Euskal Herriko Unibertsitatea

Kudeaketaren eta Informazio Sistemen Informatikaren Ingeniaritzako Gradua

Informazio sistemen segurtasuna kuadeatzeko sistemak

# 2.lana

*Egileak:*
Iker Etxeberria
Imanol Ganzedo

*Gainbegiralea:*
Mikel Egana

2022ko Azaroaren 17a

# Gaien Aurkibidea

# 1 Sarrera

## 1.1 Sarrera

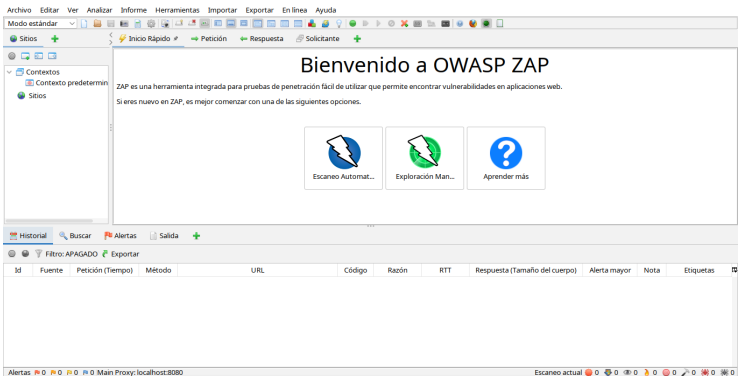
Lan honen helburaua, lehengo entregatik abiatuta, web-auditoria bat egitea da. Horretarako pentesting-tresna bat erabili dugu,ZAP kasu honetan. Tresna honek, webguneen ahuleziak bilatzeko erabiltzen da, webguneari eskaerak eginez, eta eskaera horiek aztertuz. Gure 1.entregan egindako lanaren ahuleziak bilatu eta gero, ahulezi horiek konpondu egin ditugu. Konpondu beharreko ahulezi batzuk hauek dira:

- Sarbide-kontrola haustea
- Akats kriptografikoak
- Injekzioa
- Diseinu ez-segurua
- Akatsak segurtasunaren monitorizazioan
- Osagai kalteberak eta zaharkituak
- Identifikazio- eta autentifikazio-akatsak
- Datuen eta softwarearen osotasunaren akatsak

# 2 ZAP-en funtzionamendua

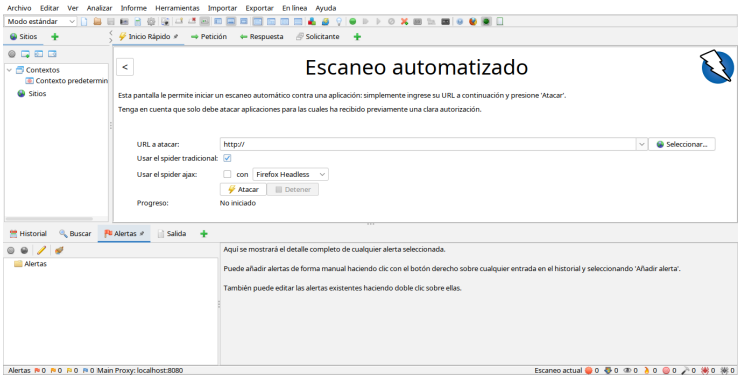
## 2.1 ZAP-en funtzionamendua

Lehenik eta behin ZAP aplikazioa deskargatu behar dugu gure sistema eragilerako. Gure kasuan, linux eragilea erabiltzen ari garenez, linux-en nola instalatu azalduko dugu. Lehenik eta behin goiko link-etik linux sistema eragilerako ZAP.sh fitxategia deskargatuko dugu; behin fitxategia deskargatuta, exekutatzeko baimena emango diogu eta terminalean "Descargas"karpetan sartuko gara eta .sh fitxategia exekutatuko dugu hurrengo komandoarekin: ./fitxategiaIzena.sh . Behin hori eginda, aplikazioa instalatuko dugu, eta instalatuta dagoenean, aplikazioa irekiko dugu eta hurrengo pauzuak jarraituko ditugu:




Irudia 1: ZAP-en pantaila printzipala

1.irudian ikus dezakegu, ZAP aplikazioaren orrialde printzipala. Gure web orrialdaren ahuleziak bilatzeko, "escaneo automatizado"botoian eman behar diogu.

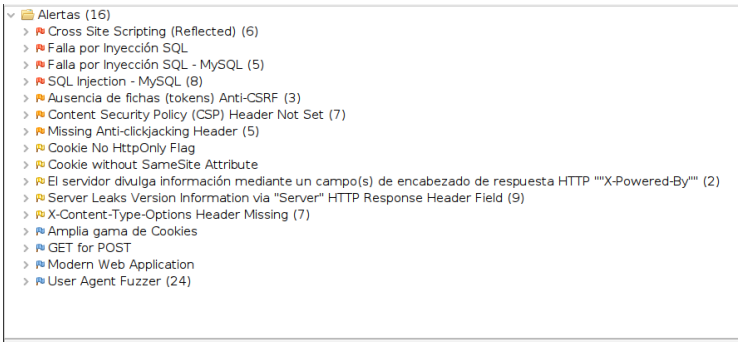



Irudia 2: Web orrialdetako ahuleziak bilatzeko

Behin botoi horri emanda, 4.irudiko pantaila agertuko zaigu. Bertan "URL a atacar"atalean, gure webgunearen URL-a sartu beharko dugu(gure kasuan: "http://localhost:81/"), eta "atacar"botoiari emanda, gure webgunea erasotuko du. Behin prozesua amaituta, "alertas"atalean, gure webgunearen ahuleziak ikusi ahalko ditugu.

# 3 Lehengo auditoria

- **Lehen ZAP emaitza:** Segurtasun eta ahaulezi kode aldaketak egin baino lehen, hurrengo argazkian ikusten ditugun ahulezi guztiak agertzen zizkigun.




Irudia 3: ZAP: hasierako emaitza

# 4 Webgunearen ahuleziak eta konponbideak

Lehen esan bezala, gure web orriak hainbat ahulezi desberdin zituen; hona hemen guk aurkitutako batzuk, eta horiek konpontzeko egindako kode aldaketak:

- **Injekzioa:** SQL kontsultetan, interpreteak zuzenean kontsulta dinamikoak edo parametrizatu gabeko deiak erabiltzen genituen, hau da, "escaping-ik gabe. Hori konpontzeko SQL kontsultak parametrizatu egin genituen. Adibidez, aldagaiak zutabe-izenak aipatzen baditu, interpreteak ziurtatzen du parametroak zutabe-izenak direla kontsultara gehitu eta exekutatu aurretik.

```
23      $_SESSION['saioa'] = 1;
24      $stmt = $conn->prepare("SELECT * FROM erabiltzaileak WHERE
        ↪  izen_abizenak=? AND pasahitza=?");
25      $stmt -> bind_param("ss",$izen_abizenak,$pasahitzaZifratuta
);
26
27      //execute
```

- **Identifikazio- eta autentifikazio-akatsak:** Identifikazio eta erregistroan hainbat ahulezi genituen, hemen daukagu izandako arazo guztiak eta konponbideak:

  - **Hash-ak eta testu ahula:** Gainera, pasahitzak datu basean testu soilean gordetzen genituen, hash-eatu gabe eta gatzarik gabe. Hori konpontzeko erabiltzaileak erregistratzean sartutako pasahitzari gatz bat gehitu genion, eta gero pasahitza guztia hash-eatu ondoren, datu basean gorde genuen.

```
16    $pasahitza = $_POST['pasahitza'];
17    $gatza = md5("4hJUd5sPP97hT");
18    $pasahitzaZifratuta = md5($pasahitza.$gatza);
```

  - **URL-an saio identifikatzaileak :** Gure kasuan hasieratik ahulezi hau saihestu genuen, eta web nabigatzaileko bilatzailean ez ziren agertzen erabiltzaileen saio identifikatzaileak; beraz, ez dugu aldaketarik egin atal honetan.

– **Pasahitza ahulak saihestu:** Hasteko, hasieran pasahitz ahulak jartzeko aukera uzten zuen, hori konpontzeko erregistratzeko orduan, gakoak hainbat berezitasun izan behar zituen. Adibidez, gakoak zortziko luzera edo handiagoa izatea, gutxienez letra larri bat izatea, gutxienez zenbaki bat izatea eta gutxienez karaktere berezi bat izatea.

```
65        function pasahitzaEgiaztatu(){
66          let pasahitza =
    ↪  document.formularioregistro.pasahitza.value;
67          if(pasahitza.length >= 8){
68
69
70            if(pasahitza.match(/([a-zA-Z])/) &&
    ↪  pasahitza.match(/([a-z].*[A-Z])|([A-Z].*[a-z])/) &&
    ↪  pasahitza.match(/([0-9])/) &&
    ↪  pasahitza.match(/([!,%,&,@,#,$,^,*,?,_,~])/)){
71              return true;
72            }
73            else{
74              window.alert("8 karaktere edo gehiago izan
    ↪  behar ditu, letra larriak, zenbakiak eta karaktere
    ↪  espezialak erabili behar dira");
75              return false;
76            }
77          }
78          else{
79            window.alert("Pasahitza hau oso motza da (8
    ↪  karaktere edo gehiago izan behar ditu, gainera letra
    ↪  larriak , zenbakiak eta karaktere espezialak erabili
    ↪  behar dira)");
80            return false;
81          }
```

– **Logout eta aktibitate eza:** Gure kasuan hasieratik logout egiteko botoi bat jarri genuen gure webgunean; horrela, erabiltzaileak web orrialdetik alde egin baino lehen sesio ixteko aukera zuen. Hala ere, aktibitate faltagatik sesioa ixteko kodea falta zitzaigun, eta bigarren entrega honetan, arazo hori konpondu egin dugu, hurrengo argakian ikusi dezakezuen kodea inplementatuz.

```php
$inactivo = 60;

if(isset($_SESSION['denbora']) ) {
$vida_session = time() - $_SESSION['tiempo'];
    if($vida_session > $inactivo)
    {
      session_destroy();

      echo'
        <script>
          window.alert("Saioa berrabiarazi behar duzu")
          window.location= "index.php"
        </script>
      ';


    }
}

$_SESSION['denbora'] = time();
```

- **Sarbide-kontrola:**
  - **Sarbide-mekanismo bakarra :** Kontrolerako sarbide-mekanismoak behin bakarrik ezartzea eta aplikazio osora zabaltzea nahi genuen. Hori lortzeko, behin saioa hasita, erabiltzaile datuak sesio aldagai desberdinetan gordeko dira, horrela kontsultak egiterakoan sesio aldagaien bidez erabiltzaileak kontsultak egin ahalko ditugu, berriro logeatu behar gabe.

```php
    $telefonoa = $row["telefonoa"];
    $jaiotzeData = $row["jaiotze-data"];
    $timestamp = strtotime($jaiotzeData);
    $newDate = date("d/m/Y", $timestamp );
    $email = $row["email"];
    $pasahitza = $row["pasahitza"];
    $balorazioa = $row["Balorazioa"];

```

– **Sartzeko ahalegin guztiak logeatzea :** Nahiz eta atal hau ez implementatu, gure idea logeatzeko ahalegin guztiak gure datu basean gordetzea da. Eta erabiltzaile batek, logeatzeko ahalegin gehigi eginez gero (5 aldiz pasahitza txarto sartuz), erabiltzailearen gmailera bere erabiltzailearen pasahitza aldatzeko mezu bat bidaltzea zen helburua. (hau egiteko, kode erraldoi bat deskargatu eta inportatu behar da. Beraz, zurekin komentatu genuenez, ez dugu inportatu eta funtzionalitate hori ez du funtzionatzen. Hala ere, ulertzen dugu nola funtzionatzen duen eta hemen behean uzten dizugu hau lortzeko erabili behar den kodea.)

```php
<?php session_start();

$mail = $_SESSION['email'];

//Create a new PHPMailer instance
$mail = new PHPMailer();
$mail->IsSMTP();

//Configuracion servidor mail
$mail->From = "segurikerimanol@gmail.com"; //remitente
$mail->SMTPAuth = true;
$mail->SMTPSecure = 'tls'; //seguridad
$mail->Host = "smtp.gmail.com"; // servidor smtp
$mail->Port = 587; //puerto
$mail->Username ='segurikerimanol@gmail.com'; //nombre
↪  usuario
$mail->Password = 'Seguridad2022'; //contrasena

//Agregar destinatario
$mail->AddAddress(".$mail.");
$mail->Subject = "sarbide kontrola";
$mail->Body = "Norbait zure erabiltzaileraekin sartzen
↪  saiatu da";

//Avisar si fue enviado o no y dirigir al index
if ($mail->Send()) {
    echo'<script type="text/javascript">
            alert("Enviado Correctamente");
            window.location = "../index.php";
        </script>';
}
```

– **Eskaeren tasa mugatzea :** Aurreko atalarekin lotuta, norbaitek logeatzeko ahalegin gehiegi eginez gero(5 saiakera ), erabiltzaile horrek betirako blokeatuta geratu egongo da; hala ere, administratzaile desblokeatu ahal izango du nahi izanez gero.

```
29      $txarto = $_SESSION['pasahitzaTxarto'];
30      if($txarto >= 5){
31        echo'
32        <script>
33            window.alert("Erabiltzaile bloqueatuta")
34            window.location = "../index.php";
35        </script>
36    ';
```

- **Osagai kalteberak eta zaharkituak:**
  - **Erabiltzen ez dena ezabatu :** Hasierako entregan bidalitako fitxategian erabiltzen ez ziren dependentziak, funtzionalitateak, osagaiak, aldagaiak... genituen, eta kode osoa birpasatu ondoren, erabiltzen ez ziren atal guztiak ezabatu genituen.
- **Akats kriptografikoak :**
  - **Algoritmo eguneratuak (MD5, SHA1...) gako indartsuak:** Akats kriptografikoak saihesteko, pasahitza enkriptatu egin dugu. Lehen enkriptatatu barik gordetzen genuen datu basean. Beraz, segurtasun akats baten ondorioz, datu basea engaiatua geratzen bada, erabiltzaileen pasahitzak lortzea posiblea izango litzateke.
- **Content Security Policy (CSP) Header Not Set:**
  - **Content Security Policy:** Eraso mota batzuk prebenitzen eta arintzen laguntzen duen segurtasun-geruza gehigarria da, besteak beste, Cross Site Scripting ( XSS (en-US) ) eta datuen injekzio erasoak. Hau konpontzeko, gure HTML kodean hurrengo esaldi hau jarri behar izan dugu:

```
6     <meta http-equil="Content-Security-Policy" content="
↪   'self'  'unsafe-inline'">
```

- **Anti-CSRF Tokens :** Eraso honek biktimaren web-arakatzailea, zerbitzu batzuetan baliozkotua (adibidez, posta edo etxeko bankuan) eskaera bat bidaltzera behartzen du web-aplikazio zaurgarri batera. Arazo hori konpontzeko, formularioan hurrengo lerroak gehituko ditugu; token-ak sortuz eta formulario era ezkutuan sartuz.

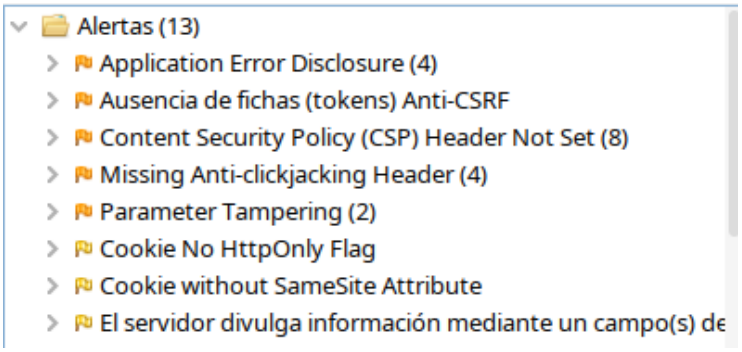
```
18      $_SESSION['token'] = bin2hex(random_bytes(24));
19
20      echo '<div class="laukia">
21          <form action="php/index.php" method="post"
    ↪  style="text-align:center" name="formulariologin">
22              <div class="login"><h1>LOGIN</h1></div>
23                  <input type="hidden" name="token"
    ↪  value="'.$_SESSION['token'].'" />
24                  <input class="bete" type="text"
    ↪  placeholder="izena" name="izen_abizenak">
25                  <input class="bete" type="password"
    ↪  placeholder="Pasahitza" name="pasahitza">
26                  <input class="botoia" type="button"
    ↪  value="Sartu" onclick="datuakKonprobatu()">
27                  <p><a  href="registro.php">Oraindik ez zaude
    ↪  erregistratua?</a></p>
28              </div>
29          </form>
30        </div> ';
```

Eta gero, logina egin baino lehen, sesioko token-a eta formularioaren token-a berdinak diren konprobatuko dugu.

```
21  if (!hash_equals($_SESSION['token'], $_POST['token'])) {
```

# 5 Azken auditoria

- **Azken ZAP emaitza:** Aldaketa hauek guztiak egin eta gero, hau da ZAP erasoa eginez lortzen dugun emaitza finala, ikuz dezakegu akats asko desagertu direla. Haien artean ikurrina gorria zutenen erasoak, larrienak zirenak.




Irudia 4: ZAP: amaierako emaitza

# 6 Bibliografia


- 'Prevención de ataques de falsificación de solicitudes entre sitios (CSRF)'

- 'Pruebas de Penetración con Zed Attack Proxy'
- 'Egelako apunteak'
- 'Cómo crear claves secretas en PHP'
- 'Cierre de sesión por inactividad'
- 'Como usar phpMailer'