Universidad del País Vasco
Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

Informazio Sistemen Segurtasuna Kudeatzeko Sistemak
Hirugarren maila
1. Lauhilabetea

ALTZARI DENDA
PEN TESTING

Eneko Perez, Eneko Basauri eta Aimar Larrazabal.

2022ko Azaroaren 19a

1 Sarrera

Lehen entregan sortutako altzari dendaren web orrialderen pen testing-a egin dugu, honen ahuleziak aurkituz eta zuzenduz. Dokumentu honetan zehar ahulezi hauek eta konpontzeko egindakoa aztertu egingo ditugu.

2 Pen-testing

2.1 AUDITORIAK EGITEKO INSTRUKZIOAK:

- ZAP aplikazioa instalatuko dugu gure ordenagailuan.
- Terminalaren bidez zap.sh egingo dugu aplikazioa irekitzeko.
- Proiektua docker bidez hedatu aurreko pausuetan esaten duen bezala.
- Interneten http://localhost:81 helbidean sartuko gara, konprobatzeko ondo egin dugula.
- Azkenik ZAP-ean eskaneo automatikoa egingo dugu, http://localhost:81 helbidea jarriko dugu eta eraso botoiari emango diogu.
- "Alertas" jartzen duen lekuan klikatuko dugu, gure web sistema daukan erroreak ikusteko.
- "Alerta" bakoitzaren barruan klikatzen badugu informazio guztia aterako zaigu; errore mota, soluzio posibleak eta abar.

2.2 Auditorien emaitza

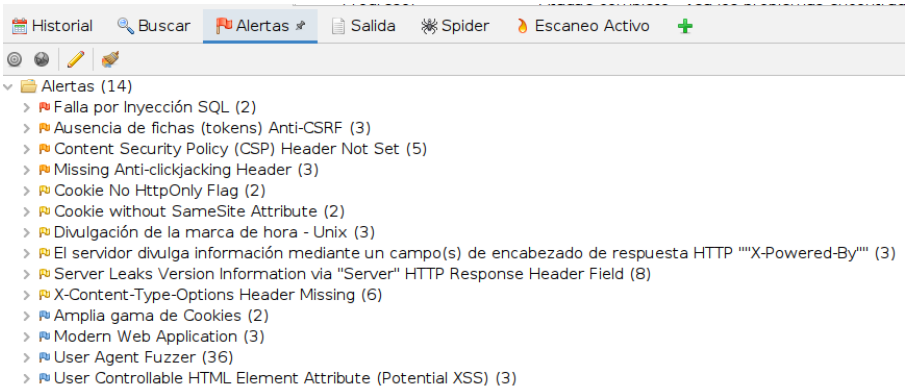

Fig. 1: Egindako hasierako ZAP pen-testing

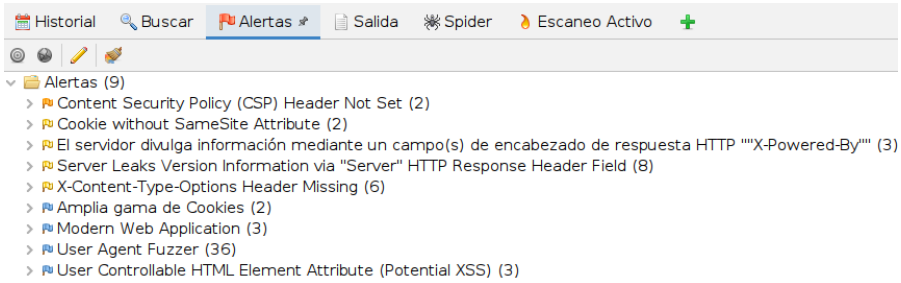

Fig. 2: Egindako amaierako ZAP pen-testing

3 Erabiltzaile proba

Erabiltzaile bat sortu dugu "proba" erabiltzaile izena duena eta "proba" pasahitza duena. Irudian ikusi ahal dugunez phpMyAdmin-en pasahitza hasheatuta dago.

NAN	Pasahitza	IzenAbizena	TelefonoZenbakia	JaiotzeData	Email	ErabId
12345678A	$2y$10$6tB8JHj3o0Oj6YDt3yXb1ufVptIk0w7X.9lrFQoKKir...	proba	123456789	2000/01/01	proba@pro.ba	proba

Fig. 3: Erabiltzaile proba

4 Konpondutako ahuleziak

4.1 Sarbide-kontrola haustea

- Path transversal errorea:[1] Lanaren lehenengo entregan segidan sartu ahal zinen erabileremu.php-an nabigatzailean http://localhost:81/erabileremu.php sartzen bazenuen. Arazo hau konpondu dugu sesioan aldagai bat gordez, aldagai honek indexean hasieratuko da eta login egokia egiten badu erabiltzaileak aldagai honetan guk nahi duguna jarriko dugu. Php guztietan sartzerakoan konprobatu egingo da aldagaiaren balioa egokia den. Zuzena bada orrialdea ikusi ahalko dugu, bestela index-era bidaliko gaitu.

```
//login2.php ez badator index.php-ra bidali.
if($_SESSION['sesioIzena']!="login2.php"){
    header("Location: http://localhost:81/index.php");
}
```

Fig. 4: Seioa hasi duela konprobatzeko.

- Unix.time() enkriptazioa: Unix.time() funtzioa erabiltzean, Unix denbora lortzeko, informazioa enkriptatu barik dago. Informazioa hau date() funtzioarekin kodifikatu dugu errorea ezabatzeko. [2]

```
<link rel="stylesheet" href="styles.css"?v=<?php $tiempo_UNIX=time(); echo date("d/m/Y", $tiempo_UNIX); ?>>
```

Fig. 5: Unix informazioa enkriptatzeko

4.2 Akats kriptografikoak

- Pasahitzen enkriptazioa: Alde batetik pasahitzak textu planu bezala ez gordetzea egin dugu. Pasahitzei gatza gehitu egingo zaie eta hasheatu egingo dira datu basean gordetzeko. Horrela, erabiltzaile batek saioa hasterakoan bere pasahitzari erabiltzaile horren gatza gehitu eta hasheatuko da datu basean jada hasheatuta dagoen pasahitzarekin konparatzeko. Hau password_`hash() eta password_verify()` php funtzioekin egin dugu.[4]

```
//pasahitza hasheatu
$pasahitza_hash = password_hash($pasahitza, PASSWORD_DEFAULT);
```

Fig. 6: Pasahitzen enkriptazioa.

```
if(($nr == 1)&&password_verify($pasahitza, $row['Pasahitza'])){
```

Fig. 7: Pasahitza konprobatu.

4.3 Injekzioa

- MySQL komando zaharkituak saihestu: Lehen bertsioan SQL injekzioak egin ahal ziren. MySQLi-ren Prepared Statements erabili ditugu, datuak sartzeko, kontsultatzeko, eguneratzeko edo ezabatzeko datu basetik. Hauei esker web orrialdeko inputetan nahiz eta SQL instrukzioak idatzi ez dira exekutatuko. Prepared Statements erabili ezean, edozeinek formularioetan True aldagai boolearra bueltatuko duen query-a egin dezake, horrela login sistema guztiz haustuz. Adibidez "ashj' OR 1=1 #" erabiltzen bada MySQL-ek konprobatuko du ea statement hau True den, eta OR adierazle logikoaren osteko 1=1 egia denez login-a arrakastatsu bueltatuko da eta gure autentifikazio sistema haustu daiteke. Prepared Statement-ekin ordea, query-ak jasoko duena aurretik zehazten da, zenbat aldagai eta zer motakoak izango diren, horrela erasotzaileen injekzio erasoak saihestuz.[5]

4.4 Diseinu ez segurua

- Erabiltzaileen interfazean arazoak: Erabiltzailearen pribilegioak beharbeharrezkoetara murriztu ditugu eta irekitzeko baimena ez dituen lehioetara sarbidea ez duela ziurtatu dugu, adibidez, administratzaile pribilegioak dituen saioa atxikitu beste erabiltzaileen datuak aldatu edo ezabatzeko.

```php
//prepare and bind
$stmt = $con->prepare("INSERT INTO Erabiltzaileak (NAN, Pasahitza, IzenAbizena, TelefonoZenbakia, JaiotzeData, Email, ErabId) VALUES (?, ?, ?, ?, ?, ?, ?)");
$stmt->bind_param("sssissss", $nan, $pasahitza_hash, $izena, $telefonoa, $jaiodata, $email, $erabIzena);

//set parameters and execute
$nan = $_POST['nan'];
//pasahitza hasheatu
$pasahitza_hash = password_hash($pasahitza, PASSWORD_DEFAULT);
$izena = $_POST['izena'];
$telefonoa = $_POST['telefonoa'];
$jaiodata = $_POST['jaiodata'];
$email = $_POST['email'];
$erabIzena = $_POST['erabIzena'];
$bool = $stmt->execute();

if($bool)
{ // arrakasta badu sententzia, hemen sartuko da
    // hemen sartu behar ditugu datuak log taulan (arrakastatsua bai)
    header("Location: http://localhost:81/login2.php");
    exit;
}else{
    echo '<script language="javascript">alert("ERROREA: Sartutako erabiltzailea existitzen da edo datuak txarto sartu dituzu, saiatu beste batekin!!");</script>
}
$stmt->close();
$con->close();
```

Fig. 8: MySQL injekzioa.

4.5 Segurtasun konfigurazioa ez da nahikoa

- CSP segurtasun-geruza gaineratu: Content Security Policy Header Not Set. Content Security Policy (CSP) segurtasun-geruza erantsi bat da, zenbait eraso detektatzen eta arintzen laguntzen duena, hala nola Cross Site Scripting (XSS) eta datu-injekzioko erasoak.[6]

```html
<meta http-equiv="Content-Security-Policy" content="script-src 'self' script.js 'unsafe-inline';">
```

Fig. 9: Content Security Policy.

- Segurtasun-konfigurazioa ez da nahikoa: Cookie No HttpOnly Flag. session_start() egin baino lehenago ini_set('session.cookie_httponly', 1); egingo dugu, flag hori batera jartzeko. [7]

```php
//httponly true jartzeko.
ini_set( 'session.cookie_httponly', 1 );
```

Fig. 10: Cookie No HttpOnly Flag.

- Sarbide-kontrol horizontalean arazoa: DatuakEditatu.php-an erabiltzaile batek beste erabiltzaile baten datuak aldatu ahal zituen, sesioan erabiltzailea gorde dugu. Horrela soilik uneko erabiltzailearen datuak aldatu daiteke.

- Ausencia de fichas (tokens) Anti-CSRF: Ekintza bat benetan erabiltzaileak hirugarren batek egin beharrean egiten duela ziurtatzeko, ondoren egiazta daitekeen identifikatzaile bakarren batekin lotu behar da, token izenekoa. Sesioan token aleatorio bat gorde dugu. Inprimakiak babesteko, identifikatzailea ezkutatutako eremu batean sartu ohi da, eta formularioko gainerako datuekin batera bidaltzen da. [8]

```php
$_SESSION['token'] = md5(uniqid(mt_rand(), true));
```

Fig. 11: token csrf sortu.

```php
if(isset($_POST['erregistratu'])&& $_POST['csrf'] == $_SESSION['token']) /
{
```

Fig. 12: token csrf lortu.

- missing anti-clickjacking header: X-Frame-Options-ek bere webgunearen edukia beste leku batzuetan sartzea eragozten du. Nabigatzaileak aukera ematen du beste gune batzuek iframe web-orria irekitzeko. Clickjacking erasoaren Apache web zerbitzaria ere ziurtatzen du. header("X-Frame-Options: DENY"); eta header("X-Frame-Options: SAMEORIGIN"); funtzioak erabiliz konpondu dugu.

4.6 Osagai kalteberak eta zaharkituak

- Horrelako kalteberatasunak sortzen dira web-aplikazio edo -azpiegitura zaharkitu baten barruan softwarea edo osagaiak erabiltzeagatik edo ahultasun ezagunak izateagatik. Hau ekiditzeko, erabiltzen ez ditugun artxiboak eta funtzioak kendu ditugu eta erabiltzen ditugun tresnen bertsioa eguneratu ditugu.

4.7 Identifikazio- eta autentifikazio-akatsak

- Loginean Brute Force erasoak saihestea: Loginak kontrolatzeko datu basean log izeneko taula bat sortu dugu, taula honetan sartzen saiatu diren erabiltzaile izena, sartutako pasahitza eta saiatu diren kopurua gordetzen dira. Hau erabili dezakegu administratzaileari mezu bat bidaltzeko, erabiltzaile batek sartzen saiatzen bada askotan, momentuz ez dago implementatuta eta phpMyAdmin-etik ikusi behar dugu.

4.8 Datuen eta softwarearen osotasunaren akatsak

- Akats hauek lotuta daude kodea eta azpiegitura ez babestearekin osotasunaren urraketen aurrean. Gure proiektuan ez dugu erabili plugin, bibliotekarik ezta repositoriorik. Erabiliko bagenu bere jatorria segurua dela konprobatu beharko genuke edo firmak erabili beharko genuke software edo datuen jatorria egiaztatzeko.

```
<tr>
    <td> </td>
    <td><input id="csrf" type="hidden" name="csrf" value="<?php echo $_SESSION['token'];?>"></td>
    <td> </td>
</tr>
```

Fig. 13: token csrf konprobatu.

```
//Use below in your php file which outputs response to client side.
header("X-Frame-Options: DENY");
header("X-Frame-Options: SAMEORIGIN");
```

Fig. 14: missing anti-clickjacking header.

4.9 Akatsak segurtasunaren monitorizazioan

- Saioen segimendua indartzea: Kategoria honen helburua profesionalei urratze aktiboak detektatzen, eskalatzen eta erantzuten laguntzea da. Gure kasuan, sesio saioa ez da gordetzen. Lehenengo entregan sesio saioa ez zen gordetzen, erabiltzailearen izena sesio aldagaian gorde dugu, honek beste pertsona login egitean aldatuko da. session_start() egin behar da, datu hauek kargatzeko.

Erreferentziak

[1] **Owasp top 10 ahuleziak**
Web orria: https://es.owasp.org/
https://owasp.org/www-project-top-ten/

[2] **Unix denbora**
Web orria: https://es.wikipedia.org/
https://es.wikipedia.org/wiki/Tiempo_Unix

[3] **httpOnlyFlag**
Web orria: https://stackoverflow.com
https://stackoverflow.com/questions/36877/how-do-you-set-up-use-httponly-cookies-in-php

[4] **password_hash() eta password_verify()**
Web orria: https://www.php.net

https://www.php.net/manual/es/function.password-hash.php
https://www.php.net/manual/es/function.password-verify.php

[5] **SQL Injection**
Web orria: https://www.php.net
https://www.php.net/manual/en/security.database.sql-injection.php

[6] **Content Security Policy** https://content-security-policy.com/examples/php/

[7] **httponly**
Web orria: https://www.stackoverflow.com
https://stackoverflow.com/questions/36877/how-do-you-set-up-use-httponly-cookies-in-php

[8] **token CSRF** https://diego.com.es/ataques-csrf-cross-site-request-forgery-en-php