejemplo:



Como podemos ver en la imagen:

- 1.- El atacante se loguea en la aplicación web.
- 2.- La aplicación web le asigna un ID de sesión que es el 5555.
- 3.- El atacante manda un enlace con el valor de ID asociado a la variable de tipo GET "login" a la víctima. Esto lo puede hacer mediante XSS multiplataforma, correo electrónico... (ing. social → Imaginación al poder!).
- 4.- La víctima accede a la aplicación web pero claro, debe loguearse.
- 5.- La víctima se loguea con su user y su password y esto queda automáticamente registrado en el servidor objetivo que inmediatamente va a asociar la ID que nos había dado a nosotros con la sesión de la víctima.
- 6.- El atacante vuelve a loguearse y ¡sorpresa! Puede acceder a contenidos privados del usuario víctima utilizando la misma ID de sesión.

Ahora vamos a verlo paso a paso. El ataque se divide en tres fases:

1.- Creación de sesión:

El atacante debe crear una sesión y obtener así el ID de sesión. Es posible que el servidor otorgue una caducidad a las sesiones o que estas mueran si no hay una nueva petición al servidor en un intervalo de tiempo determinado. En el primer caso, no podremos hacer mucho más que renovar la sesión cuando ésta caduque, en el segundo, deberemos mantener la sesión activa realizando peticiones de forma continuada al servidor para mantener la sesión viva. Hay que distinguir además dos tipos de servidores:

- 🚩 **Servidores permisivos:** Permiten ID's de sesión aleatorios y en caso de que no existan las sesiones, éstas se crean con los ID's propuestos. El atacante puede inventarse su propio ID siempre y cuando cumpla el formato y utilizarlo.
- 🚩 **Servidores estrictos:** No permiten ID's de sesión que no existan ya y que por tanto hayan sido generados dentro del propio servidor. El atacante debe iniciar sesión y utilizar un ID existente. Si recordáis, un poco más arriba explicaba un modo de obtener otros ID's si el mecanismo de generación era débil y carecía de una encriptación competente.

Como decía antes, el tiempo de la sesión puede expirar y para evitar esto, tendremos que mantenerla activa enviando peticiones al servidor. Los sistemas permisivos no requieren un mantenimiento de sesión.

2.- Fijación de sesión:

Llegamos a la segunda parte del ataque, en la que el atacante va a intentar que la víctima acceda a la url con el identificador de sesión enviado. Para ello, existen varios

métodos. Voy a ir exponiéndolos a continuación:

Por argumento en URL:

Consiste en enviar la url con el argumento oportuno como por ejemplo, la típica URL con la variable id=5555:

<http://www.pagina.com/login.php?id=5555>

El problema de esto, es que puede que la página que queramos atacar no utilice argumentos por URL para identificar sesiones y que lo haga por otros métodos, como por ejemplo, por cookies. Otro problema es que puede resultar bastante sospechoso para la víctima.

Por formularios ocultos:

Una forma de explotar esto sería utilizando un login panel falso para hacerse con las credenciales del usuario (lo que se conoce como Xploit) o aprovechar un XSS para hacerte con las cookies en el caso de que la página sea vulnerable. No olvidemos que el ataque XSS está entre los 10 primeros puestos de OWASP de vulnerabilidades web.

Por cookies:

Este es el método en el que nos centraremos más ya que los dos anteriores no tienen ningún misterio (a excepción del XSS que puede llegar a ser brutal si lo dominas). Las cookies son los principales transportadores de ID's de sesión y las encargadas del mantenimiento de sesión. Lo que debe de hacer el atacante, es introducir la cookie en el navegador de la víctima lo cual puede resultar imposible si tenemos en cuenta el RFC2965. Este RFC viene a decirnos que un navegador sólo podrá aceptar una cookie de un dominio o servidor determinado y por tanto la primera idea que se me ocurrió, quedaría inutilizada. Pensé en crear una aplicación web en mi propio servidor que asignase a la víctima (por supuesto, haciendo que ésta la visitase) una cookie spoofeada con el nombre del dominio del servidor víctima y no con el verdadero. Sin embargo, esto resulta *imposible*. Pongo en cursiva imposible, porque como habréis supuesto, existen métodos para poder hacerlo. Vamos a verlos:

1. **Asignación de cookie por ejecución de script en el lado del cliente:** Un claro ejemplo de explotación de este método es utilizando un XSS en la web víctima (en el supuesto caso de que la web sea vulnerable). Así podríamos asignar una cookie al navegador de la víctima con nuestro ID de sesión. Podemos utilizar

para ello Javascript, Vbscript, Flash o cualquier lenguaje de programación que se ejecute en el navegador del cliente.

```
document.cookie="id=5555";
```

De esta manera, en una página vulnerable como la siguiente: "<http://www.pagina.com/index.php?var=>", podríamos explotarlo de la siguiente manera:

```
http://www.pagina.com/index.php?var=<script>document.cookie="id=5555"</script>
```

En versiones antiguas de Microsoft Information Server, se puede engañar al servidor haciendo uso de los archivos IDC (Internet Database Connector) que sirven para integrar contenido de una base de datos en una página web de forma sencilla. Mediante un XSS, con código para crear una cookie y llamando a un archivo imaginario con extensión .idc, engañaríamos al servidor y crearíamos una cookie en el ordenador de la víctima con el ID de sesión previamente creado por nosotros:

```
http://www.pagina.com/<script>document.cookie="id=5555"</script>.idc
```

Como sabemos, el ser humano siempre quiere más. Así que, ya que podemos explotar la vulnerabilidad, cojamos el brazo entero a pesar de que nos hayan ofrecido la mano. Sabemos que los servidores y aplicaciones otorgan una caducidad a las sesiones, así que ¿por qué no darle un tiempo extra a nuestra sesión?

```
http://www.pagina.com/index.php?var=<script>document.cookie="id=5555; expires=Thu, 23 Aug 2009 00:00:00 UTC"</script>
```

Si además queremos que nuestro ataque no sólo afecte al dominio principal si no también a otros subdominios, añadiríamos:

```
http://www.pagina.com/index.php?var=<script>document.cookie="id=5555; expires=Thu, 23 Aug 2009 00:00:00 UTC; domain=.pagina.com"</script>
```

Esto podríamos utilizarlo si no encontramos una vulnerabilidad en el dominio principal pero sí en un subdominio o viceversa.

2. **Meta-tag injection:** Habrá navegadores que inhiban y por tanto no dejen ejecutar código script del lado del cliente. En este caso, el addon para firefox No-Script disponible en <http://noscript.net/getit> nos dificultaría mucho las cosas. Para estos casos podemos utilizar, aprovechando un XSS, la utilización de etiquetas

<meta> de html. Las etiquetas <meta> pueden tener varias utilidades, desde dejar datos meramente informativos sobre el sitio a indexar palabras claves para los robots de los buscadores. Si queréis más información sobre los meta-tags, os remito a la página del Guille:

http://www.elguille.info/HTMLscripts/HTML_meta.htm. En este caso lo utilizaremos para generar nuestra cookie:

```
http://www.pagina.com/index.php?var=<meta http-equiv= Set-Cookie  
content="id=5555; expires=Thu, 23 Aug 2009 00:00:00 UTC;  
domain=.pagina.com">
```

Como ya he dicho, la ventaja de este método con respecto al de la ejecución de un script en el browser del cliente, es que muchos navegadores buscan en el código html etiquetas <script> para bloquearlas pero no prestan atención a etiquetas <meta> porque en un principio no son dañinas.

No voy a explicar a fondo más métodos para crear la cookie en el navegador de la víctima aunque los hay (tan sólo usar la imaginación y tener conocimientos, claro). Una forma podría ser hackeando un servidor con un subdominio perteneciente al dominio pagina.com y crear una aplicación que genere una cookie en el navegador del cliente. Otra todavía más compleja, para la que sinceramente, o eres muy bueno o el administrador del DNS no ha puesto mucho cuidado en la seguridad, es hackear el servidor DNS que utiliza la víctima y asociar un servidor propio con un nombre de subdominio y realizar el mismo paso que acabo de explicar un poco más arriba (crear una aplicación para crear una cookie en el navegador de la víctima). Y otra interesante sería la de manipular la entrada y salida de datos del cliente mediante sniffing, pero para esto tenemos que estar en la red local y ésta no debe estar encriptada.

3.- Entrada:

Ahora sólo resta esperar a que el cliente se logúee para poder acceder a la página con todos los privilegios de la víctima. A diferencia del hijacking session, aquí no se está interceptando la ID de un usuario primero para después utilizarla, si no que lo que estamos haciendo es dar al usuario nuestra ID para que se logúee en el servidor y así éste asocie nuestra ID con la sesión de la víctima pudiendo tener acceso a todos los privilegios de dicha cuenta.

Greetz to Pr0x, Dynamique, N3t-Datagrams and Undersecurity (especialmente a OzX).