Kudeaketaren eta Informazio Sistemen Informatikaren Ingeniaritzako Gradua

Maskoten erregistroa

Entrega 3. txostena

Informazio Sistemen Segurtasuna Kudeatzeko Sistemak

Egileak:

Lucia Del Rio, Mikel Eguia
Olatz Elejalde, Asier Las Hayas
Maider Tato, Ainhoa Tomás

Irakaslea:

Mikel Egaña Aranguren

November 14, 2025

Contents

1 Sarrera **2**

2 Ahultasunen konponketa **3**

2.1 Absence of Anti-CSRF Tokens . . . 3
2.1.1 Arazoa . . . 3
2.1.2 Konponbidea . . . 3
2.1.3 Github-eko commit-a . . . 3
2.1.4 Emaitza . . . 4
2.2 Content Security Policy (CSP) Header Not Configured . . . 4
2.2.1 Arazoa . . . 4
2.2.2 Konponbidea . . . 4
2.2.3 Github-eko commit-a . . . 4
2.2.4 Emaitza . . . 5
2.3 Missing Anti-Clickjacking Header . . . 5
2.3.1 Arazoa . . . 5
2.3.2 Konponbidea . . . 5
2.3.3 Github-eko commit-a . . . 5
2.3.4 Emaitza . . . 5
2.4 Cookie Without HttpOnly Flag . . . 5
2.4.1 Arazoa . . . 5
2.4.2 Konponbidea . . . 6
2.4.3 Github-eko commit-a . . . 6
2.4.4 Emaitza . . . 6
2.5 Cookie Without SameSite Attribute . . . 6
2.5.1 Arazoa . . . 6
2.5.2 Konponbidea . . . 7
2.5.3 Github-eko commit-a . . . 7
2.5.4 Emaitza . . . 7
2.6 Information Disclosure - Sensitive Information in HTTP Response Headers (X-Powered-By) . . . 8
2.6.1 Arazoa . . . 8
2.6.2 Konponbidea . . . 8
2.6.3 Github-eko commit-a . . . 8
2.6.4 Emaitza . . . 8
2.7 Information Disclosure - Sensitive Information in HTTP Response Headers (Server) 9
2.7.1 Arazoa . . . 9
2.7.2 Konponbidea . . . 9
2.7.3 Github-eko commit-a . . . 10
2.7.4 Emaitza . . . 10
2.8 Missing X-Content-Type-Options Header . . . 10
2.8.1 Arazoa . . . 10
2.8.2 Konponbidea . . . 10
2.8.3 Github-eko commit-a . . . 11
2.8.4 Emaitza . . . 12

1 Sarrera

Entrega honen helburua aurreko entregan OWASP ZAP tresnaren bidez identifikatutako ahultasunak zuzentzea eta web sistemaren segurtasuna indartzea da. Bigarren entregan detektatutako arazo nagusiak (CSRF babesaren gabezia, HTTP goiburu ahulak eta saioko cookien konfigurazio ez egokia) konpondu dira, aplikazioaren fidagarritasuna, konfidentzialtasuna eta integritatea bermatzeko.

Lan honetan, egindako aldaketak eta hobekuntzak azalduko dira, eta konponketa ondoren egindako egiaztapenen emaitzak baita ere. Helburu nagusia web sistema seguruagoa, sendoagoa eta segurtasun-irizpide eguneratuekin bat datorren aplikazio bat lortzea da, OWASP Top 10 gomendioetan oinarrituta.

2 Ahultasunen konponketa

2.1 Absence of Anti-CSRF Tokens

2.1.1 Arazoa

Aplikazioak ez du CSRF tokenik erabiltzen datu sentikorrak kudeatzen dituzten galdetegietan (login, erregistroa edo antzeko formularioetan). Token hori ez izateak aukera ematen die erasotzaileei Cross-Site Request Forgery (CSRF) motako erasoak egiteko, erabiltzailea bere borondatearen aurka ekintzak egitera behartuz.
Horrek arriskuan jar dezake erabiltzailearen saioaren osotasuna, eta baimenik gabeko sarbideak eragin ditzake, hala nola kontuaren konfigurazioa aldatzea, pasahitza berritzea edo datu pribatuak eskuratzea. Ondorioz, sistema osoaren segurtasuna eta erabiltzaileen datuen konfidentzialtasuna arriskuan jarri daitezke, baita aplikazioaren fidagarritasuna ere.

2.1.2 Konponbidea

- Existitzen ez bada, sesioan CSRF tokena sortu behar da.
- HTML kodearen formularioetan izkutatutako eremua gehitu behar da.
- 'POST' baldintza jasotzen denean CSRF tokena existitzen dela eta zuzena dela egiaztatu behar da.

```
<?php
session_start();
...
if (empty($_SESSION['csrf_token'])) {
    $_SESSION['csrf_token'] = bin2hex(random_bytes(32));
}
...
if ($_SERVER['REQUEST_METHOD'] === 'POST') {

    if (!isset($_POST['csrf_token']) || !hash_equals($_SESSION['csrf_token'],
    $_POST['csrf_token'])) {
        die("CSRF token invalid or missing.");
    }
    ...
}
?>
<form ...>
    <input type="hidden" name="csrf_token" value="<?php echo
    htmlspecialchars($_SESSION['csrf_token']); ?>">
<\form>
```

2.1.3 Github-eko commit-a

Arazo hau konpontzeko 2 commit behar izan ditugu, izan ere, lehenengo commit-ean arazoa ez zen guztiz konpondu, eta 'POST' baldintzan CSRF-a egiaztatu ordez, beti egiaztatzen zuen.

https://github.com/LUC1A05/SEGUR_PROIEKTUA/commit/fd164cc0446e9e589226d9959253be229898da4b
https://github.com/LUC1A05/SEGUR_PROIEKTUA/commit/90e1b62f5c5f0dc230f1d1ea694910caecaf5dd9

2.1.4 Emaitza

CSRF tokena gehitu ondoren, aplikazioak formulario bakoitzaren eskaera bidezkoa dela egiaztatzen du, token bakarra erabiliz. Token hori ausazkoa eta erabiltzaile bakoitzarentzan bakarra denez, ezin da aurreikusi eta ezin dira erraz aldaketak egin. Horri esker, kanpoko domeinu edo webgune batek ezin du erabiltzailearen izenean eskaera faltsurik bidali.
Aldaketa hauek egin ondoren, ZAP berriro exekutatu da eta egiaztatu da CSRF motako ahultasuna ez dela berriro agertzen. Beraz, orain aplikazioak eskaeren jatorria eta bidezkotasuna behar bezala balioztatzen ditu, eta datu sentikorrak kudeatzen dituzten formularioak behar bezala babestuta geratu dira.

2.2 Content Security Policy (CSP) Header Not Configured

2.2.1 Arazoa

CSP (Content Security Policy) webgune batean segurtasun-geruza gehigarria da, eta hainbat motatako erasoei aurre egiteko diseinatuta dago, besteak beste Cross-Site Scripting (XSS) eta datuen injezioko erasoak. Horrelako erasoak datuak lapurtzeko, webguneen edukia manipulatzeko, edo malwarea zabaltzeko erabiltzen dira.
CSPk webguneen jabeei aukera ematen die orrialde jakin batek nabigatzailearen bidez zein iturritako edukiak kargatu ahal dituen zehazteko. Horrela, nabigatzaileak soilik baimendutako jatorrietatik datozen baliabideak kargatuko ditu, eta baimenik gabeko scriptak edo eduki arriskutsuak exekutatzea saihestuko da.
CSPren bidez kontrola ditzakeen eduki-motak honako hauek dira: JavaScript kodea, CSS estiloak, HTML markoak (frames/iframes), letra-tipoak, irudiak eta objektu txertagarriak, hala nola Java appletak, ActiveX osagaiak, edo audio eta bideo fitxategiak.

2.2.2 Konponbidea

Arazoa ematen zen goiburua konfiguratuta ez zegoelako. Hauek konfiguratzeko, .php fitxategietan goiburua definitu behar da.

```
<?php
...
session_start();
header("Content-Security-Policy: default-src 'self'; script-src 'self';
style-src'self'; img-src 'self' data:; font-src 'self'; object-src 'none';
frame-ancestors'none'; base-uri 'self'; form-action 'self';");
...
?>
```

2.2.3 Github-eko commit-a

https://github.com/LUC1A05/SEGUR_PROIEKTUA/commit/292edd6aaf1f41592410738ca41678ece0790c70

2.2.4 Emaitza

CSP goiburua gehitu ondoren, web orria hainbat erasogandik babestuta egongo da. Hala nola, Cross Site Scripting eta datu injekzioen erasoak.

2.3 Missing Anti-Clickjacking Header

2.3.1 Arazoa

Webguneak ez du anti-clickjacking goibururik ezartzen, eta horrek aukera ematen du orrialdea kanpoko webgune batek iframe baten barruan kargatzeko, erabiltzailearen baimenik gabe. Egoera horrek clickjacking izeneko erasoa errazten du.
Clickjacking-ean, erasotzaileak bere webgunean iframe baten bidez gure webgunea txertatzen du, begi bistatik ezkutatuta. Ondoren, erabiltzaileari beste elementu baten gainean klik egiten ari dela sinestarazten dio, baina benetan gure webguneko elementu sentikorretan klik egiten ari da. Horrela, erabiltzaileak bere borondatearen aurka ekintza kaltegarriak edo baimendugabeak egin ditzake, adibidez: botoi garrantzitsuak sakatzea (kontua ezabatzea, konfigurazioa aldatzea, saioa ixtea...), galdetegiak bidaltzea edo transakzio edo ekintza sentikorrak berretsi gabe burutzea.
Babes-neurri egokiak ezartzen ez badira, webgunea eraso horien aurrean askoz ere ahulagoa da. Anti-clickjacking goiburu egoki bat konfiguratuz (adibidez, X-Frame-Options edo Content-Security-Policy: frame-ancestors), arriskua murriztu eta baimenik gabeko webguneek orria iframe bidez kargatzea saihestu daiteke.

2.3.2 Konponbidea

Aurrekoan bezala, konponbidea goiburua definitzea da.

```
<?php
...
session_start();
header("Content-Security-Policy: default-src 'self'; script-src 'self';
style-src'self'; img-src 'self' data:; font-src 'self'; object-src 'none';
frame-ancestors'none'; base-uri 'self'; form-action 'self';");
...
?>
```

2.3.3 Github-eko commit-a

https://github.com/LUC1A05/SEGUR_PROIEKTUA/commit/292edd6aaf1f41592410738ca41678ece0790c70

2.3.4 Emaitza

Goiburua definitu ondoren, erabiltzaileak Clickjacking edo click bahiketa erasoaz babesten dira.

2.4 Cookie Without HttpOnly Flag

2.4.1 Arazoa

Aplikazioak sortzen duen saio cookieak ez du HttpOnly atributua ezarrita, eta horrek esan nahi du cookie hori JavaScript bidez irakur daitekeela. Egoera horrek arrisku handia sortzen du, batez ere XSS (Cross-Site Scripting) ahultasun baten aurrean.

HttpOnly atributurik gabe, cookiea document.cookie bidez eskura daiteke, eta horrek erasotzaileari aukera ematen dio webgunera script maltzur bat txertatuz erabiltzailearen saio identifikatzailea lapurtzeko. Behin cookie hori eskuratuta, erasotzaileak erabiltzailearen saioan sartzeko, bere izenean ekintzak egiteko, datu pertsonalak eskuratzeko edo kontuaren funtzio kritikoak aldatzeko gaitasuna izango du.
Horrelako erasoak oso larriak dira, eta askotan saio bahiketa (session hijacking) eragiten dute, erabiltzailearen nortasuna ordezkatzeko aukera emanez. Beraz, saio cookieetan HttpOnly atributua ez ezartzeak segurtasun arrisku serioa sortzen du.

2.4.2 Konponbidea

Cookie-n balioak definitzea:

```
<?php
ini_set('session.cookie_httponly', 1);
ini_set('session.cookie_secure', 0);
ini_set('session.use_only_cookies', 1);
...
?>
```

2.4.3 Github-eko commit-a

Commit honetan cookie-n balioak definitu ziren baina cookie_secure balioa 1-ean egon beharrean 0-n egon behar zen, izan ere, web orrialdea http da eta 1 balioa https orrientzako da. Hurrengo commit-ean akats hori konpondu zen:
`https://github.com/LUC1A05/SEGUR_PROIEKTUA/commit/16b26e7cbdf43d7623eee9adfb6fc727dfcaa0a6`

2.4.4 Emaitza

HttpOnly atributua aktibatzean lortu dena saio-cookieak JavaScript bidez irakur ezin izatea izan da. Horrek zuzenean murrizten ditu XSS (Cross-Site Scripting) erasoen arriskuak, eta erasotzaileek ezin dute cookiea document.cookie erabilita lapurtu.
Laburbilduz, HttpOnly atributua aktibatzeak Cookie Without HttpOnly Flag ahultasuna konpontzen du, eta saioaren segurtasuna modu eraginkorrean hobetzen du.

2.5 Cookie Without SameSite Attribute

2.5.1 Arazoa

Webguneak sortzen duen cookie bat SameSite atributurik gabe bidaltzen da. Horrek esan nahi du cookie hori beste webgune batetik egiten diren eskaeretara ere bidal daitekeela. Hau arriskutsua da, batez ere CSRF (Cross-Site Request Forgery) erasoen aurrean. CSRF-k aukera ematen dio erasotzaile bati erabiltzailearen saioa erabiliz webgunean ekintza ez baimendunak burutzeko, adibidez: formularioak bidaltzea, kontuaren ezarpenak aldatzea edo transakzioak egitea.
Erabiltzaile batek gure webgunean saioa hasi ondoren beste webgune arriskutsu bat bisitatzen badu, eta cookie hori eskaera horietan bidaltzen bada, erasotzaileak gure webguneko ekintzak bere saioan egiten ari dela sinestaraziko dio zerbitzariari, eta horrek ekintza faltsuak eragin ditzake.

2.5.2 Konponbidea

Aurrekoan bezala, cookie-n balioak definitzean datza arazo honen konponketa.

```
<?php
...
if (isset($_COOKIE[session_name()])) {
    setcookie(
        session_name(),
        session_id(),
        0,
        '/; samesite=Lax',
        '',    // dominio
        false, // secure
        true   // httponly
    );
}
...
?>
```

Aipatzekoa da, index.php fitxategian konfigurazioa ezberdina:

```
$lifetime = 0;
$path = '/; samesite=Lax';
$secure = false;
$httponly = true;
```

2.5.3 Github-eko commit-a

`https://github.com/LUC1A05/SEGUR_PROIEKTUA/commit/32e8cd79e57087b8cb78cfb3a375feb5110485b6`
index.php:
`https://github.com/LUC1A05/SEGUR_PROIEKTUA/commit/ca00527a09b8e624b2a6ba9ff3d66b2bff5128ee`

2.5.4 Emaitza

SameSite=Lax atributua gehitzeak beste webgune batetik datozen eskaeretan cookieak ez bidaltzea eragiten du. Horrek esan nahi du erabiltzaileak kanpoko webgune arriskutsu bat bisitatzen badu ere, erasotzaileak ezin izango dituela bere saio-cookieak erabili gure webgunera eskaera faltsuak bidaltzeko.
Ondorioz:

- CSRF (Cross-Site Request Forgery) erasoak eragitea askoz zailagoa da, cookieak ez direlako automatikoki bidaltzen cross-site eskaeretara.
- Erabiltzailearen izenean ekintza faltsuak (formularioak bidaltzea, kontua aldatzea, transakzioak egitea...) ezin dira automatikoki gauzatu.
- Webguneak bidaltzen dituen eskaerak fidagarriagoak eta jatorrizko domeinutik datozela ziurtatzen dira.

- Saioaren osotasuna eta segurtasuna nabarmen hobetzen da, eta arriskua asko murrizten da erabiltzaileak kanpoko webguneak bisitatzen dituenean.

2.6 Information Disclosure - Sensitive Information in HTTP Response Headers (X-Powered-By)

2.6.1 Arazoa

Web zerbitzariak X-Powered-By HTTP goiburu bat bidaltzen duenean, erabiltzaile edo erasotzaile batek webguneak erabiltzen dituen teknologia eta framework-ak identifika ditzake, adibidez PHP edo beste framework edo hizkuntza batzuk.
Honek erasoen arriskua handitzen du, izan ere, erasotzaile batek jakin dezake zein den teknologia erabiltzen den eta horren arabera ahultasun zehatazak bilatu ditzake. Horrek, lapurketak, injezioak edo bestelako erasoak egitea errazagoa bihurtzen du.
Informazio horren bidez, erasotzaileak:

- Softwarearen bertsio zehatzak identifikatu, eta horien ahultasunak probatu ahal ditu.
- Zerbitzariaren konfigurazioak edo framework-ean dauden hutsuneak aprobetxatu ahal ditu.
- Atake motak planifikatu ahal ditu, arriskua handituz.

2.6.2 Konponbidea

Kasu honetan aurretik zehaztuta dagoen goiburua kendu behar dugu:

```
<?php
header_remove('X-Powered-By');
...
?>
```

2.6.3 Github-eko commit-a

Hurrengo hiru commit-en artean alerta kentzea lortzen da:
`https://github.com/LUC1A05/SEGUR_PROIEKTUA/commit/701aafb134368e1121a3601c6bec5b6cfc9f54e5`
`https://github.com/LUC1A05/SEGUR_PROIEKTUA/commit/2467aca76e432e579d0b3a5766129628014d41c0`
`https://github.com/LUC1A05/SEGUR_PROIEKTUA/commit/b202486c1352f4c7adec3e71c865a1d5a7a99d6b`

2.6.4 Emaitza

X-powered by goiburua kendu eta gero, informazioa zabaltzea zailagoa izango da. Goiburuarekin erasotzaileak webgunearen teknologiari buruzko informazioa lor zezaketen, hau da, ahuleziak errazago aurkitu zezaketen.
Konponketa honekin honelako erasoak sahiestea lortzen da.

2.7 Information Disclosure - Sensitive Information in HTTP Response Headers (Server)

2.7.1 Arazoa

Web zerbitzariak Server HTTP goiburu bat bidaltzen duenean, erabiltzaile edo erasotzaile batek jakin dezake zein zerbitzari software eta bertsio erabiltzen den, adibidez: Server: Apache/2.4.54. Hori arazo bat da, segurtasun-arriskuak handitzen dituelako. Erasotzaile batek zerbitzariaren bertsioa ezagututa, horren ahultasun zehatzak probatu ditzake, eta horrela lapurketa, injezioak edo bestelako erasoak errazago egin ditzake.
Goiburu horren bidez, erasotzaileak softwarearen bertsio zehatza identifika dezake, eta horren arabera konfigurazio hutsuneak edo ahultasunak aprobetxatu. Gainera, lortutako informazioa erabilita, erasotzaileak erasoak modu planifikatuan burutu ditzake, webgunearen arriskua handituz.

2.7.2 Konponbidea

apache_security.conf eta .htaccess fitxategiak gehitu eta konfiguratu dira. Gainera docker-compose.yml, defoult.conf eta Dockerfile fitxategiak aldatu dira:

.htaccess:

```
<IfModule mod_headers.c>
    Header unset Server
</IfModule>
```

config/apache_security.conf:

```
ServerSignature Off
ServerTokens Prod

<IfModule mod_headers.c>
    Header unset Server

    Header always set X-Content-Type-Options "nosniff"
    Header always set X-Frame-Options "DENY"
    Header always set X-XSS-Protection "1; mode=block"
    Header always set Referrer-Policy "no-referrer-when-downgrade"
    Header always set Content-Security-Policy "default-src 'self';
    script-src 'self'; style-src 'self'; img-src 'self' data:;
    font-src 'self'; object-src 'none'; frame-ancestors 'none';
    base-uri 'self'; form-action 'self';"
</IfModule>
```

Dockerfile:

```
...
COPY config/apache-security.conf /etc/apache2/conf-enabled/security.conf
```

```
RUN a2enmod headers
...
```

docker-compose.yml:

```
...
web:
    volumes:
      ...
      - ./config:/config
...
```

nginx/default.conf (aipatzekoa da alerta hau hurrengoaren ostean zuzendu zela, beraz, fitxategia bertan sortzen da):

```
...
server_tokens off;
...
```

2.7.3 Github-eko commit-a

https://github.com/LUC1A05/SEGUR_PROIEKTUA/commit/2f4d4db74dcdbab298a73059459c46765d62e962

2.7.4 Emaitza

Erasotzaileak web orriaren teknologiari buruzko informaziorik ez badu, hainbat ahulezi aurkitzea zailagoa izango da. Izan ere, teknologiaren bertsioa izanda hainbat ahulezi publiko aurkitzea oso erreza da. Konponketa ondoren web orria ahulezi publiko horien aurkako erasoak egitea zaildu egingo du.

2.8 Missing X-Content-Type-Options Header

2.8.1 Arazoa

Web zerbitzariak ez du bidaltzen X-content-type-options goiburua eta hau arriskutsua izan daiteke MIME Sniffing erasoak direla eta. Izan ere, goiburu honek eraso horietatik babesten ditu erabiltzaileak.

2.8.2 Konponbidea

default.conf fitxategia sortu eta editatu eta docker-compose.yml eta Dockerfile fitxategiak eraldatu:

defoult.conf:

```
server {
    listen 81;

    server_name localhost;
```

```
    location / {
        proxy_pass http://web:80;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

        proxy_hide_header Server;
        add_header Server "WebServer";
        add_header X-Content-Type-Options "nosniff";
    }
}
```

docker-compose.yml:

```
...
  nginx:
    image: nginx:latest
    ports:
      - "81:81"
    volumes:
      - ./nginx/default.conf:/etc/nginx/conf.d/default.conf:ro
    depends_on:
      - web
...
```

Dockerfile:

```
...
RUN printf '%s\n' \
'ServerSignature Off' \
'ServerTokens Prod' \
'ServerName localhost' \
'' \
'<IfModule mod_headers.c>' \
'    Header always unset X-Powered-By' \
'    Header always set X-Content-Type-Options "nosniff"' \
'</IfModule>' \
> /etc/apache2/conf-available/hide_server.conf \
&& a2enconf hide_server
...
```

2.8.3 Github-eko commit-a

https://github.com/LUC1A05/SEGUR_PROIEKTUA/commit/f9333c0a5d951e5f079e7eac0cecb154ab32b7c9

2.8.4 Emaitza

Aldaketa honekin MIME Sniffing funtzioa desaktibatzen da:

- Zerbitzariak fitxategiak fitxategi motaren arabera tratatuko ditu. Hau da, CSS fitxategi bat dela esaten badu eta CSS ez den kodea badu, ez da exekutatuko.

Laburbilduz, web orriaren erabiltzaileengan egin daitezkeen MIME sniffing erasoak kentzea lortzen da.