TRABAJO FINAL DE SWAP

INGENIERÍA INFORMÁTICA

Cookie Poisoning

Servidores Web de Altas Prestaciones
Horas Empleadas: 17 horas

Autores
Daniel Alconchel Vázquez
Mario Rodrígues López
Sergio Zapata de la Hoz

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, Junio de 2024

Índice general

1. Introducción

En un mundo donde la tecnología se ha entrelazado con casi todos los aspectos de nuestra vida diaria, la protección de datos y la privacidad se han convertido en preocupaciones primordiales. Desde nuestras interacciones en línea hasta nuestras transacciones financieras, cada acción deja un rastro digital que puede ser explotado si no se maneja adecuadamente.

Imagina por un momento que estás navegando en tu tienda en línea favorita y, sin darte cuenta, alguien está manipulando la información que el sitio web guarda sobre ti. Este proceso, conocido como **cookie poisoning** o **envenenamiento de cookies**, es una técnica sutil pero poderosa que los cibercriminales utilizan para robar datos, acceder a cuentas y causar estragos en el mundo digital.

El objetivo de este trabajo es desentrañar los secretos detrás del cookie poisoning. Exploraremos qué son las cookies, cómo pueden ser manipuladas para recopilar información personal sin nuestro conocimiento y consentimiento, y las implicaciones que esto tiene para nuestra privacidad en línea.

¿Te has preguntado alguna vez cómo un atacante puede acceder a tu cuenta sin conocer tu contraseña? ¿O cómo pueden robar tu información personal mientras navegas tranquilamente por internet? Pues sigue leyendo para descubrir cómo protegerte y comprender mejor una de las amenazas más insidiosas del mundo digital moderno.

2. Antecedentes

Este apartado revisará la literatura existente y presentará los conceptos fundamentales que sustentan el desarrollo de este trabajo, reforzando la necesidad de estudiar y comprender esta amenaza.

Historia y Evolución de las Cookies

Las cookies fueron introducidas por primera vez en 1994 por Lou Montulli, un programador de Netscape Communications, como una forma de gestionar la información de estado en el navegador web de los usuarios ([8]).

Originalmente diseñadas para mejorar la experiencia del usuario al recordar sus preferencias y facilitar el comercio electrónico, las cookies han evolucionado hasta convertirse en una herramienta crucial para la funcionalidad de la web moderna.

Conceptos Fundamentales

- **Cookies**: Pequeños archivos de texto que los sitios web almacenan en el navegador del usuario para recordar información sobre la sesión del usuario, preferencias, y otros datos útiles [11].
- **Sesiones Web**: Según el estudio de Wang y Goldberg (2006) ([15]), es un mecanismo utilizado por las aplicaciones web para mantener el estado de un usuario entre diferentes solicitudes a un servidor. Las cookies de sesión son fundamentales para este proceso.
- **Protocolo HTTP**: El protocolo HTTP/1.1 se define en el RFC 2616 (Fielding et al., 1999) ([7]). Se define como un protocolo de solicitud-respuesta utilizado en la comunicación entre el cliente y el servidor en la World Wide Web.

  Funciona como un protocolo sin estado, lo que significa que cada solicitud se procesa de forma independiente, sin conocimiento de solicitudes anteriores. Las cookies se utilizan para mantener el estado de la sesión del usuario entre solicitudes, permitiendo a las aplicaciones web realizar seguimiento del estado de la sesión, recordar preferencias y mantener la autenticación del usuario.

Uso de cookies actualmente

Actualmente los principales usos de las cookies son:

- **Gestión de Sesiones** : Mantienen a los usuarios autenticados mientras navegan por un sitio web, recordando su estado de inicio de sesión.

- **Personalización** : Almacenan preferencias del usuario, como el idioma y temas visuales, mejorando la experiencia de navegación.
- **Análisis y Seguimiento** : Recopilan datos sobre el comportamiento del usuario para análisis de tráfico web y marketing dirigido.
- **Carros de Compra** : Guardan los artículos añadidos en los carros de compras en sitios de comercio electrónico, incluso si el usuario navega fuera del sitio.

Puesto que se almacenan datos sensibles es necesario una buena protección de las mismas y es uno de los temas que discutiremos posteriormente en este trabajo.

3. Objetivos

A modo de resumen, este proyecto consiste en explorar y comprender el envenenamiento de cookies, así como identificar sus mecanismos, riesgos y medidas de protección, y realizar una prueba práctica para evaluar la efectividad de las estrategias de mitigación propuestas.

Desglose de Objetivos

1. **Definir y explicar el concepto de envenenamiento de cookies, incluyendo sus métodos y técnicas comunes.**
    - Describir las diferentes formas en que se puede realizar el envenenamiento de cookies.
    - Identificar las vulnerabilidades que hacen posible este tipo de ataque.
2. **Analizar casos reales y estudios de envenenamiento de cookies para ilustrar la gravedad y las consecuencias de estos ataques.**
    - Revisar incidentes documentados y reportes de seguridad.
    - Evaluar el impacto de estos ataques en diferentes sectores y aplicaciones web.
3. **Examinar y evaluar las técnicas y herramientas existentes para la prevención y detección del envenenamiento de cookies.**
    - Investigar las mejores prácticas de seguridad recomendadas para proteger las cookies.
    - Comparar diferentes enfoques y tecnologías utilizadas para mitigar estos riesgos.
4. **Realizar una prueba práctica que simule un ataque de envenenamiento de cookies y evaluar la eficacia de las medidas de protección implementadas.**
    - Diseñar y llevar a cabo un experimento controlado que reproduzca un escenario de envenenamiento de cookies.
    - Medir y analizar los resultados para determinar la efectividad de las estrategias de mitigación propuestas.

5. **Documentar y presentar los hallazgos del trabajo, proporcionando conclusiones y sugerencias para futuras investigaciones en el área de la ciberseguridad relacionada con las cookies.**
    - Resumir los resultados obtenidos y su relevancia para la comunidad de ciberseguridad
    - Ofrecer recomendaciones para estudios futuros y posibles mejoras en las técnicas de protección.

Estos objetivos están estructurados para proporcionar un análisis completo y detallado del envenenamiento de cookies, desde su comprensión teórica hasta la aplicación práctica de medidas de seguridad, culminando en una prueba práctica que validará la investigación realizada.

4. Desarrollo de los objetivos

4.1. ¿Qué es el envenenamiento de cookies? Conceptos y métodos utilizados

Como bien hemos hablado antes en los antecedentes, las cookies son datos específicos de un sitio web, que incluyen información de interés sobre los usuarios y que se almacenan en sus navegadores. Se utilizan principalmente para rastrear tendencias de uso, personalizar la experiencia de una web, etc...

Los atacantes pueden interceptar estas cookies antes de que vuelvan al servidor para extraer información de ellas o modificarlas. Las cookies también pueden ser falsificadas creando una desde cero como un medio de hacerse pasar por un usuario para acceder a datos adicionales de estos. Por tanto, el **cookie poissoning** es el nombre que recibe al conjunto de métodos utilizados para robar datos de las cookies de los usuarios u otros usos con intenciones maliciosas sobre las mismas.

Principalmente se usa esta técnica para la suplantación de identidad. Uno de los principales usos de las cookies es la autenticación, ya que contienen información que permite a los usuarios navegar sin estar autenticándose constantemente, por lo que la técnica de envenenamiento de cookies es un método recurrido por los atacantes para robar la identidad de los usuarios, con el objetivo de conseguir acceso no autorizado a un servidor web o para realizar fraude.

Algunos de los medios y técnicas que permiten a los atacantes realizar este tipo de ataques son los siguientes ([13], [1]):

- **Manipulación de cookies del lado del cliente:** Los atacantes pueden usar herramientas de desarrollo del navegador para modificar manualmente el contenido de las cookies o usar plugins que permitan la manipulación automática de las mismas.

- **Intercepción del tráfico:** Los atacantes pueden interceptar el tráfico de red, pudiendo capturar y modificar las cookies en tránsito, especialmente si no están cifradas (muy común cuando se usa el protocolo `HTTP` en lugar de `HTTPs`). A esta técnica de interceptar la comunicación entre un cliente y un servidor se le conoce como **Man in the Middle**.

- **Inyección de scripts (Cross-Site-Scripting - XSS):** Los atacantes pueden hacer uso de las vulnerabilidades que presentan las webs que estén ejecutando un software inseguro, mediante la inserción de scripts maliciosas que permitan obtener estos datos.

Dado estos distintos tipos de técnicas, podemos diferenciar entonces entre los siguientes tipos de cookie poissoning ([2], [12]):

- **Cookie Overwriting:** Es cuando el atacante fuerza al navegador del usuario a aceptar una cookie modificada, remplazando la original, normalmente mediante la inserción de scripts maliciosas en un sitio web.
- **Session Fixation:** El atacante establece una cookie de sesión específica en el navegador del usuario y luego, a través del uso de técnicas de ingeniería social o enlaces maliciosos, induce al usuario a autenticarse. De esta forma, el atacante recibe los datos de dicha autenticación, permitiéndole suplantar la identidad del usuario.
- **Cookie Hijacking:** El atacante roba una cookie válida de un usuario (por ejemplo, mediante XSS) y la usa para suplantar su identidad.
- **Cookie Replay:** El atacante reutiliza una cookie válida capturada previamente para repetir una acción autorizada. Esto es posible cuando los servidores no verifican adecuadamente la temporalidad y unicidad de las cookies.

Hemos visto los distintos tipos de cookie poissoning existentes y las técnicas usadas en las mismas, pero, ¿cómo es posible que ocurra este tipo de ataques? Algunas de las razones que permiten a los atacantes realizar este tipo de ataques son la **falta de cifrado (Uso de HTTP en lugar de HTTPs)**, que permite que las cookies sean mucho más fáciles de interceptar al no estar cifradas; el uso de **cookies no seguras**, que son aquellas que no usan los flags de `Secure`[1] y `HttpOnly`[2]; la **validación inadecuada del lado del servidor**, la reutilización de tokens de sesión, la falta de expiración adecuada de las cookies de sesión u otras vulnerabilidades de la aplicación web en cuestión.

4.2. Análisis de casos reales

Para comprender mejor la gravedad y las consecuencias del envenenamiento de cookies, es útil analizar casos reales. Algunos casos conocidos son los siguientes:

- **Ataque a Yahoo (2014-2016) [3]**: Entre 2014 y 2016, Yahoo fue víctima de un masivo ataque de envenenamiento de cookies que comprometió más de 3 mil millones de cuentas de usuarios. Los atacantes

[1] Asegura que las cookies solo se envíen mediante conexiones HTTPs
[2] Impide el acceso a la cookie mediante JavaScript, mitigando el riesgo XSS

utilizaron técnicas avanzadas para crear cookies falsificadas que permitieron el acceso a las cuentas de usuario sin necesidad de conocer las contraseñas. Este incidente afectó gravemente a la reputación de Yahoo, que ya estaba en negociaciones para ser adquirida por Verizon. La noticia del ataque redujo el valor de la adquisición en varios cientos de millones de dólares. Además, supuso un compromiso de información personal, como correos electrónicos, contactos, etc...

- **Exploit de Microsoft ASP.NET (2010)**: En 2010, se descubrió una vulnerabilidad en el framework ASP.NET de Microsoft que permitía a los atacantes realizar envenenamiento de cookies para obtener datos sensibles. El exploit, conocido como "Padding Oracle Attack," permitía a los atacantes descifrar datos encriptados en cookies y otros campos. Esta vulnerabilidad podía ser explotada para acceder a información cifrada, comprometiendo potencialmente millones de aplicaciones web basadas en ASP.NET.

- **Ataque de envenenamiento de cookies a LastPass (2022)** [14] : Un ataque de envenenamiento de cookies permitió a los ciberdelincuentes acceder a las cuentas de LastPass de los usuarios y robar sus contraseñas.

- **Vulnerabilidad en Shopify (2017)**: En 2017, se descubrió una vulnerabilidad en la plataforma de comercio electrónico Shopify que permitía el envenenamiento de cookies a través de un exploit de inyección de código. Este ataque podría haber permitido a los atacantes hacerse pasar por otros usuarios, accediendo a sus cuentas y datos personales. Aunque la vulnerabilidad fue rápidamente parcheada, el riesgo de comprometer datos financieros y personales de comerciantes y clientes de Shopify era significativo.

- **Ataque de envenenamiento de cookies a Barneys New York (2014)** : Un ataque de envenenamiento de cookies permitió a los ciberdelincuentes robar información personal de los clientes de Barneys New York, incluyendo nombres, direcciones, números de teléfono y direcciones de correo electrónico.

- **Estudio de la Universidad de California en Berkeley (2016)** : Un estudio de la Universidad de California en Berkeley encontró que el 70 % de los sitios web populares eran vulnerables a ataques de envenenamiento de cookies.

4.3. Detección y Prevención

Vamos ahora a estudiar algunas técnicas de prevención y detección del envenenamiento de cookies ([10], [6], [9]). Alguna de las medidas más utilizadas son las siguientes:

- **Uso de HTTPs** : Pese a que la configuración de https para los sitios web es una tarea más compleja que el simple uso de http, permite aumentar en gran medida la seguridad de la misma; en especial ayuda a prevenir la interceptación y manipulación de las cookies durante su transmisión.

- **Cookies Seguras (`Secure` y `HrrpOnly`)**: Ya hemos visto previamente estos conceptos. El uso de este tipo de cookies permite que solo se transmitan mediante https y mitigan los ataques de tipo XSS.

- **Implementar cookies en el mismo sitio** : Otra forma de evitar el robo de cookies es implementar cookies del mismo sitio, que restringen el alcance de las cookies al mismo dominio o subdominio que la aplicación web. Esto evita que los atacantes exploten las vulnerabilidades de falsificación de solicitudes entre sitios.

- **Supervisar y expirar sesiones** : Para detectar y responder al robo de cookies, también debe supervisar y caducar las sesiones de su aplicación web. Puede utilizar identificadores de sesión, marcas de tiempo, direcciones IP, agentes de usuario y otros atributos para realizar un seguimiento de la actividad y la validez de sus sesiones. También puede establecer un breve tiempo de expiración para sus sesiones y cookies, y solicitar a los usuarios que se vuelvan a autenticar periódicamente o después de ciertas acciones. De esta manera, puede reducir la ventana de oportunidad para que los atacantes utilicen cookies robadas e invalidarlas si detecta alguna anomalía o comportamiento sospechoso.

- **Políticas de Seguridad de Contenidos (CSP)** : Implementar políticas de seguridad para controlar cómo se carga y ejecuta el contenido web, ayudando de esta forma a prevenir ataques de inyección de código que podrían modificar las cookies.

- **Validación de cookies en el servidor** : Otra medida recomendada es implementar validaciones en el servidor que aseguren que las cookies no hayan sido alteradas.

- **Cookies firmadas y encriptadas** : Usar este tipo de cookies permite garantizar su integridad y confidencialidad, impidiendo su modificación y lectura por parte de los atacantes.

Además de estas medidas que son configurables e integrables por parte de los programadores, se pueden usar tras herramientas externas que ayuden a aumentar la seguridad de nuestra web, así como la detección de este tipo de ataques. Algunas de estas herramientas externas son los **Web Application Firewalls (WAFs)**, que permiten filtrar y monitorizar el tráfico HTTP hacia y desde una aplicación web para detectar y mitigar ataques, **herramientas de auditoría de seguridad**, como `Burp Suite` o `OWASP ZAP`, que permiten detectar vulnerabilidades en nuestra aplicación web, las cuales podrían ser utilizadas para el cookie poisoning o el uso de **Intrusion Detection Systems (IDS)**, como `Snort`, para monitorizar la red y los sistemas en busca de actividad sospechosa.

Incluso con todas estas herramientas, a día de hoy se siguen buscando nuevas formas de combatir este tipo de ataque. Hace pocos días `Google` anunció que está trabajando en nuevas medidas de seguridad. Está desarrollando una nueva función de seguridad para Chrome llamada **Credenciales de Sesión Vinculadas al Dispositivo (DBSC)** [[4], [5]] para combatir el robo de cookies de sesión y el acceso no autorizado a cuentas de usuario. Esta función vincula las sesiones de autenticación al dispositivo del usuario, haciendo que las cookies robadas sean inútiles para los atacantes, a menos que puedan operar localmente en el dispositivo comprometido. DBSC utiliza un par de claves pública/privada generadas localmente en el dispositivo, protegiendo la clave privada mediante TPMs y considerando soluciones aisladas por software. Cada sesión está asociada con una clave pública, y los servidores verifican la correspondencia de la clave privada para garantizar la continuidad del acceso. Además, DBSC mantiene las cookies de corta duración a través de un punto final definido por DBSC en el sitio web, reduciendo los cambios necesarios en sitios web y aplicaciones. Esta solución respeta la privacidad de los usuarios al no permitir la correlación de claves entre sesiones en el mismo dispositivo, y permite a los usuarios eliminar las claves cuando lo deseen. Google espera que DBSC se convierta en un estándar web abierto, colaborando con proveedores de servidores, IdPs y navegadores para presentar un estándar que preserve la privacidad y mejore la seguridad en internet.

Por último, también es aconsejable que se eduque a la población para que sea consciente de este tipo de problemas. Vivimos en una era digitalizada, donde todo el mundo usa, transmite y publica datos por la red; pero son pocas las personas que son conscientes de los peligros que existen en la misma y de las malas prácticas que deben evitar.

4.4. Prueba práctica

A continuación vamos a realizar una pequeña prueba práctica de interceptación de cookies. Todo se realizará en un entorno controlado y local. Además, iremos detallando paso a paso el experimento para que pueda ser replicado por cualquiera.

En nuestro caso estamos usando ***Ubuntu 22.0*** *para la prueba y el navegador* ***FireFox****. Usar otro sistema o navegador puede alterar ligeramente alguno de los pasos de configuración, pero el proceso sigue siendo el mismo salvo ajustes.*

Comenzamos creando una carpeta para el proyecto. Una vez creada y accedido a dentro del directorio, vamos a usar `Node.js` para crear un pequeño servidor web (*Si no tienes esta herramienta, puedes instalarla desde este enlace*).

Para crear el proyecto usamos los siguientes comandos:

```
# Inicializa un nuevo proyecto de Node.js
npm init -y

# Instala Express
npm install express cookie-parser
```

Lo siguiente es crear el servidor. Para ello, creamos el archivo `server.js` y añadimos el siguiente contenido:

```
const express = require('express');
const cookieParser = require('cookie-parser');
const app = express();

app.use(cookieParser());
app.use(express.json());

const PORT = 3000;

// Middleware para verificar autenticacion
function checkAuth(req, res, next) {
    const userCookie = req.cookies.user;
    if (userCookie) {
        const user = JSON.parse(userCookie);
        req.user = user;
        next();
    } else {
        res.status(401).send('No estas autenticado');
    }
}

// Ruta de login
app.post('/login', (req, res) => {
```

```
24          const { username, role } = req.body;
25          if (username && role) {
26              const user = { username, role };
27              res.cookie('user', JSON.stringify(user), { httpOnly
    : true });
28              res.send('Inicio de sesion exitoso');
29          } else {
30              res.status(400).send('Faltan parametros');
31          }
32      });
33
34      // Ruta protegida
35      app.get('/admin', checkAuth, (req, res) => {
36          if (req.user.role === 'admin') {
37              res.send(`Bienvenido, admin ${req.user.username}`);
38          } else {
39              res.status(403).send('Acceso denegado');
40          }
41      });
42
43      // Ruta publica
44      app.get('/', (req, res) => {
45          res.send('Pagina publica');
46      });
47
48      app.listen(PORT, () => {
49          console.log(`Servidor escuchando en http://localhost:${
    PORT}`);
50      });
```

Como podemos observar, es un código muy sencillo con 3 endpoints. El path `/` define la página principal y muestra el texto "Página pública". El path `/login` está configurado para recibir peticiones `POST` y permite iniciar sesión al usuario. Por último, el path `/admin` muestra la página sólo si somos administradores, en caso contrario nos devuelve un error `403`, indicando que no tenemos permisos.

Para levantar el servidor, simplemente ejecutamos el siguiente comando desde la terminal:

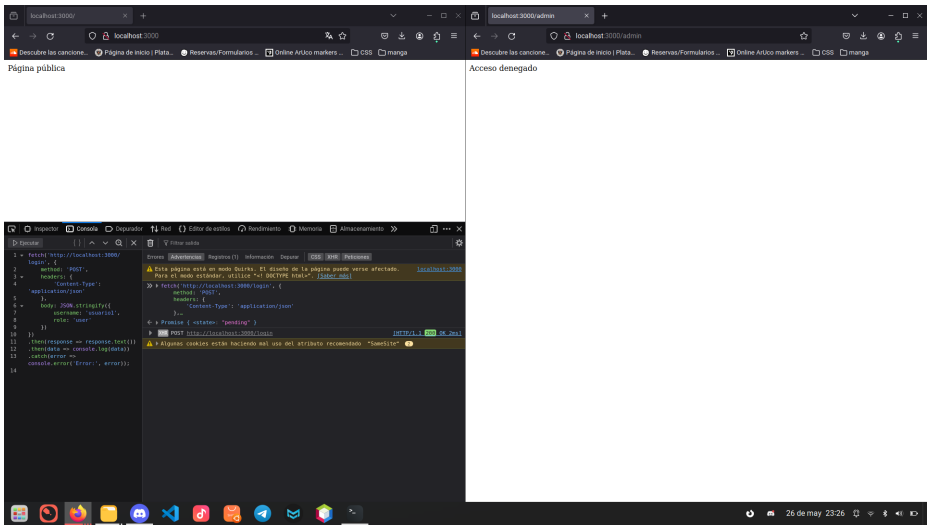
```
node server.js
```

Puedes acceder a la web desde el navegador usando la url `http://localhost:3000`.

A continuación, vamos a usar el navegador para mandar una solicitud `POST` a `http://localhost:3000/login` con el siguiente cuerpo `JSON`:

```
{
    "username": "usuario1",
    "role": "user"
}
```

Esto creará una cookie de usuario con el rol "user". Luego, intenta acceder a `http://localhost:3000/admin`. Deberías observar el texto 'Acceso Denegado'.

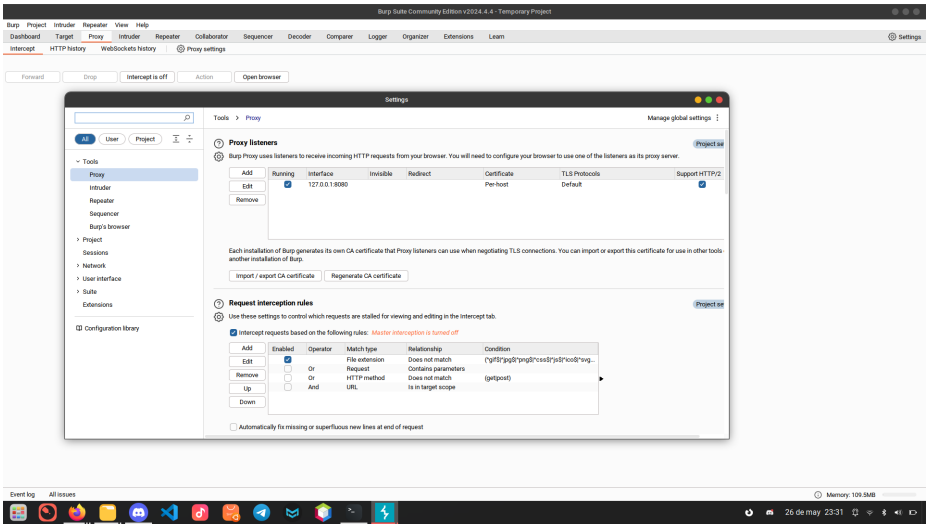


Para mandar la solicitud `POST` desde la consola del navegador puedes usar el siguiente código:

```
fetch('http://localhost:3000/login', {
    method: 'POST',
    headers: {
        'Content-Type': 'application/json'
    },
    body: JSON.stringify({
        username: 'usuario1',
        role: 'user'
    })
})
.then(response => response.text())
.then(data => console.log(data))
.catch(error => console.error('Error:', error));
```

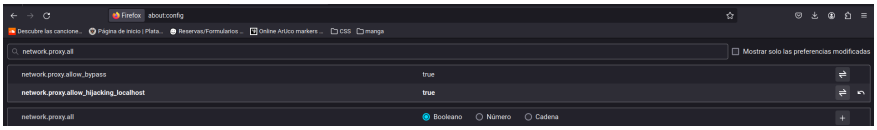
A continuación, vamos a instalar `Burp Suite Community Edition` para interceptar el tráfico y usarlo como proxy del navegador. Para instalarlo, sigue los pasos del link anterior.

Una vez instalados, podemos abrir el programa y creamos un proyecto temporal. Una vez dentro de la interfaz, nos vamos al apartado de `Proxy >Proxy Settings` y establecemos la siguiente configuración:

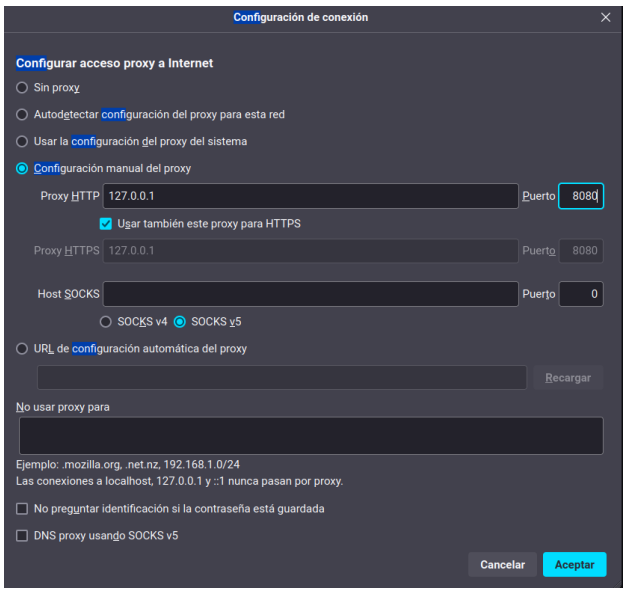


A continuación, vamos al navegador. En nuestro caso estamos utilizando `FireFox`, por lo que habrá que configurar unas cuantas cosas:

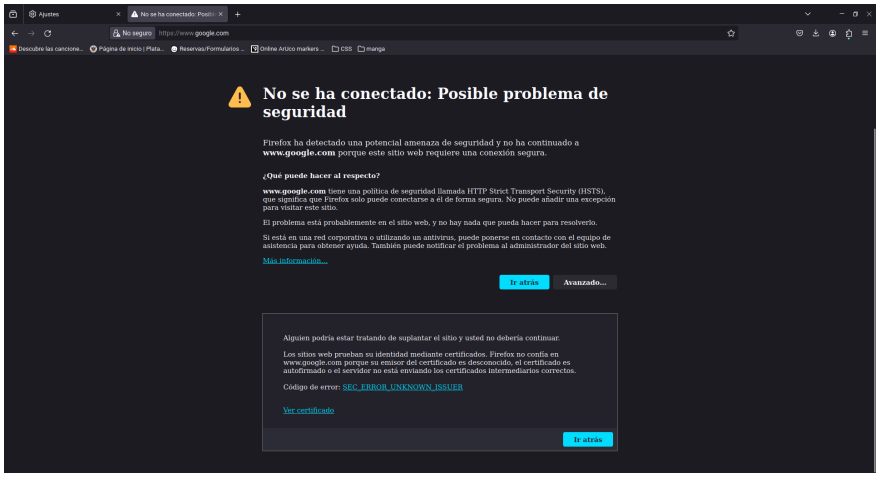
1. **Habilitar el proxy para localhost, el cual viene desactivado por defecto**: Para ello, simplemente escribimos `about:config` en la url y buscamos la siguiente configuración, la cual, debemos establecer a `true`:



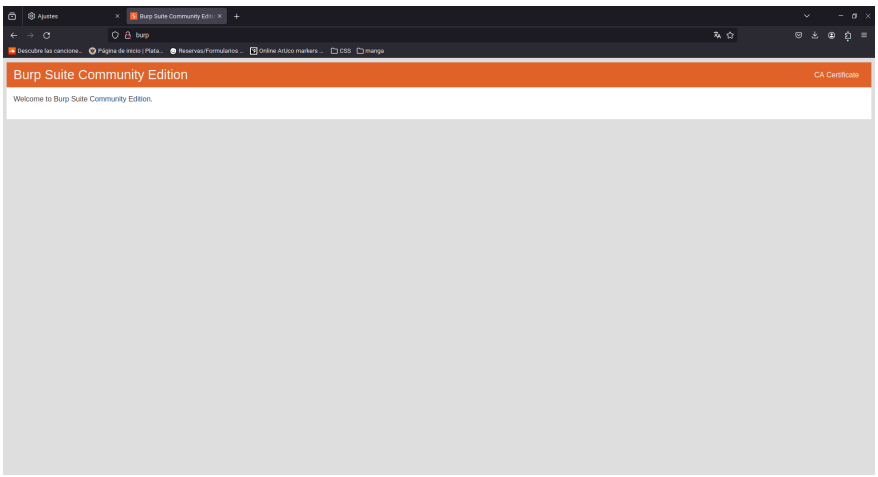
2. **Establecer Burp Suite como proxy** : Ve a la ventana de `Ajustes` de FireFox y busca `Configuración de red`. Allí, establece la siguiente configuración:



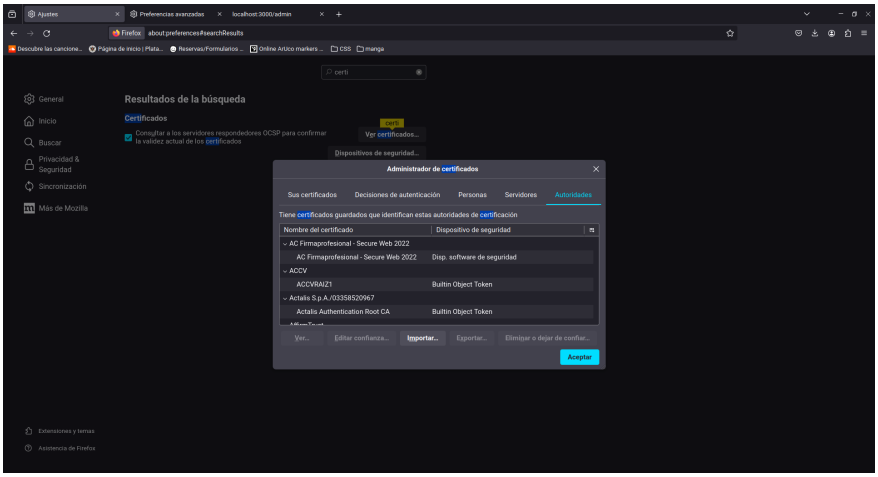
Acepta la configuración. Si intentas acceder a alguna web, como por ejemplo `google.com`, seguramente obtengas el siguiente error:



3. **Instalar certificado de `Burp Suite` en el navegador**: Para arreglar el error anterior, accede a `http://burp`. Deberías observar una web como la siguiente:

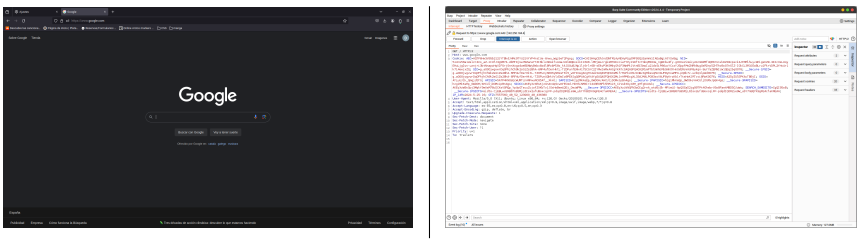


Una vez dentro, haz click en el botón para descargar el certificado y, a continuación, ve a `Ajustes >Certificados` para añadir el certificado al navegador:



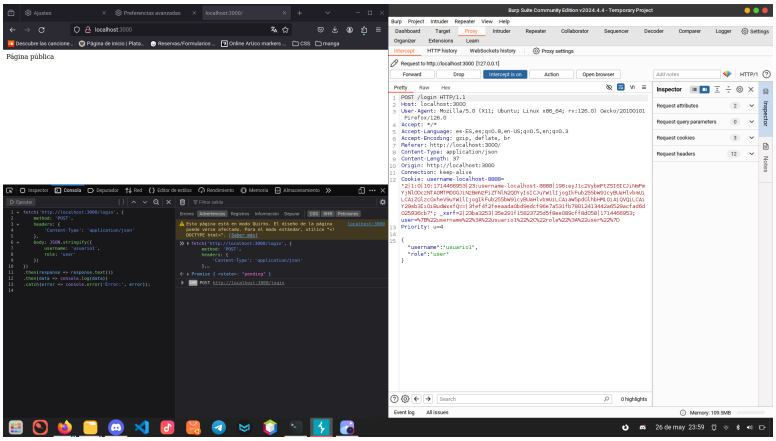
Dale a importar, marca todas las casillas y añade el certificado al navegador. Después de esto, podrás navegar de forma normal. Para interceptar las peticiones en `Burp Suite`, asegurate de que aparezca `Intercept is on` (*Si no aparece basta con hacer click en* ***`Intercept is off`***).

Comprueba que puedes usar el navegador correctamente y que llegan las peticiones a `Burp Suite`:

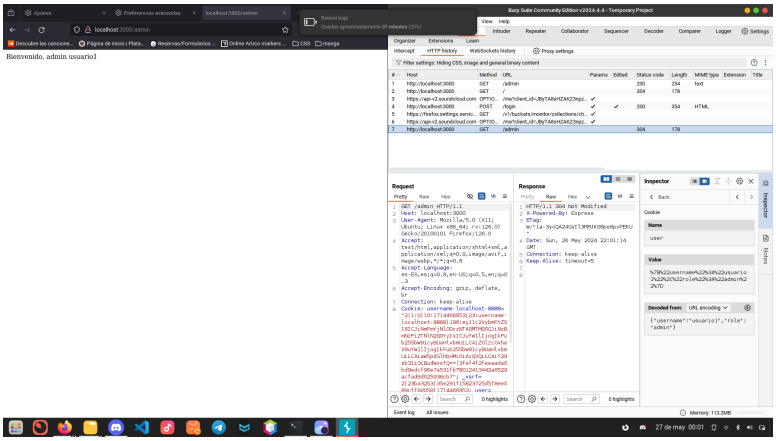


Es importante que recuerdes deshabilitar estas configuraciones cuando acabes el experimento.

Ahora vamos a volver a enviar la solicitud post 4.4. Esta vez al ejecutar el código, veremos la siguiente petición `HTTP` en `Burp Suite`:



Cambia el valor de "role" y establécelo a "admin". A continuación, haz click en `Forward`, lo cuál provocará que la petición se mande con la cookie modificada. Podemos acceder nuevamente a `http://localhost:3000/admin` para comprobar el resultado:



5. Conclusiones

En este trabajo, hemos investigado diversas formas de realizar ataques de **cookie poisoning**, una amenaza que puede parecer poco conocida a priori. Hemos identificado las vulnerabilidades más comunes y las técnicas más efectivas para prevenirlos y detectarlos. A continuación, vamos a presentar las conclusiones más claras que encontramos sobre este y algunas posibles mejoras en la protección frente a estos ataques.

Vulnerabilidades más comunes y prueba práctica expuesta

Como hemos visto anteriormente, las vulnerabilidades más comunes que permiten el *cookie poisoning* incluyen la falta de validación adecuada de las cookies y la insuficiente implementación de medidas de seguridad, como el uso de cookies HttpOnly y Secure o la falta de cifrado de estas.

La prueba práctica realizada con *Burp Suite* demuestra claramente cómo es posible modificar el valor de una cookie para obtener acceso a áreas restringidas de una aplicación web. Este pequeño experimento destaca la necesidad de una validación robusta y continua de la integridad de las cookies.

Impacto de estos ataques y recomendaciones finales

Los ataques expuestos a grandes empresas, como a Yahoo y LastPass, muestran la gravedad y el impacto de estos ataques y resaltan la necesidad de implementar medidas de seguridad para proteger la privacidad y la integridad de los datos de las aplicaciones.

Como recomendación para evitar ataques de **cookie poisoning**, toda aplicación debería implementar varias medidas de seguridad esenciales. Entre ellas, el cifrado adecuado de las cookies, el uso de la bandera *HttpOnly* y *Secure*, y la validación obligatoria de las cookies en el servidor.

Además, estar al tanto de los últimos desarrollos, como las **Device Bound Session Credentials (DBSC)** de *Google*, puede otorgar una capa adicional de protección contra el robo de cookies de sesión y el acceso no autorizado a cuentas de usuario.

En conclusión, este tipo de ataques es una amenaza importante para la seguridad de las aplicaciones web, pero con la implementación de las medidas adecuadas y la realización de investigaciones continuas, es posible mitigar estos riesgos y proteger de manera más segura la información de los usuarios.

Bibliografía

[1] AccessQ. Envenenamiento de cookies: Qué es y cómo prevenirlo, 2024. Accedido: 26 de mayo de 2024. URL: https://www.accessq.com.mx/envenenamiento-de-cookies/.

[2] AppSec Monkey. Session fixation, 2024. Accedido: 26 de mayo de 2024. URL: https://www.appsecmonkey.com/blog/session-fixation.

[3] BBC Mundo. El misterioso asesinato de una familia francesa que la policía aún no logra explicar, 2016. Accedido: 26 de mayo de 2024. URL: https://www.bbc.com/mundo/noticias-internacional-37444591.

[4] BitLife Media. Google desarrolla una nueva función de seguridad en chrome para evitar el robo de cookies de sesión, 2024. Accedido: 26 de mayo de 2024. URL: https://bitlifemedia.com/2024/04/google-desarrolla-una-nueva-funcion-de-seguridad-en-chrome-para-evitar-el-robo-de-c

[5] Escudo Digital. Google trabaja en una nueva función para evitar el robo de cookies, 2024. Accedido: 26 de mayo de 2024. URL: https://www.escudodigital.com/ciberseguridad/google-trabaja-en-nueva-funcion-evitar-robo-cookies_58578_102.html.

[6] F5 Networks. Cookie poisoning, 2024. Accedido: 26 de mayo de 2024. URL: https://www.f5.com/glossary/cookie-poisoning.

[7] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol – http/1.1. RFC 2616, 1999. URL: https://datatracker.ietf.org/doc/html/rfc2616.

[8] D. M. Kristol. Http cookies: Standards, privacy, and politics. *ACM Transactions on Internet Technology (TOIT)*, 1(2):151–198, 2001. Artículo que describe de forma completa que son las cookies, así cómo sus políticas de seguridad, privacidad y su integración con el protocolo HTTP.

[9] LinkedIn. ¿cómo detectar y responder al robo de cookies?, 2024. Accedido: 26 de mayo de 2024. URL: https://www.linkedin.com/advice/3/how-do-you-detect-respond-cookie-stealing?lang=es&originalSubdomain=es.

[10] LinkedIn. ¿qué es el envenenamiento de cookies http y cómo puedes prevenirlo?, 2024. Accedido: 26 de mayo de 2024. URL: https://www.linkedin.com/advice/1/what-http-cookie-poisoning-how-can-you-prevent-aaruc?lang=es&originalSubdomain=es.

[11] M. Rosenblum. *Web Security: A Step-by-Step Reference Guide*. Prentice Hall, 2001. Incluye una definición de cookie.

[12] IBM PTC Security. Cookie jar overflow attack, 2024. Accedido: 26 de mayo de 2024. URL: https://medium.com/@ibm_ptc_security/cookie-jar-overflow-attack-ae5135b6100.

[13] TechTarget. Cookie poisoning (envenenamiento de cookies), 2024. Accedido: 26 de mayo de 2024. URL: https://www.techtarget.com/searchsecurity/definition/cookie-poisoning.

[14] Una al Día - Hispasec. Criminales han tenido acceso a los sistemas de desarrollo de lastpass durante cuatro días, 2022. Accedido: 26 de mayo de 2024. URL: https://unaaldia.hispasec.com/2022/09/criminales-han-tenido-acceso-a-los-sistemas-de-desarrollo-de-lastpass-dura
html.

[15] X. Wang and I. Goldberg. Improved browsing for large web archives with in-page search. In *Proceedings of the 15th International Conference on World Wide Web*, 2006.