Informazio Sistemen Segurtasuna Kudeatzeko Sistemak

Lana 3 – Pentesting (14/11/2025)

Uxue Aurtenetxe
Aimar Basterretxea
Ane Moreno
Yoel Justel
Xinyan Wang

Sarrera

Hurrengo dokumentuan, gure web orriaren proiektuan egindako pentesting probak aurkeztuko ditugu. Garrantzitsua da aipatzea aurreko entregan hainbat ahultasun detektatu genituela ZAP-ekin egindako analisiari esker, eta entrega berri honetan horiek zuzentzeko beharrezko konponbideak aplikatu ditugu. Gure helburua, webgunea ZAP erabiliz berriro atakatzea eta erasoa egitean alertarik ez agertzea, webgunearen osotasuna eta segurtasuna bermatzeko.

Irakaslearen argibideei jarraituz, GitHuben Entrega_3 izeneko adar gehigarri bat sortu dugu. Adar horretan honakoa sartu dugu:

- Web Sistemaren bertsio eguneratua, 2. Entregan hautemandako kalteberatasunak zuzentzeko beharrezko aldaketa guztiak biltzen dituena.
- Aurkitutako kalteberatasunak nola konpondu diren dokumentatzen duen PDF txostena, urratsez urrats zien alerta identifikatu ziren eta nola konpondu diren azalduz.

Eskatu zitzaigun bezala, berriro ZAP exekutatu dugu, baieztatzeko jada ez daudela kalteberatasunik egiteke. Gainera, txostenean horietako bakoitza nola konpondu den zehazten dugu. Era berean, erabilitako baliabideak, hau da, kanpo dokumentazioa kontsultatu badugu, txostenean behar bezala aipatu dugu.

Berrikusi eta konpondu ditugun nahitaezko alertak (edo ZAP-en alerta gisa agertzen ez zirenean justifikatuta) honako hauek dira:

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

Azkenik, irakasleak azaldutakoan oinarrituz, ZAP-ek ez ditu soilik goian aipatutako kalteberatasunak detektatzen. Horregatik, dokumentatu eta zuzendu ditugun kalteberatasun batzuk ez dira agertzen aipatutako zerrendan. Era berean, aipatu dugun bezala, irakaslearen gidan jasotako ahultasun batzuk ez zaizkigu alerta gisa agertu, eta behar bezala justifikatu dugu txosten honetan.

Aurreko Entregarekiko Aldaketak

Irakaslearen feedbacka jaso ondoren, lehenengo entregatik oraingora aldaketa batzuk zuzendu ditugu proiektuan. Ez dira oso nabariak baina funtzionalitatean edo informazioa bistaratzeko orduan nabariak dira.

Lehenengoa filmak bistaratzerakoan informazio gehiago pantailaratzea da. Pelikularen izena soilik agertu beharrean, horren argitalpen urtea eta zuzendaria ere agertzen dira oraingoan.

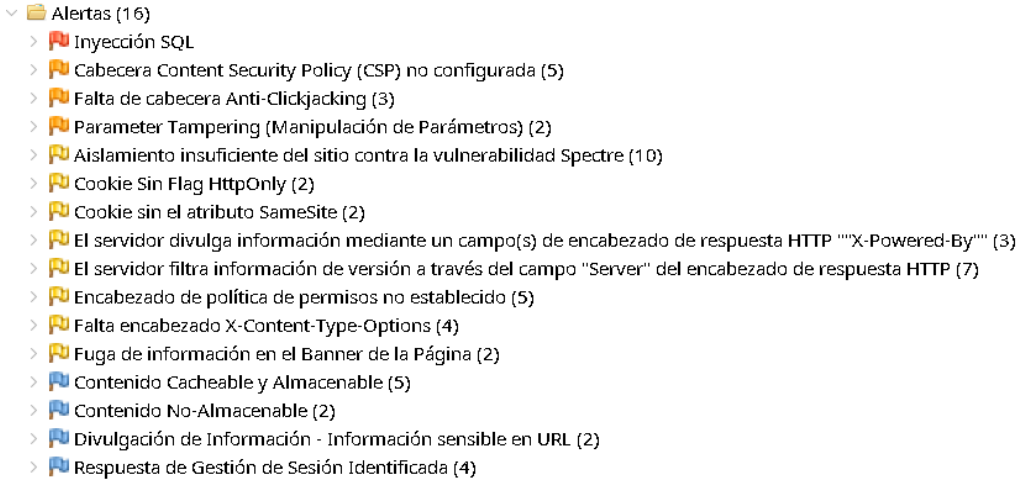
Beste alde batetik erabiltzaileak filmak editatzeko zuen aukeretan parametroen balioztatze prozesua aldatu dugu. Hau da, pelikularen izena, generoa eta bestelako informazioa aldatzerako orduan, “-” sinboloak sartzeko aukera genuen, egungo bertsioan soilik zenbakiak eta hizkiak onartuko dira.

Jasotako Emaitzak

- Egindako segurtasun-azterketa honetan hurrengoak dira aurkitu diren akatsak larrienetik informatibo hutsera ordenatuta. Dena den, badaude akats batzuk behin baino gehiagotan errepikatzen direnak; taulan kontuan hartu arren, ondoren emango diren azalpenak behin bakarrik azalduko dira.

Taula 1: Lehenengo proba lortutako emaitzak

		Konfidentzialtasuna			
		Altua	Ertaina	Baxua	Guztira
Arriskua	Altua	0	1	0	1
	Ertaina	1	1	1	3
	Txikia	2	6	0	8
	Jakinarazpenak	1	3	0	4
	Guztira	4	11	1	16



Irudia 1: Jasotako alertak

1. SQL injekzioa

Aplikazioak erabiltzaileak sartutako datuak behar bezala balioztatzen edo iragazten ez dituenean gertatzen da errore hau. Datu horiek zuzenean SQL kontsulta baten barruan txertatzen direnean hain zuzen ere.

Gure webgunearen kasuan, horrelaxe gertatzen da, erabiltzaileak saioa hasterako orduan edo pelikula berria datu basean sartzerako orduan eskuragarri dituen kutxatiletan edozer idatz dezake. Informazio hori zuzenean datu basean gordetzen da kodetik pasatu gabe.

Irudia 2: SQL injekzio alerta

Hau konpontzeko aurreko entregan azaldutako pausuak jarraitu ditugu. Garrantzi handiena duen soluzioa parametrizatzea izan da. Honek eta kontsultak prestatzeak SQL aginduak eta erabiltzailearen informazioa isolatzen ditu. Hona hemen adibidea index.php fitxategian:

```
36  $query = mysqli_query(mysql: $conn, query: “SELECT * FROM usuarios…
```

Bigarrenik, erabiltzaileak sartu beharreko eskariak prestatzea zen. Horretarako *prepared statement* erabili dugu. Adibidez add_item.php fitxategian:

```
90  $stmt = $conn->prepare(query:”INS…”);
```

Eta azkenik, erabiltzaileak sartutako edozein informazio guretzat egokia den ziurtatu, karaktere berezirik duen, luzera egokia duen…

```
74  If(empty($izena)) || strlen(string:$izena)> 100){
75     $mezua = “Izen baliogabea”;
76     $mezua_type = “error”;
77  }…
```

Adierazi diren pausuak erabiltzaileak input-a sartu behar duen orri guztietan aplikatuz, inolako komandorik zuzenean datu basera sartzea ekiditu dugu. Guri interesatzen zaizkigun parametro zehatzak sartu daitezke soilik, eta gainera, parametrizazioarekin, oraindik gehiago bereizten dira SQL komandoak eta datuak.

2. Goiburuak

Web-konfigurazio lehenetsi batean, nabigatzaile askok erasotzaileek ustia ditzaketen portaerak baimentzen dituzte, adibidez kanpo script-ak kontrolik gabe kargatzea, orrialdeak hirugarrenen iframe-etan txertatzea, fitxategi baten mota zehatza asmatzea, nabigazio-goiburuetan informazio sentikorra bidaltzea, etab.

Horrek guztiak ahultasunak dakartza, hala nola eduki injekzioak, saio lapurretak, etab. Arrisku horiek murrizteko segurtasun goiburuak ezarri behar dira esplizituki adierazteko nabigatzaileari zein baliabide karga ditzakeen, nola jokatu behar duen eta zein ekintza blokeatu behar diren.

Esandakoa bermatzeko **.htaccess** artxiboan (testu laueko konfigurazio ezkutuaren fitxategia)goiburu batzuk gehitu ditugu. Segurtasun-goiburuak **.htaccess** fitxategian konfiguratu dira, webgune osoari modu globalean aplika dakizkion, PHP, HTML, irudia, etab. fitxategi bakoitzean eskuz gehitu behar izan gabe.

Hauek izango lirateke gehitutako goiburuak:

```
<IfModule mod_headers.c>
    # Anti-Clickjacking
    Header always set X-Frame-Options "DENY"

    # MIME sniffing ekiditu
    Header always set X-Content-Type-Options "nosniff"

    # XSS babes minimoa
    Header always set X-XSS-Protection "1; mode=block"

    # Referrer Policy goiburua
    Header always set Referrer-Policy "strict-origin-when-cross-origin"

    # Content Security Policy (CSP) goiburua
    Header always set Content-Security-Policy "default-src 'self';
    script-src 'self'; style-src 'self'; img-src 'self' data:; font-src
    'self'; connect-src 'self'; frame-ancestors 'none'; base-uri 'self';
    form-action 'self'; object-src 'none';"

    # Feature/Permissions Policy goiburua
    Header always set Permissions-Policy "geolocation=(self),microphone=(),
camera=()"
    # Remove server information headers
    Header always unset X-Powered-By
    Header unset X-Powered-By
    Header always unset Server
</IfModule>
```

Konfiguratutako HTTP goiburu hauen helburua webgunearen segurtasuna indartzea eraso komunen aurrean, eta nabigatzailearen sarbidea mugatzea arriskutsuak izan daitezkeen baliabide edo funtzionalitate jakin batzuetara da.

Gehitutako goiburuen arrazoiak:

- ***X-Frame-Options:*** *X-Frame-Options* goiburuak 'DENY' balioarekin, orrialdea beste edozein domeinuko iframe baten barruan kargatzea eragozten du eta horrela Clickjacking erasoetatik babesten du. Eraso mota hau, erasotzaile batek legezko orrialde bat beste baten gainean ezkutuan jartzen saiatzen da, erabiltzailea engainatzeko ekintzak egitea lortzeko erabiltzen da.

- ***X-Content-Type-Options:*** *X-Content-Type-Options:nosniff* goiburuak nabigatzailea fitxategien eduki mota "asmatzen" saiatzen da. Zerbitzariak adierazitako MIME mota errespetatzera behartzen du eta horrek manipulatutako fitxategiak exekutatzeko saiakerak edo kode-injekzioak blokeatzen ditu, modu faltsuan tipifikatutako baliabideen bidez.

- ***X-XSS-Protection:*** *X-XSS-Protection "1; mode=block"* goiburuak babes-mekanismo bat aktibatzen du nabigatzaile zaharretan *Cross-Site Scripting* (XSS) oinarrizko erasoak detektatzeko. Nabigatzaileak script injekzio-saiakera posible bat detektatzen duenean, blokeatzen du orrialdearen karga partzialki iragarrita erakutsi beharrean. Nabigatzaile modernoak babes horren mende egon ez arren, neurri gehigarri gisa erabilgarria izaten jarraitzen du bertsio zaharragoak erabiltzen dituzten erabiltzaileak dituzten inguruneetarako.

- ***Referrer-Policy:*** *Referrer-Policy: strict-origin-when-cross-origin* goiburuak erabiltzaile batek orrialde batetik bestera nabigatzen duenean 'Referer' goiburuan zenbat informazio bidaltzen duen kontrolatzen du. Politika horrekin, veste domeinu batzuetako eskaerak egitean, jatorriaren zatia bidaliko da soilik, URL-en ibilbide osoak edo baliozko datu sentikorrak ezkutatuz. Horrekin pribatasuna hobetuko da eta informazio filtroak saihestuko dira.

- ***Content-Security-Policy (CSP):*** *Content-Security-Policy (CSP)* goiburuak jatorri-kontroleko politika zorrotz bat definitzen du orriak kargatutako baliabide guztietarako.

  Zaztu den moduan: *'default-src 'self'; script-src 'self' 'unsafe-inline'; style-src 'self' 'unsafe-inline'; img-src 'self'; font-src 'self'; connect-src 'self'; frame-ancestors 'self';'*
  Honela, baliabideak gunearen domeinu beretik kargatzera behartzen du nabigatzailea (irudiak, script-ak, estiloak, iturriak, etab). Gainera, edukia kanpo iframeetan bistaratzea blokeatzen du. Hori dela eta, CSPa neurri eraginkorrenetako bat da XSS erasoen aurka, urruneko script-ak injektatzearen aurka edo hirugarrenek edukia manipulatzearen aurka.

- ***Feature-Policy:*** *Feature-Policy* goiburuak (oraingo bertsio modernoetan *Permissions-Policy*-z ordezkatuta) nabigatzailearen API batzuen erabilera mugatzen du. Kasu honetan geolokalizazioa domeinutik bertatik bakarrik ahalbidetuz '*self*' eta mikrofonorako sarbidea eta kamera '*none*' erabat blokeatuz, erabiltzailearen esposizioa baimenik gabeko sarreretara murriztuz.

- ***Header always unset X-Powered-By, Header unset X-Powered-By eta Header always unset Server:*** Konfigurazio ploke honetan *Header always unset X-Powered-By, Header unset X-Powered-By eta Header always unset Server* direktibak zerbitzariak bere buruari buruz bidal dezakeen edozein informazio ezabazteko erabiltzen dira. X-Powered-By-k, erabilitako teknologia adierazten du (adibidez, PHP edo haren bertsioren bat), eta *Server* goiburuak web zerbitzariaren mota eta bertsioa erakusten ditu. Horiek ezabatzeak ez du xehetasunik ematen erasotzaile batek bertsio espezifikoei lortutako ahuleziak identifikatzeko. *Unset* eta *always unset* konbinazioak bermatzen dute goiburu horiek ezabatu egingo direla, bai erantzun arrunten aurrean, bai errore-orrietan edo eduki estatikoen aurrean eta horrela, egoera guztietan babesa mantenduko dela bermatuko da.

3. Parametroen Manipulazioa

Web aplikazioetan ohiko ahultasunetako bat da erabiltzaileek URLaren bidez edo formularioetan bidalitako datuak behar bezala ez balioztatzeak eta ez garbitzeak eragiten du. Parametroen manipulazioak aukera ematen die erasotzaileei aplikazioaren logika aldatzeko edo kode kaltegarria injektatzeko eta horrek, besteak beste, SQL injekzio eta XSS (Cross-Site Scripting) erasoak eragin ditzake.

Proiektu honetan arrisku horiek murrizteko bi babes-mekanismo nagusi ezarri dira:

1. Datu-basearekin interakzio segurua: Prestatutako Sententziak (Prepared Statements)
   SQL injekzioa saihesteko modurik eraginkorrena prestatutako sententziak erabiltzea da teknika honen bidez:
   - Kontsultaren egitura eta datuak bereizita bidaltzen dira.
   - Sarrerako parametro guztiak datu huts gisa tratatzen dira eta ez dira inoiz SQL komando bihurtzen.

Proiektuko fitxategi guztietan teknika hau aplikatu da, mysql liburutegiko prepare() eta bind_param() funtzioak erabiliz.

add_item.php

```
$stmt = $conn->prepare("INSERT INTO pelikulak (izena,
deskribapena, urtea, egilea, generoa) VALUES (?, ?, ?, ?,
?)");
$stmt->bind_param("ssiss", $izena, $deskribapena, $urtea,
$egilea, $generoa);

if ($stmt->execute()) {
   $mezua = "Pelikula ondo gehitu da!";
} else {
   $mezua = "Arazo bat egon da: " . $stmt->error;
}
$stmt->close();
```

Sistema honek datu-basearen osotasuna bermatzen du eta injekzio bidezko erasoak guztiz neutralizatzen ditu.

2. Sarrerako datuen garbiketa

XXS erasoak saihesteko, erabiltzailearen sarrerako datuak nabigatzailean erakutsi baino lehen garbitu egin behar dira. Horretarako, htmlspecialchars() funtzioa erabili da modu simetrikoan.

modify_user.php

```
<input type="text" name="telefonoa" style="width: 100%;"
value="<?= htmlspecialchars($erabiltzailea['telefonoa'])
?>" required>
```

3. URL parametroen balidazioa

GET bidez jasotako balioak zuzenean erabiltzea arriskutsua da. Horregatik, parametro horiek zenbaki bihurtu eta balioztatzea egin da intval() erabiliz.

delete_item.php

```
$item_id = intval($_GET['item']);

    $stmt = $conn->prepare("SELECT * FROM pelikulak WHERE id
= ?");
    $stmt->bind_param("i", $item_id);
```

Honek saihestu egiten du erabiltzaile gaiztoek ID faltsuak, testuak edo komandoak txertatzea.

Parametroen kudeaketa segurua funtsezkoa da web aplikazio fidagarriak garatzeko. Proiektu honetan ezarritako neurriak aplikazioaren eta erabiltzaileen datuen segurtasuna bermatzeko daude pentsatuta. Mekanismo horiek eraginkortasunez babesten dute sistema SQL injekzio eta XSS bezalako web eraso ohikoenetatik.

4. Spectre

Spectre ahultasuna arakatzaileek erabiltzen duten mikroarkitektura-mailako exekuzio espekulatiboarekin lotuta dago. Horrek aukera ematen du web orrialde batek beste jatorri bateko datuak memoriatik irakurtzeko, baldin eta nabigatzaileak isolamendu nahikorik ez badu.

Spectre ahultasunari aurre egiteko, bi konponbide nagusi daude. Lehenengo aukerak isolamendu-maila maximoa eskaintzen du, baina konfigurazio konplexuagoa eskatzen du. Bigarren aukerak, ordea, CSP politika zorrotz baten bidez lortzen du segurtasun-maila nahikoa, gure kasuan egokiagoa izan den irtenbidea.

Lehenengo aukera: COOP + COEP + CORP politikak

Aukera hau hiru header bereziren konbinazioan datza:

- Cross-Origin-Opener-Policy "same-origin": Leiho berriak jatorri berekoak izan daitezen bermatzen du.
- Cross-Origin-Embedder-Policy "require-corp": Baliabide guztiek cross-origin politika argi bat izan behar dute.
- Cross-Origin-Resource-Policy "same-origin": Beste jatorrietako baliabideak kargatzea eragozten du.

Konfigurazio hau bereziki egokia da web aplikazio konplexuentzat, iframe-ak eta kanpoko baliabide asko erabiltzen dituztenentzat. Hala ere, desabantaila nagusi ditu: nabigatzaile zaharrekin bateraezina izan daiteke, eta aplikazioaren funtzionalitate batzuk apurtzeko arrisku dago, bereziki kanpoko API deiak edo integrazioak daudenean.

Bigarren aukera: CSP zorrotza (guk hautatutakoa)

Gure kasuan, bigarren aukera erabili genuen, hau da, CSP politika oso zorrotz bat ezartzea. Hau da gure konfigurazioa `.htaccess`-en:

```
Header always set Content-Security-Policy \
"default-src 'self'; \
 script-src 'self'; \
 style-src 'self'; \
 img-src 'self' data:; \
 font-src 'self'; \
 connect-src 'self'; \
 frame-ancestors 'none'; \
 base-uri 'self'; \
 form-action 'self'; \
 object-src 'none';"
```

Aukera hau hobeagoa izan zen guretzat, gure aplikazioak ez baitzituen kanpoko baliabiderik erabiltzen, ez iframe-ik eta ez zuen jatorri anitzeko komunikaziorik behar. COOP/COEP konfigurazio konplexua ezartzea "gehiegizkoa" izango zen, konplikazio gehiago ekarriko zizkigun arren segurtasun irabazirik esanguratsurik lortu gabe.

5. Cookie-ak txarto konfiguratuta

Nabigatzailean cookie-en konfigurazioari buruz agertzen ziren alertak zuzentzeko , zehazki '*Cookie Sin Flag HttpOnly'* eta '*Cookie sin el atributo SameSite*' alertak, PHP saioko cookie-ak nola sortzen ari ziren berrikusi behar zen. Arazoaren jatorria, proiektuak PHP 7.2 erabiltzen duela, SameSite atributua modu natiboan konfiguratzeko aukerarik ematen ez duen bertsioa da. *Session_start()* exekutatzean, PHPk automatikoki cookie bat bidaltzen zuen, HttpOnly, Secure eta SameSite atributurik gabea eta horrek nabigatzaile modernoetako alertak eragiten zituen.

Konpontzeko, fitxategi espezifiko bat sortzea erabaki zen, *'session.php'* izenekoa, saioa konfigurazio pertsonalizatua erabiliz hasteko.

Session.php fitxategia:

```
<?php

/**
 * Session initialization with SameSite support for PHP 7.2
 * Include this at the top of files that need sessions
*/
if (!headers_sent()) {
    // Configure session before starting
    ini_set('session.use_only_cookies', 1);
    ini_set('session.use_strict_mode', 1);
    ini_set('session.cookie_httponly', 1);

    session_start();

    $sessionName = session_name();
    $sessionId = session_id();

    header_remove('Set-Cookie');

    $cookieParams = session_get_cookie_params();
    $cookieValue = sprintf(
        '%s=%s; Path=%s; HttpOnly; SameSite=Strict',
        $sessionName,
        $sessionId,
        $cookieParams['path']
    );

 //Secure 'flag'-a gehitu HTTPS bada
     if (!empty($_SERVER['HTTPS']) && $_SERVER['HTTPS'] !==
'off') {
        $cookieValue .= '; Secure';
    }
     header('Set-Cookie: ' . $cookieValue, false);
    }  ?>
```

Fitxategi hau, saioaren cookiea eskuz sortzeaz arduratzen da, beharrezko segurtasun-ezaugarriekin. Edukiari dagokionez, nabigatzailera oraindik goibururik bidali ez dela egiaztatzen da, izan ere, aldez aurreko irteerarik balego, ezinezkoa izango litzateke cookie-a aldatzea. Ondoren, *ini_set()* bidezko hainbat konfigurazio aplikatzen dira, hala *nola session.use_only_cookies*, *session.use_strict_mode* eta *session.cookie_httponly.* Adibidez, *ini_set* lineak (*'session.cookie_httponly', 1)* cookiea HttpOnly gisa markatzea bermatzen du, JavaScript bidez eragotziz, eta horrek detektatutako alertetako bat ezabatzen du.

Esandako konfigurazioak doitu ondoren, *session_start()* exekutatzen da. Hala ere, funtzio honek automatikoki PHP 7.2-n osatu gabeko cookie bat bidaltzen du, beraz fitxategiak lehenetsitako goiburu hori ezabatzen du *header_remove ('Set-Cookie')* erabiliz, horrela, cookie okerra nabigatzailera bidaltzea saihesten da.

Jarraian, cookie-aren jatorrizko parametroak lortzen dira *session_get_cookie_params()* bidez eta cookie bat eskatuz berreraikitzen da, erikitzen den cookie hori ezaugarri egokiak ditu. Kodearen zati hori jarraibide honetan agertzen da:

```
$cookieValue = sprintf(
        '%s=%s; Path=%s; HttpOnly; SameSite=Strict',
        $sessionName,
        $sessionId,
        $cookieParams['path']
    );
```

Hemen, esplizituki gehitzen dira HttpOnly eta SameSite = Strict atributua, lehen ez zeudenak. Gainera, cookie-a kanal seguruen bidez bakarrik bidaltzen dela bermatzeko, Secure atributua gehitzen da konexioa HTTPS denean, honakoa erabiliz:

```
if (!empty($_SERVER['HTTPS']) && $_SERVER['HTTPS'] !== 'off')
{
        $cookieValue .= '; Secure';
    }
```

Azkenik, berreraikitako cookie-a eskuz bidaltzen da goiburuarekin:

```
  header('Set-Cookie: ' . $cookieValue, false);
```

Prozesu guztiari esker, saioaren cookie-a behar bezala konfiguratuta geratzen da eta nabigatzailearen abisuak desagertuko dira. *Includes/session.php* inplementatu ondoren, saioak erabili behar dituen orrialde bakoitzak hasieran fitxategi hau baino ez du barne hartzen. Adibidez, *index.php*-n: *requiere_once 'includes/session.php';*

Horrek bermatzen du proiektuaren orrialde bat kargatzen den bakoitzean saioa *session.php*-n ezarritako konfigurazio seguruarekin hastea, kodearen gainerako zatia aldatu beharrik gabe. Horrela, saioko cookie-ek behar bezala txertatzen dituzte HttpOnly eta SameSite = Strict atributuak, hasierako oharrak erabiliz eta aplikazioaren segurtasun orokorra hobetuz.

6. X-Powered-By

`X-Powered-By` headerra segurtasun-arriskua zen, gure teknologiari buruzko informazio sentikorra ematen zuelako, erabiltzen ari ginen PHPren bertsioa zehazki. Header hori automatikoki agertzen zen zerbitzariaren erantzun guztietan, bai PHP orrietan, bai fitxategi estatikoetan.

Hasieran, PHP mailako soluzio bat probatzen dugu `security.headers.php` fitxategian:

```
header_remove('X-Powered-By');
```

Hala ere, hurbilketa horrek muga garrantzitsu bat zuen: PHP kodea exekutatzen zuten orrietarako bakarrik funtzionatzen zuen, baina CSS bezalako fitxategi estatikoek, irudiek edo errore-orriek header errebelatzailea erakusten jarraitzen zuten.

Behin betiko soluzioa web zerbitzaria zuzenean `.htaccess` fitxategiaren bidez konfiguratuz aurkitzen dugu. Hemen inplementatzen dugu:

```
Header always unset X-Powered-By
Header unset X-Powered-By
Header always unset Server
ServerSignature Off
```

Konfigurazio horrek headerra erantzun guztietatik salbuespenik gabe ezabatzen dela ziurtatzen du, web zerbitzariaren mailan jarduten baitu edozein eduki bezeroarengana iritsi aurretik. Neurri hori osatzeko, zerbitzariaren informazioa ezkutatuko dugu `ServerSignature Off`-ekin, erasotzaile bati gure konfigurazioaren urrakortasun espezifikoak identifikatzen lagun diezaiokeen edozein arrasto ezabatzeko.

Gakoa izan zen `always unset` erabili ordez, besterik gabe, `unset`, hau header ezabatzea, nahiz eta errore-erantzunak eta salbuespenezko kasuetan non beste direktibak agian ez dira aplikatuko.

7. Server goiburua (zerbitzariaren informazio-ihesa)

Apache zerbitzariak eta PHP-k automatikoki gehitzen zituzten Server eta X-Powered-By header-ak, zeinek beraien bertsioen informazioa erakusten zuten. Informazio hau kaltegarria zen segurtasunarentzat, erasotzaileei zehazki zein bertsio eta teknologia erabiltzen genituen jakiteko aukera emanez, eta horrela bertsio horietako ahultasun espezifikoak erabiltzeko.

Ahultasun hau konpontzeko hiru geruzatako konponbidea erabili genuen, bakoitzak bere aldetik babesten duena:

- `no-expose.ini`-en aldaketak:

  ```
  expose_php = Off
  server_tokens = Off
  ```

  `expose_php: X-Powered-By` headerra erabat ezabatzen du.

  `server_tokens`: PHP-ren bertsioa ezkutatzen du Server headerretik.

- `.htaccess`-en aldaketak:

  ```
  Header always unset Server
  ServerSignature Off
  ```

  `Header always unset Server`: Server headerra guztiz ezabatzen du.

  `ServerSignature Off`: Error orrialdeetan zerbitzariaren informazioa ezkutatzen du.

- `no-expose.ini`-en aldaketak:

  ```
  header_remove("X-Powered-By");
  ```

  PHP mailako segurtasun neurri gehigarria.

Hiru geruzako estrategia hau erabili genuen erredundantzia maximoa lortzeko geruza batek huts egiten badu, besteek jarritako babesa mantentzen da, osotasuna ziurtatzeko, PHP kodea exekutatzen ez duten CSS edo irudi bezalako fitxategiak ere babesteko, eta error kasuak estaltzeko, erabiltzaileari inoiz ez erakustea zerbitzariaren informazioa.

8. Banner-raren edo Footer-raren informazio-ihesa

Apache zerbitzariak bere erantzunetan eta errore-orrietan informazio tekniko gehigarria bistaratzen du (adibidez: `Apache/2.4.25 (Debian) Server at localhost Port 80`). Informazio honek zerbitzariaren izena, bertsioa eta ingurune teknikoa agerian uzten ditu. Banner informazioak erasotzaileei sistema eragilea eta bertsioa erakusten die, hauen aurka zuzendutako erasoak erraztuz.

Konponbidea:

1. **Apache konfigurazioa eguneratu Dockerfile-n:**

```
RUN echo “ServerTokens Prod” >> /etc/apache2/apache2.conf\
&& echo “ServerSignature Off” >> /etc/apache2/apache2.conf
```

- ServerTokens Prod: Zerbitzariak soilik "Apache" erakutsiko du (ez bertsioa ezta OS informaziorik).
- ServerSignature Off: Errore-orrietan zerbitzariaren informazioa ezkutatzen du.

2. **Errore-orri pertsonalizatua sortu:**
`errorea.html` fitxategia:

```
<h1>404 - Orria ez da aurkitu</h1>
```

Nahiz eta lehen Apacheko errore-orri estandarraren informazio gehigarria ezkutatu zen, errore-orri propioa sortu da, sistemaren informaziorik gabe, segurtasun handiagoa izateko.

3. **Errore orri pertsonalizatua ezartzea Dockerfile-n:**

```
RUN echo "ErrorDocument 404 /errorea.html" >>
/etc/apache2/apache2.conf
```

Lehen, 404 errore bat gertatzean Apache-ren orri estandarra agertzen zen. Orain, erabiltzaileari errorea.html fitxategi propioa erakusten zaio, mezu sinple batekin (404 - Orria ez da aurkitu).

9. Cache-an Gorde Daitezken Datuak

Cache mekanismoek web aplikazioaren errendimendua nabarmen hobetzen dute, baina segurtasunez konfiguratzen ez badira, datu sentikorren esposizioa eragin dezakete. Nabigatzaileak edo proxy zerbitzariek saioarekin lotutako orrialde pribatuak gordetzen badituzte, beste erabiltzaile batek informazio konfidentziala atzitu ahalko luke, nahiz eta jatorrizko erabiltzaileak saioa itxi.

Hori saihesteko, proiektu honetan cache-kontrol zorrotza ezarri da saioarekin eta datu pertsonalekin lotutako orrialdeetan.

Cache kontrol goiburuak ezartzea

Segurtasun-maila altua behar duten orrialdetan berrabiarazi behar denean, cachea erabiltzea eragozten duten HTTP goiburuk hauek gehitu dira:

```
header("Cache-Control: no-store, no-cache, must-revalidate, max-age=0");
header("Pragma: no-cache");
header("Expires: 0");
```

Cache-Control: no-store, no-cache, must-revalidate, max-age=0

Goiburu honek nabigatzaileari argi eta garbi adierazten dio ez duela edukiaren kopiarik gorde behar. Bereziki, *no-store* aukerak debekatu egiten du edukiaren kopia bat cachean edo diskoan gordetzea eta *must-revalidate* aukerak behartzen du beti bertsio freskoa eskatzea. Horrela, erabiltzailearen datuak ez dira inolaz ere gailuan gordeko.

Pragma: no-cache

Goiburu hau HTTP/1.0 bezeroekin bateragarritasuna bermatzeko erabiltzen da. Nahiz eta modernoki Cache-Control nahikoa izan, Pragma gehitzeak antzerako bateragarritasuna ziurtatzen du.

Expires: 0

Goiburuak adierazten du orrialdea berehala iraungita dagoela. Horrek nabigatzaileari esaten dio ezin duela kopia zaharra eta derrigorrez eskatu behar duela bertsio eguneratua

Goiburu horiek aplikatuta erabiltzailearen datu konfidentziala erakusten duten orrialdeek ez dute informazio hori cachean uzten eta horrela saioa amaitu ondoren orria ezin da berriro atzitu, gailu partekatuetan datu pertsonalak ez dira “Atzera” botoiarekin ikusiko eta proxy edo nabigatzaileek ez dute kopiarik gordeko. Ezarpen honek erabiltzaileen pribatutasuna eta saioaren segurtasuna bermatzen ditu, cachearen bidezko informazio-ihes arriskua guztiz minimizatuz.

10. Cache-an Gorde Ezin Daiteken Informazioa

ZAP-ek arakatu eta gorde ezin daitekeen edukiari buruz detektatutako alerta ez da benetako kaltegaberatasuna, segurtasun-iruzkin edo segurtasun-gomendio bat baizik. Ez dago cachean biltegiratutako datu sentikorrik, ezta erabiltzaileen informazio-esposizioaren arriskurik ere. Beraz, ez da zuzendu beharreko akatsa, baizik eta miaketa egiteko jardunbide egokiei buruzko ohar informatiboa.

Hala ere, gure kasuan cache-kontrola dago inplementatuta `session.php` fitxategian. PHP-ren `session_start()` funtzioak automatikoki gehitzen ditu cache-kontrolerako header-rak saioa hasten denean:

```
// session.php fitxategian
if (!headers_sent()) {
    // Configure session before starting
    ini_set('session.use_only_cookies', 1);
    ini_set('session.use_strict_mode', 1);
    ini_set('session.cookie_httponly', 1);

    // Start session - HONEK AUTOMATIKOKI GEHITZEN DITU CACHE-
HEADERRAK:
    // Cache-Control: no-cache, must-revalidate
    // Expires: (data iraungitua)
    // Pragma: no-cache
    session_start();

    $timeout = 60;
}
```

Gure aplikazioan orrialde sentikor guztiek (`login.php`, `register.php`, etab.) `session.php` inportatzen dute:

```
// login.php eta register.php fitxategietan
require_once 'includes/session.php';
```

Eta horrek bermatzen du nabigatzaileak ez duela informazio sentikorrik cache-an gordeko. Beraz, alerta hau lehen agertu arren, gure sistemak jada badu cache-an gorde ezin daitekeen informazioa babesteko mekanismo egokia inplementatuta.

11. Informazioaren espozizioa – URL-tan informazio sentikorra

Web aplikazioa batek URL-en bidez datuak transmititzen dituenean, kontuan hartu behar da mekanismo honek berezko arriskuak dituela. GET parametroak bistakoak dira eta hainbat tokitan gordetzen dira automatikoki eta horrek informazio sentikorra isurtzeko edo manipulatzeko aukera eman dezake. Hauek dira arrisku nagusiak:

- Information Discloseure: Pasahitzak, saio-tokenak edo datu pribatuak URL-an agertzen badira, informazio hori kontrolik gabeko moduan gordetzen da hainbat sistematan.
- Tampering: URL-ko parametroak oso erraz manipula daitezke. Horrela, erasotzaile batek beste erabiltzaile batzuen erregistroetara sartzen saiatu daiteke edo aplikazioaren logika ustiatu.

Arrisku hau konpondu ahal izateko hurrengo pausuak jarraitu ditugu:

1. URL parametroen erabilera mugatua
   Proiektu honetan arrisku horiek murrizteko, URL-en bidez transmititzen diren datuak ahalik eta gutxienekoetara murriztu dira eta ondorengo irizpideak ezarri dira:
   - Datu sentikorretarako POST metodoa erabiltzea
     Pasahitzak, saio-datuak eta beste edozein informazio sentikor ez dira inoiz URL-en bidez bidaltzen- Horiek beti HTTP gorputzean bidaltzen dira POST metodoa erabiliz, bistaratzea eta isurketa ekidinez.
   - ID-ak soilik URL bidez bidaltzea
     URL-an transmititzen diren parametro bakarrak erregistroen identifikatzaileak dira. Hauek ez dira datu sentikorra eta gainera beti zenbaki oso gisa tratatzen dira.
2. ID parametroen balioztatzea
   Erregistroen identifikatzaileak URL-an bidaltzen direnean ezinbestekoa da parametro horiek balioztatzea datu-basean erabili aurretik. Horretarako, intval() erabiltzen da balioa zenbaki oso segurua dela ziurtatzeko. Prestatutako sententziekin batera erabiltzen da, manipulazio saiakerak eta injekzioa arriskuak desaktibatzeko.

```
$item_id = intval($_GET['item']);

    $stmt = $conn->prepare("SELECT * FROM pelikulak WHERE id = ?");
    $stmt->bind_param("i", $item_id);
    $stmt->execute();
```

Balioztatze-mekanismo hau parametroen manipulazioa atalean azaldu den prestatutako sententzien erabileraren osagarri zuzena da eta URL bidezko manipulazio gaiztoak saihesten ditu.

.

12. Saio-kudeaketaren erantzun identifikatua

Lehen, web aplikazioaren erantzunetan saio-tokenak agertzen ziren edo ez ziren behar bezala babestuta.

- Token horiek erabiltzaileen saioak identifikatzeko erabiltzen dira.
- Desegoki erabilita, erasotzaile batek beste erabiltzaile baten saioa eskuratu edo saioaren jarraipena manipulatu zezakeen.
- Tokenak ezkutatu gabe agertzen baziren, informazio sentikorra arriskuan egongo litzateke.

Konponbidea:

Beheko kodea saioa hasteko modulu gisa sortu da eta index.php, login.php, modify.php fitxategietan sartu da.

3. **Saioaren denbora-muga ezartzea**:

```
$timeout = 60;
if (isset($_SESSION['last_activity']) && (time() -
$_SESSION['last_activity'] > $timeout)) {
      session_unset();
      session_destroy();
      exit;
}
$_SESSION['last_activity'] = time();
```

Erabiltzaileak 60 segundoko inaktibitatearen ondoren saioa automatikoki itzaltzen du. Horrek saio-tokenaren gehiegizko erabilera saihesten du.

4. **Cookie seguruak eta SameSite ezarpena:**

```
header_remove('Set-Cookie'); // Ezabatu cookie zaharrak
$cookieValue = sprintf(
    '%s=%s; Path=%s; HttpOnly; SameSite=Strict',
    $sessionName,
    $sessionId,
    $cookieParams['path']
);
if (!empty($_SERVER['HTTPS']) && $_SERVER['HTTPS'] !==
'off') {
    $cookieValue .= '; Secure';
}
header('Set-Cookie: ' . $cookieValue, false);
```

HttpOnly → JavaScript-ek ezin du cookiea irakurri (XSS babesa).
Secure → cookiea HTTPS bidez soilik bidaltzen da.
SameSite=Strict → kanpoko webguneek ezin dute cookiea erabili (CSRF babesa).

5. **Saio ID berria sortzea saio hasieran:**

```
session_regenerate_id(true);
$_SESSION['initiated'] = true;
```

session_regenerate_id(true) erabiltzen da saio ID-a berritzeko saioa hasi bezain laster. Horrek saio-token zaharrak baliogabetzen ditu eta saioaren lapurreta zailtzen du. Hau ez dago session.php fitxategian, login.php fitxategian baizik.

Beste segurtasun hobekuntzak

1. Pasahitzen Babesa

Aplikazioan erabiltzaileen pasahitzak datu-basean gordetzen dira inolako hash edo zifratzerik gabe. Hau ikus daiteke `register.php` fitxategian, adibidez, non erabiltzailearen datuak zuzenean sartzen diren datu-basean.

Konponbidea erraza izan da, pasahitzaren datua hartzen dugun bakoitzean hash bat egitea erabaki da, honela, datu-basea edo komunikazioak filtratuko balira ere pasahitzak ez lirateke eskuragarriak izango.

```
$hashed_password = password_hash($pasahitza,
PASSWORD_DEFAULT);

$stmt = $conn->prepare("INSERT INTO usuarios (nombre,
nan, telefonoa, jaiotze_data, email, pasahitza) VALUES
(?, ?, ?, ?, ?, ?)");

$stmt->bind_param("ssssss", $izena, $nan, $telefonoa,
$data, $email, $hashed_password);
```

2. HTTP Segurua

Aplikazioa soilik HTTP bidez komunikatzen da, eta ez du HTTPS erabiltzen. Horrek esan nahi du datuak (login-ak, cookie-ak, token-ak) testu arruntean bidaltzen direla sarean, eta horrek Man-in-the-Middle (MitM) erasoak ahalbidetzen ditu.

Jatorria:

- docker-compose.yml-ean soilik 80 portua mapeatuta dago.
- Dockerfile-ean ez da SSL modulua aktibatzen ezta ziurtagiririk konfiguratzen.

Konponbidea:

Web sistemari **SSL/TLS zifraketa** gehitu diogu HTTPS aktibatuz. Horretarako, Apache-n beharrezko moduluak gaitu eta ziurtagiri bat sortu dugu.

1. **Apache SSL moduluak aktibatzea:**

```
RUN a2enmod ssl && a2enmod rewrite
```

   - ssl: HTTPS konexioak ahalbidetzen ditu.
   - rewrite: HTTPtik HTTPSra birbideratzeko erabiltzen da.

2. **Ziurtagiri digitala sortu (SSL):**

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048
-keyout localhost.key -out localhost.crt -subj
"/C=US/ST=Bizkaia/L=Bilbao/O=EHU/CN=localhost"
```

   Honako bi fitxategi sortzen ditu:
   - /etc/ssl/private/apahce-selfsigned.key → giltza pribatua
   - / etc/ssl/certs/apahce-selfsigned.crt → ziurtagiri publikoa

   Ziurtagiri hau urtebetez baliozkoa da, eta gure zerbitaria identifikatzeko erabiltzen da. Autofirmatua da, ez da CA ofizial batek emandakoa, horregatik, nabigatzailean “Ez segurua” agertzen da.

3. **Fitxategiak Docker irudian gehitzea:**

```
COPY ssl/localhost.crt /etc/ssl/certs/
COPY ssl/localhost.key /etc/ssl/private/
```

4. **Apache konfigurazioa (ssl-config.conf):**
   ssl-config.conf fitxategia sortu:

```
<VirtualHost *:80>
  ServerAdmin webmaster@localhost
  DocumentRoot /var/www/html
  Redirect "/" "https://localhost:8443/"
```

```
</VirtualHost>

<VirtualHost *:443>
  ServerAdmin webmaster@localhost
  DocumentRoot /var/www/html
  SSLEngine on
  SSLCertificateFile /etc/ssl/certs/localhost.crt
  SSLCertificateKeyFile /etc/ssl/private/localhost.key
</VirtualHost>
```

Fitxategi honen lehen zatian, 80. portutik (HTTP) sartzen den trafikoa automatikoki birbideratzen da HTTPS-ra. Bigarren zatian, berriz, bere funtzioa da 443. portutik (HTTPS) sartzen den trafikoa zifratuta kudeatzea, guk sortutako ziurtagiria erabiliz.

Ondoren, fitxategi hau kontainer barrura gehitzen da:

```
COPY ssl-config.conf /etc/apache2/sites-available/000-
default.conf
```

Azkenik, HTTPS-aren portua gehitzen da:

```
<VirtualHost *:80>
  ServerAdmin webmaster@localhost
  DocumentRoot /var/www/html
  Redirect "/" "https://localhost:8443/"
</VirtualHost>

<VirtualHost *:443>
  ServerAdmin webmaster@localhost
  DocumentRoot /var/www/html
  SSLEngine on
  SSLCertificateFile /etc/ssl/certs/localhost.crt
  SSLCertificateKeyFile /etc/ssl/private/localhost.key
</VirtualHost>
- "8443:443"
```

Konpondu beharreko ahuleziak

Proiektuan aztertu beharreko segurtasun-alerten artean, "*Osagai kalteberak eta zaharkituak*" izenekoa ez zen gure web sistemaren alerta zerrendan agertu. Horregatik ez da alerta hori konpontzeko neurririk hartu, izan ere, ez zen inolako alertarik jaso zaharkitutako edo kalteberak ziren osagaien inguruan. Gainerako detektatutako alertak identifikatu eta behar bezala konpondu dira.

Erreferentziak

Danilo Ivanovic 1 *et al.* (2019) *How to fix 'set samesite cookie to none' warning?*, *Stack Overflow*. Eskuragarri: https://stackoverflow.com/questions/58191969/how-to-fix-set-samesite-cookie-to-none-warning (2025eko azaroaren 14an aipatuta).

Apache Module mod_headers (no date) *mod_headers - Apache HTTP Server Version 2.4*. Eskuragarri: https://httpd.apache.org/docs/current/mod/mod_headers.html (2025eko azaroaren 14an aipatuta)

CSP (no date) *ZAP By Checkmarx*. Eskuragarri: https://www.zaproxy.org/docs/alerts/10055/ (2025eko azaroaren 14an aipatuta).

(No date) *Chatgpt*. Eskuragarri: https://chatgpt.com/ (2025eko azaroaren 14an aipatuta).

Domain India (n.d.) Master PHP Sessions with Ease: A Comprehensive Developers Handbook. Eskuragarri: https://www.domainindia.com/login/knowledgebase/691/Master-PHP-Sessions-with-Ease-A-Comprehensive-Developers-Handbook.html (2025eko azaroaren 14an aipatuta)

Tshenolomos, T. (n.d.) Secure Apache with SSL in Docker. Eskuragarri: https://medium.com/@tshenolomos/secure-apache-with-ssl-in-docker-9efd86329129 (2025eko azaroaren 14an aipatuta)

Ask Ubuntu (2012) Setting ServerTokens and ServerSignature in Apache. Eskuragarri: https://askubuntu.com/questions/184365/setting-servertokens-and-serversignature-in-apache (2025eko azaroaren 14an aipatuta)