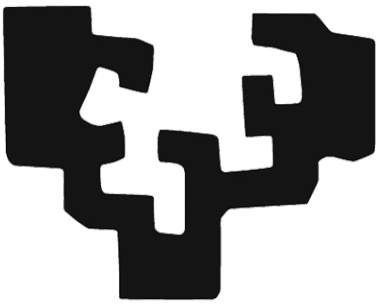
Lana-2

WEB SISTEMAREN ANALISIA



Universidad del País Vasco

Euskal Herriko Unibertsitatea

Titulazioa:

Kudeaketaren eta Informazio Sistemen Informatikaren Ingeniaritza

Egileak:

Asier Monge, Gorka Piedra, Jon Ander Duque, Aitzol Luengo , Oiher Rodriguez

Data:

2024-11-24

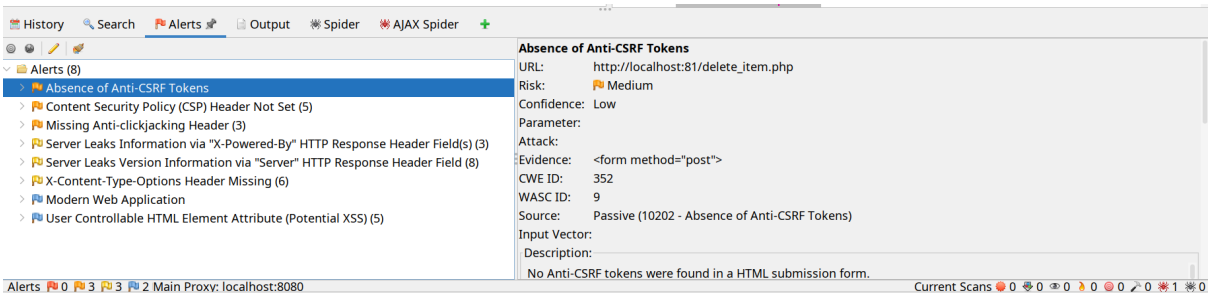
AURKIBIDEA

1. Auditoriak egiteko jarraitutako prozesua

Gure proiektuaren segurtasun auditorioa egiteko, ZAP programa erabili dugu. Programa honek web sistemaren ahuleziak zeintzuk diren esango dizkigu. Baina ez ditu ahulezia guztiak aurkituko, beraz, OWASP 2021 zerrendan oinarrituko gara.

Auditoriarekin hasteko beharrezkoa da jakitea zeintzuk diren web sistemaren ahuleziak. Horretarako ZAP tresna erabiliko dugu.

Hona hemen lortutako emaitza ZAP erabiltzerakoan:



Argazkian ikus daitekeen moduan 8 ahulezia desberdin ikusi ahal dira garrantziaren arabera ordenatuta. Ahulezi bakoitzean klik egiten badugu, haren deskribapena ikus dezakegu. Horrek arazoa nondik datorren jakiteko balio ahalko zaigu.

2. Auditoriaren emaitzak eta hausnarketak

8 ahuleziak OWASP 2021-eko zerrendan oinarrituta konponduko ditugu.

OHARRA:
Administratzaile moduan jarduteko hurrengo gmaila eta kontraseña sartu behar da web orrira sartzekoan:

Korreoa: issksadmin@admin.com
Kontraseña: admin#ja777s

2.1 Sarbide-kontrola haustea

Garrantzitsua da erabiltzaile bakoitzak haien baimenak ahalbidetzen dituzten ekintzak soilik egin ahal izatea. Gure kasuan, erabiltzaile bakoitzak beraien datuak aldatzea da autentifikatuta dauden erabiltzaileen funtzionalitate bakarra. Beraz gure web orrian sartzeko lehenengo identifikatu egin beharko dira. Horretarako erabiltzaileak sisteman saioa hasi duela konprobatu beharko dugu. Eta saioa hasi ez badu saio hasteko horrira birbidaliko diogu.

Horretaz gain, web sistemako formularioekin arazoak konpondu behar izan ditugu. Arazo hau konpontzeko CSRF-a ekiditen dituen“token”-ak erabili ditugu. Hasieran formularioan gordetzen den eremu ezkutu bati ez dakigun balio bat esleituko zaio. Eta horretaz gain “token”-a sortzen dugu.

```php
if (!isset($_SESSION['email']) || !$_SESSION['logged_in']) {
    header("Location: login.php");
    exit();
}

if (empty($_SESSION['csrf_token'])) {
    $_SESSION['csrf_token'] = bin2hex(random_bytes(32));
}

include 'databaseConnect.php';

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    if (!isset($_POST['csrf_token']) || $_POST['csrf_token'] !== $_SESSION['csrf_token']) {
        echo "Tokena ez da zuzena";
        exit();
    }
}
```

Bestalde formularioa bidaltzerakoan sortutako “token”-a eta formularioarena bat datozen konprobatuko ditugu. Gainera, formularioa bidaltzen denean “token”-aren balia aldatu egingo da.

```php
<input type="hidden" name="csrf_token" value="<?php echo $_SESSION['csrf_token']; ?>"><br>
```

2.2 Akats kriptografikoak

Datuak babesteko teknikak erabili behar izan ditugu. Modu honetan erasotzaileren bat gure datu-basera heltzerakoan ezin izango ditu pasahitzak lortu. Gainera pasahitzei gatza sartuko diegu, horrela, pasahitzek luzeera desberdina izango dute eta eraso bat jasotzea zailagoa izango da.

Hurrengo irudian ikus dezakegunez, *PASSWORD_DEFAULT* erabiliz, bcrypt laburtze metodoa aplikatu dugu. Horrek errendimenduaren eta segurtasunaren arteko balantzea bermatzen du, eta denbora igaro ahala, hobekuntza berriekin eguneratuko da.

```php
$hashed_pasahitza = password_hash($pasahitza, PASSWORD_BCRYPT);
```

2.3 Injekzioa

Talde honetako ahuleziak aplikaziok erabitzaileak sartutako dauak balioztatu, garbitu edo iragazten ez direnean gertatzen dira.

Gure kasuan web sistemaren analisia egiterakoan ez dugu horrelako arazorik eduki. Izan ere, SQL injekzioa lehenengo entregan implementatu genuen. Gure web sisteman datu-basearekin konexio zuzena duten input-en bitartez.
Denetan prozesu berdina erabili dugu: kontsulta bakoitzean jasotzen diren datuak zuzenean sartu beharrean “?” karaktereak sartzen dira, Horrela, “?” bakoitza gero sartu nahi den datu bakoitzarekin ordezkatzen da, eta gero kontsulta exekutatzen da.

```php
$stmt = $conn->prepare("INSERT INTO bideojokoa (titulu, egilea, prezioa, mota, urtea)
        VALUES (?, ?, ?, ?, ?)");
if ($stmt == false) {
    echo "Errorea datu basearekin: " . $conn->error;
}
$stmt->bind_param("ssdss", $titulu, $egilea, $prezioa, $mota, $urtea);
if ($stmt->execute()) {
    header("Location: home.php");
    exit();
} else {
    echo "Errorea datuak gordetzean";
}
$stmt->close();
```

Beste arazo bat Cross Site Scripting motako izan da. Arazo hauetan script-ak sartzen dira web orrian. Eraso mota hauek baimentzen duten puntuak inputak dira. Beraz, arazoa nondik agertzen den jakinez gero konpondu dezakegu. Input-etan sartzen diren balioak kontrolatzen ez badira, sartutako informazioa javascript moduan edo html kode moduan sar daiteke. Beraz htmlspecialchars funtzioa erabili dugu arazo hau konpontzeko.

```php
$game_id = htmlspecialchars($titulu) . '-' . htmlspecialchars($egilea); // Crear un id único combinando título y autor
echo "<div class='bideojoko' onclick='toggleDetalles(\"$game_id\")'>";
echo "<h2 class='bideojoko-titulua'>" . htmlspecialchars($titulu) . "</h2>";
echo "<h2 class='bideojoko-titulua'>" . htmlspecialchars($egilea) . "</h2>";
echo "</div>";
echo "<table id='detalles-$game_id' style='display:none;'>";
echo "<tr><th>Atributua</th><th>Balioa</th></tr>";
echo "<tr><td>Titulua</td><td>" . htmlspecialchars($titulu) . "</td></tr>";
echo "<tr><td>Egilea</td><td>" . htmlspecialchars($egilea) . "</td></tr>";
echo "<tr><td>Prezioa</td><td>" . htmlspecialchars($prezioa) . "</td></tr>";
echo "<tr><td>Mota</td><td>" . htmlspecialchars($mota) . "</td></tr>";
echo '<tr><td><button class="item_modify_submit" onclick="window.location.href=\'modify_item.php\'">Editatu</button></td></tr>';
echo '<tr><td><form action="delete_item.php" method="post" onsubmit="return confirm(\'Ziur zaude ezabatu nahi duzula?\');">';
echo '<input type="hidden" name="titulu" value="' . htmlspecialchars($titulu) . '">';
echo '<input type="hidden" name="egilea" value="' . htmlspecialchars($egilea) . '">';
echo '<input type="hidden" name="csrf_token" value="' . $_SESSION['csrf_token'] . '">';
echo '<button type="submit" class="item_delelte_submit">Ezabatu</button>';
echo '</form></td></tr>';
echo "</table>";
```

2.4 Diseinu ez-segurua

Atal honek diseinu akats guztiak biltzen ditu. Gure web sistemaren kasuan lehenengo entregan proposatu genuen diseinuan arazo nagusi bat aurkitu dugu. Izan ere, edonork egin ditzake aldaketak bideojokuaren katalogoan. Hau da, edozein

erabiltzailek bideojokuak gehitu, editatu eta ezabatu ditzake bideojokuaren katalogotik.

Arazo hori konpontzeko modu errezena administratzaile rol bat sartzea da. Horrela administratzaileek bakarrik egin ahalko dituzte aldaketak katalogoan eta diseinu seguruago geratuko da administratzaileak bideojokuak kudeatzeko. Horretarako, erabiltzaile bat administratzaile moduan ibili ahal izango da baimena badu eta ez badauka mezu bat agertuko zaio ekintza hori egiteko baimendik ez duela esanez.

```php
if ($_SESSION['role'] !== 1) {
    echo "Ez daukazu baimenik hona sartzeko.";
    exit();
}
```

Bestalde gure web orria seguruago egiteko erabiltzailearen aktibitatea jarraituko dugu. Hau da, erabiltzaileak minutu oso batean ez badu ez saguaren kurtsorea mugitzen edo ez badu saguko botoiren bat ikutzen erabiltzaile horren saioa itxi egingo dugu.

Azkenik, erabiltzaile bakoitzak web orrira sartzeko login atalean bere kontraseña sartzeko limite bat izango du. Horrela erasotzaileak beste erabiltzaile batzuen kontuekin sartzea ekidingo dugu.

2.5 Segurtasun-konfigurazio ez nahikoa

Web orrian agertzen diren erroreen informazioa ez agertzeko, Dockerfile fitxategia aldatu dugu. 404 errorea agertzen denean, web sistemari buruzko informazioa agertu daiteke erasotzaileentzat baliagarria izan ahal dena. Hori konpontzeko hurrengo aldaketak egin ditugu:

```
FROM php:7.2.2-apache
RUN echo "ErrorDocument 404 /404.html" >> /etc/apache2/apache2.conf
RUN echo "ServerSignature Off" >> /etc/apache2/apache2.conf
RUN echo "ServerTokens Prod" >> /etc/apache2/apache2.conf
RUN echo "expose_php = Off" >> /usr/local/etc/php/php.ini
```

Horretaz gain, header desberdinak erabili ditugu gure web sistema seguruagoa izateko. Horretarako apache zerbitzarian aldaketak egin ditugu Dockerfile fitxategiaren bidez:

```
RUN a2enmod headers
RUN echo "Header set X-FRAME-OPTIONS \"SAMEORIGIN\"" >> /etc/apache2/conf-enabled/security.conf
RUN echo "Header unset X-Powered-By" >> /etc/apache2/conf-enabled/security.conf
RUN echo "Header set X-XSS-Protection \"1; mode=block\"" >> /etc/apache2/conf-enabled/security.conf
RUN echo "Header set X-Content-Type-Options nosniff" >> /etc/apache2/conf-enabled/security.conf
```

Goiko argazkian header desberdinak daude eta bakoitzak bere funtzioa dauka.

- X-Frame-Options: “Clickjacking” erasoak saihesteko balio du.
- X-Powered-By: Honek erabiltzen dugu teknologiari buruzko informazioa filtratu dezake beraz Header hori desgaitu egin dugu.
- X-XSS-Protection: Header honek XSS erasoa dagoela antzematen badu orria blokeatu egingo du.
- X-Content-Type-Options: MIME sniffing motako erasoak saihesteko balio du.

2.6 Osagai kalteberak eta zaharkituak

Segurtasun-akats ezagunak dituzten edo segurtasun eguneratzerik jasotzen ez duten osagaiak erabiltzen baditugu, arazoak sor daitezke, erasotzaileek sistema arriskuan jartzeko balia ditzaketelako. Oso zaila da osagai hauen arrisku-maila egiaztatzea, asko direlako eta bakoitza iturri desberdin batetik datozelako. Dena den, hurrengo pausuak jarraitu ditugu puntu honek gure sisteman duen eragina aztertzeko eta murrizteko.

Sisteman erabiltzen diren osagai guztien bertsioak ezagutu behar dira, ahal den neurrian,eta ez sinpleki azken bertsioa “latest” erabili. Honen arrazoia, eguneratze berriek bateraezintasunak edo oraindik konpondu ez diren hainbat arazo eragin ditzaketela da. Era berean, erabiltzen ditugun osagaiek eguneratze euskarria izatea eta berriztuta egotea oso estrategia ona da. Beraz, oreka aurkitu beharko dugu azken eguneratze bertsioa eta akatsik eragiten ez duen bertsio bat erabiltzearen artean.

Etorkizunari begira, badaude jarraitu beharko genituzken zenbait aholku. Hauen artean hurrengoak daude: sistema normalean eskaneatu ea ahuleziak aurkitzen ditugun, osagai berriak deskargatzerako orduan, sinatuta eta iturri ofizialetatik datozenak soilik erabiltzea.

2.7 Idetifikazioa eta autentifikazio-akatsak

Identifikazio eta autentifikazio akatsen arloan, pasahitz bakoitza 10000 pasahitza ohikoen lista batean dagoen konprobatuko dugu, hau da, ohikoak diren pasahitz errazak gure sistematik kanpoan uzteko. Common_passwords.txt barruan 10.000 pasahitz komunak daude jarrita.

```php
$common_passwords = file('common_passwords.txt', FILE_IGNORE_NEW_LINES);
    // Pasahitza ohiko pasahitzen listarekin konparatu
if (in_array($pasahitza, $common_passwords)) {
    // Pasahitza listan badago prozesua amaitu
    echo "<script nonce='$nonce'>";
    echo "alert('Zure pasahitza oso ohikoa da');";
    echo "</script>";
    echo "<script nonce='$nonce'>window.location.href = 'modify_user.php';</script>; ";
    exit();
```

Php-n saioak kudeatzeko cookie-ak erabiltzen dira, beraz, hauen konfigurazio desberdinak egin ditugu segurtasuna hobetzeko:

```php
ini_set('session.cookie_httponly', 1);
ini_set('session.cookie_secure', 1);
ini_set('session.cookie_samesite', 'Strict');
```

2.8 Datuen eta softwarearen osotasunaren akatsak

Atal honetan ez dugu aldaketa nabarmenik egin behar. Izan ere, gure web sistema ez da era automatiko batean berriztuko.

2.9 Akatsak segurtasunaren monitorizazioan

Gure web sisteman gertatzen diren ekintzak monitorizatu ditugu, segurtasun helburuekin.Gure web siteman gertatzen diren errore guztiak erro_log fitxategia gordeko ditugu horrela web orriak dituen erroreak errazago antzeman izan ahako ditugu. Horretarako, bolumen bat muntatu behar izan dugu kontainerrearen eta gure makinaren arteko fitxategiak sinkronizatzeko. Horrela, log-ak gure makinan ere gordeko dira.

```
volumes:
  - ./app:/var/www/html/
```

3. Aldaketen ondorengo auditoria

Agertutako aldaketak egin ondoren, auditoria berri bat egin dugu emaitza berriak aztertzeko. Horretarako ZAP erabili dugu.

4. Bi auditorien arteko desberdintasunei buruzko ondorioak

Bi auditoreen arteko desberdintasun nagusia detektatutako ahultasunetan eta horiei aurre egiteko egindako aldaketetan ikus daiteke. Hala ere, lehen azaldu dugun moduan, aldaketa askok ez dute kanpotik eragin nabarmenik. Aldaketa horiek oso garrantzitsuak diren arren, web orriaren administratzaileak bakarrik izan dezake hauek egiaztatzeko aukera.