Informazio Sistemen Segurtasuna Kudeatzeko Sistemak

Lana 2 - Pentesting

Uxue Aurtenetxe
Aimar Basterretxea
Ane Moreno
Yoel Justel
Xinyan Wang

Sarrera

Hurrengo dokumentuan gure webgunearen proiektuan egindako pentesting-probak adieraziko ditugu. Proba hauek webgunearen osotasuna eta segurtasuna berrikusiko dituzte. Horrela, segurtasuna bermatu dezakegu.

Proba hauek irakaslearen gomendioari jarraituz ZAP aplikazioaren bidez egin ditugu. ZAP web eskaner erabiliena da, doakoa eta kode irekikoa, beraz entrega honetako beharretara zuzen egokitu ahal izan dugu. ZAP 'bitartekari-proxy' bat da erabiltzailearen nabigatzaile eta web-aplikazioen artean bidalitako mezuak antzemateko, ikuskatzeko, edukia aldatzeko (beharrezkoa bada) eta pakete horiek helmugara bidali ahal izateko erabiltzen da.

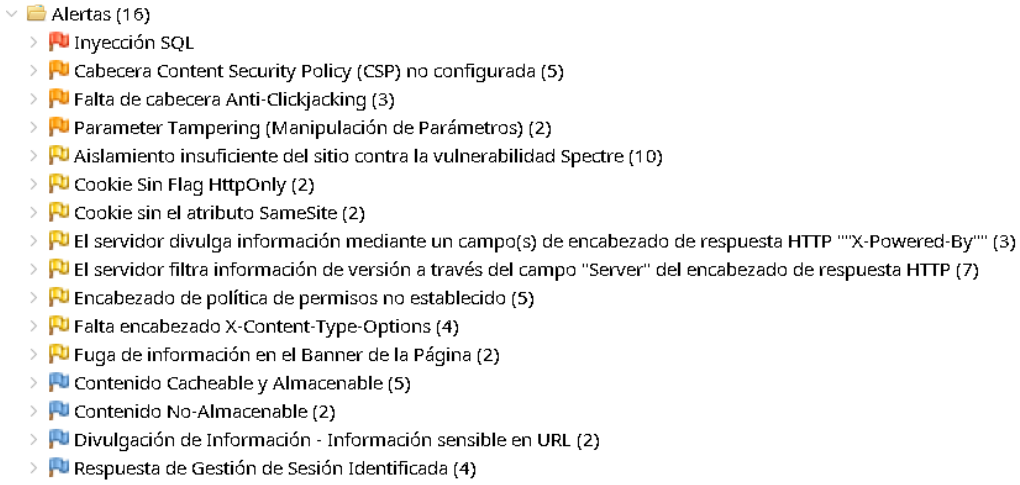
Entrega honen helburu nagusia garatutako web orriak dituen arazo edo arriskuak aurkitzea da, modu honetan web orriaren segurtasuna ziurtatuko da.

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

Honek gure webgunearen segurtasunean arrakala handia suposatzen du. Datu basean SQL kode bat gordetzen baldin bada, erakutsi beharko ez litzatekeen informazioa bistaratzea lortu daiteke. Nahiz eta guk informazio sentsiblerik gorde ez oraindik, edozeinen nortasun agiri zenbakia, pasahitza edo beste edozer lortu daiteke.

Eraso honen adibide famatua “1=1” erasoa da. Honek “or” logikaren funtzioan beti betetzea du helburua. Demagun hurrengoa dugula:

```
$erabiltzailea= $_POST['erabiltzailea'];
$pasahitza= $_POST['pasahitza'];

$query = "SELECT * FROM erabiltzaileak WHERE erabiltzailea=
'$erabiltzailea' AND pasahitza= '$pasahitza'";
$emaitza= mysqli_query($conexion, $query);
```

Eta erabiltzaileak hurrengo izena jarri nahi duela:

```
‘ OR ‘1’=’1
```

Bada izen horrekin pasahitz gabe saioa hasteko aukera emango luke 1=1 beti beteko delako.

Honek hainbat soluzio edo denen baturaren soluzio nagusi bat izan dezake.

- Aldagaiak Parametrizatu eta Prestatu: erabiltzaileak sartutako input-a beti parametroetan gorde, horrela aurrez prestatu eta SQLko komando gisa ez exekutatzea bermatu.
- Erabiltzailearen Sarrera Filtratu: kontsultan espero ditugun sinboloak soilik sartzen utzi, eta erabat debekatu arriskua izan dezaketen “/”,”-”, “*”, edo beste edozein sinbolo berezi.
- Baimenak: erabiltzaileak ez du administratzailearen baimenik izan behar, beharrezkoak soilik.

2. Goiburuak

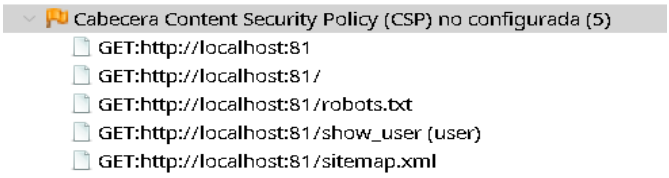
2.1 CSP goiburua

Webguneak ez du CSP (Content Security Policy) erabiltzen. Hau da, ez du esaten zein edukiaren jatorriak diren fidagarriak eta zeintzuk ez. Hori dela medio, nabigatzaileak edozein webgunetik (domeinu, zerbitzari edo fitxategi) JS, CSS edo irudiak kargatu daitezke, eta ondorioz kanpoko baliabide horrek izan dezaken malwarea kargatzen du.

CSP hau jarrita gure baliabideak nondik kargatu adierazi, XSS injekzioak saihestu, eta nabigatzailearen konfiantza handitu dezakegu.

Hori egiteko, esandako 'Content Security Policy' goiburua gehitu beharko litzateke:

Header("Conten-Security-Policy: <policy-directive>; <policy-directive>;");



Irudia 3: CSP goiburua konfiguratu gabe dagoela alerta.

2.2 X-Frame-Options goiburua

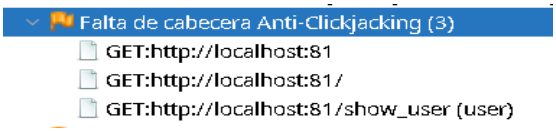
Bestalde, webguneak ez du anti-clickjacking neurri egokirik ezarri. 'Clickjacking' izeneko erasoa pasiboki gertatzen da: beste jatorri batek web-orriari iframe edo antzeko elementu batean txertatzen dionean; horrela, erabiltzailearen klikak manipulatu eta ekintza txarrak eragin daitezke (adibidez, baimenak ematea edo formularioak bidaltzea).

Hau ekiditeko, 'X-Frame-Options' goiburuarekin, hau da, goiburua web-garapenean erabiltzen den mekanismo bat da eta aukera ematen du zehazteko ea web-orri bat beste webgune bateko frame baten barruan agertu edo integratu den. Horregatik, lehen esandako 'Clickjacking' atakeak ekiditeko erabiltzen dira.

Beraz, arazoa ekiditeko, esandako X-Frame-Options goiburua gehitu beharko litzateke.

Header("X-Frame-Options: DENY X");

Header("Frame-Options: SAMEORIGIN");



Irudia 4: Anti-clicjacking goiburua falta dela jakinarazi

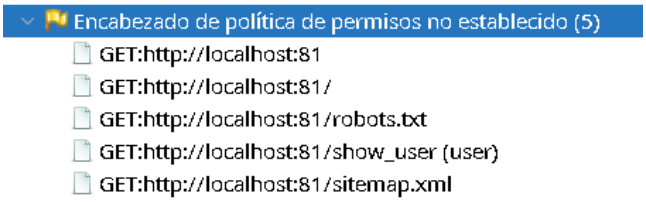
2.3 Baimen-Politikaren Goiburua

HTTPS Permissions-Policy goiburuak nabigatzailearen funtzioetarako sarbidea kontrolatzea ahalbidetzen du, aurreko Feature-Policy-a ordezkatuz. Goiburu horrek funtzioen erabilera mugitzeko aukerak ematen ditu, bai bakoitza bere edukian, bai kanpoko iFrame-etan, baimendu, domeinura mugatu edo desgaitu ahal diren direktiboekin.

Goiburu hau ez badago, nabigatzaileak ez du mugarik izango webgunean zein APIk erabili dezakeen zehazteko(kamera, mikrofonoa, etab.)

Hori ekiditeko, goiburua gehituz, Feature-Policy-a ezarriko da:

Header("Permissions-Policy: <directive>=<allowlist>");



Irudia 5: Feature-Policy ezarri gabeko alerta

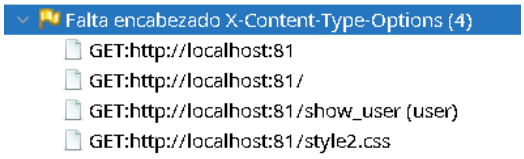
2.4 X-Content-Type-Options goiburua

'X-Content-Type-Options' goiburuak zerbitzariaren instrukzio gisa balio du nabigatzaileek Content-Type goiburuan adierazitako MIME mota, modu zorrotzean errespetatu behar dutela adierazteko, aldatzen saiatu gabe. Horri esker, 'MIME type sniffing' desaktibatzen da, eta horrek bermatzen du edukia garatzaileak definitu duen bezala interpretatuko dela.

Hori lortzeko, hurrengo goiburua definitu beharko litzateke:

Header("X-Content-Type-Options: nosniff");

Goiburu hori gabe, nabigatzailea fitxategi mota asmatzen saia daiteke, zerbitzariak beste bat adierazi arren.



Irudia 6: X-Content-Type-Options goiburua falta

3. Parametroen Manipulazioa

Parameter Tempering web-aplikazioen aurkako eraso bat da non erabiltzaileak edo erasotzaileak nabigatzaileak bidaltzen dituen parametroak aldatzen dituen: URLak, formulario ezkutuko eremuak, cookie-ak... Honekin aplikazioak kodera bidaltzen duen informazio edo jarduera manipulatzeko.

Honetako adibide oso tipikoak prezioak aldatzea, bestelako erabiltzaile baten `user_id` ezkutuan sartzea eta administratzaile bihurtzea dira besteak beste.

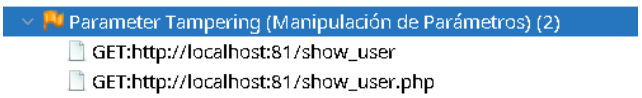
Webguneak hurrengo URL-a ematen badigu:

https://webgunea.com/checkout?product_id=10&price=100

Edonork prezioa 1era bihurtu dezake eta erosketa hori euro 1era gauzatzea ahalbidetuko luke.

Hurrengoak dira proposatutako zuzenketak

- Rol-ak prezioak edo bestelako erabakiak hartzeko beti datu-basetik edo autentifikazio sistematik hartu behar dira. Horrelako funtzioak norabide bakarrean egin beharko dira.
- Erabiltzailearen user_id bere saioaren tokenarekin egiaztatu behar da.
- HttpOnly erabili, cookiak seguru mantendu daitezen eta baimendutako orriak erabili ditzaten.



Irudia 7: Parametroen manipulazioa alerta

4. Spectre

Prozesadore-mailako ahultasuna da, 2018an aurkitu zena, eta modernoenak diren CPU ia guztiei eragiten die (Intel, AMD, ARM, etab.).

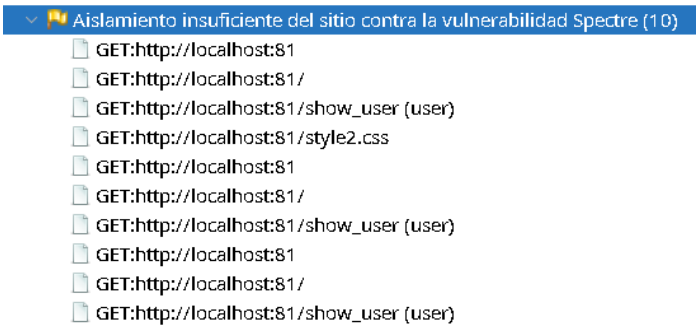
Eraso honek ez du sistema apurtzen “ohiko” web edo kode injekzio bidez, baizik eta hardwarearen funtzionamendua bera ustiatzen du, bereziki branch prediction eta speculative execution mekanismoak, hau da, prozesadoreek iragarpen bidez instrukzioak aurreratzen dituzten mekanismoak.

Prozesadoreak azkartasuna irabazteko erabiltzen duen teknika baliatzen du erasotzaileak

- Espekulazioa: prozesadoreak hurrengo zein adar exekutatuko den aurreikusten du, eta aldez aurretik exekutatzen hasten da emaitza ziurtatu aurretik. Iragarpena okerra bada baztertu egiten du.
- Nahiz eta iragarpena okerra izan, cache memorian aztarna gelditzen da

Horrela erasotzaileak kode jakin bat exekutatzen du web nabigatzailean, eta cachearen portaera aztertuz atzitu den memoria adarra jakin dezake, eta hortik datu pribatuak berreskuratu ditzake

Honen soluzioa nabigatzailea isolatzea da. Webgune bakoitzak prozesu isolatu bat erabiltzen du, eta ez du beste webguneko daturik ikusiko.



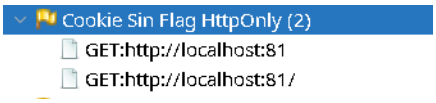
Irudia 8: Spectre

5. Cookie-ak txarto konfiguratuta

5.1 Cookie-a HttpOnly flag-a barik

Cross-Site Scripting (XSS) eraso baten aurrean, erasotzaileak cookie-ak irakur ditzakela esan nahi du. Cookie horietan aurkitzen den edozein token edo informazio baliagarria eskuratu dezake.

Hau konpontzeko, cookie-a zerbitzarian konfiguratuz lortzen da. Hurrengo atributua gehitu beharko litzateke.



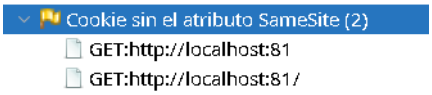
Irudia 9: Flag gabeko cookie-a

5.2 SameSite atributu gabeko cookie-a

Cookie-ak txarto konfiguratuta daudela esaten denean, atributuak txarto ezarrita daudela esan nahi du. Horregatik, alerta hauek izatean web-orrian segurtasun arazoak daudela adierazten dute.

Jasotako alerta, aukera ematen du beste leku batzuetatik egindako eskaeretan bidaltzeko. Cookie-ari falta zaion atributua 'SameSite' babes eraginkorra ematen du lekuen arteko eskaerak faltsutzeko erasoen aurrean (CSRF), gune arteko inkluzio script-en aurrean (XSS) eta tenporizazioan oinarritutako erasoen aurrean.

Horretarako, SameSite atributua behar bezala konfiguratu behar da cookie-ak ezartzean.



Irudia 10: SameSite atributu gabeko cookie-a

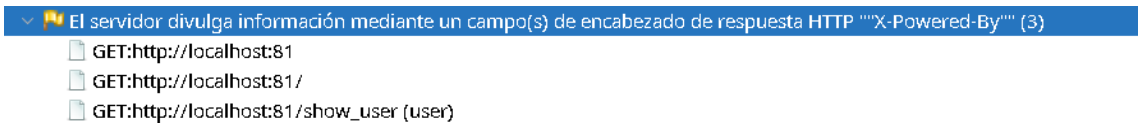
6. X-Powered-By

HTTP erantzunetan agertzen den `X-Powered-By` goiburuak atzealdeko plataforma edo hizkuntza adierazten du eta askotan bere bertsioaren xehetasunak ematen ditu (adibidez: `X-Powered-By: PHP/7.2.2`). Gure ingurunean, kontainer irudiaren lehenetsitako PHP konfigurazioek ohikoena dute goiburu hori automatikoki gehitzea erantzunetara; horren ondorioz, `Dockerfile`-ean ez dago `expose_php = Off` bezalako ezarpenik eta ondorioz web zerbitzariak goiburu hori bidaltzen du. Hau egiaztatzeko, nahikoa da komando hau exekutatzearekin terminalean: `curl -I http://localhost:81`. Erantzunean: `X-Powered-By` agertzen bada, arazoa baieztatuta dago.

Goiburu honek ez du informazio pertsonalik ematen, baina erasotzailearentzat baliagarria den teknologia-fingerprinting-a ahalbidetzen du. Zehazki, bertsio-informazioarekin erasotzaileek automatikoki egokiak diren eskaneoak eta exploit bilaketak abiarazi ditzakete: aurkitutako bertsioari lotutako CVEak azkar bilatu eta eskura daitezkeen esploitazioak aplikatu. Hortik aurrera, `X-Powered-By`-ren presentziak beste ahultasun txikiagoen erabilera errazten du (adibidez, SQL injection), beraz informazio-ihes hori bakarrik ez den bitartean, osatutako informazio-pakete baten zati bihur daiteke eta esploitazio probabilitatea igotzen du. OWASPek hori "Security Misconfiguration/Information Exposure" kategorietan sartzen du eta gomendatzen du erantzun publikoetan teknologia- eta bertsio-informazioa ez erakustea.

Arazoaren printzipioa sinplea da: ez eman beharrezko ez den informazioa kanpoko erantzunetan. Teorikoki, horretarako egon daitezkeen neurri nagusiak hauek dira:

- PHP mailako ezarpenek `expose_php = Off` aktibatzea, horrela PHPk ez du goiburu hori automatikoki gehituko.
- Webservereko mailan (Apache) mod_headers bezalako mekanismoekin goiburu hori erabiltzaileari bidaltzen zaion erantzunetik kentzea (header unset edo filter bat aplikatuz).
- Zerbitzariaren konfigurazio orokorra indartzea komeni da: `ServerTokens` eta `ServerSignature` moduen bidez ematen den informazioa minimizatzea (ez bertsio zenbakiak, soilik izena edo ezta hori ere).
- Sistema-hardeninga eta irudi-kudeaketa: kontainer eta irudiak eguneratuta mantendu eta ez erakutsi garapeneko/diagnostiko fitxategi edo `phpinfo()` moduko baliabiderik ingurune publikoetan.



Irudia 11: X-Powered-by

7. Server goiburua (zerbitzariaren informazio-ihesa)

HTTP erantzunetan “Server” izeneko goiburua automatikoki gehitzen da eta bertan web zerbitzariaren softwarearen izena eta bertsio zehatza adierazten dira (adibidez: `Server: Apache/2.4.25 (Debian)`). Informazio hau Apache bezalako zerbitzarien konfigurazio lehenetsian agertzen da eta gure ingurunean (Docker bidez eraikitako Apache kontainerrak erabilita) baliteke erantzunetan datu hori bistaratzea. Goiburu hori `docker-compose.yml` eta `Dockerfile` fitxategietan ezarritako zerbitzari-konfigurazioaren ondorioz agertzen da, non ez den zehaztu informazio hori murrizteko edo ezkutatzeko neurririk (ServerTokens edo ServerSignature). Hau egiaztatzeko, nahikoa da komando hau exekutatzea terminalean: `curl -I http://localhost:81`. Erantzunean: `Server: Apache/...` agertzen bada, arazoa baieztatuta dago.

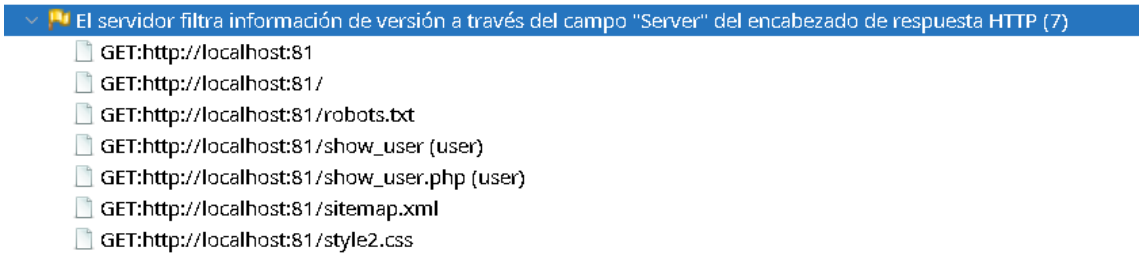
“Server” goiburuak ez du zuzenean informazio pertsonalik erakusten, baina teknologia eta bertsioaren inguruko xehetasun baliotsuak eskaintzen dizkio erasotzaileari. Horren bidez, aplikazioaren azpiegitura identifikatu dezake eta zaurgarritasun espezifikoak bilatu (CVE publikoak, adibidez: Apache 2.4.25 edo 2.4.49 bertsioetan aurkitutakoak).

Eraso-prozesu arruntean urrats bat izaten da fingerprinting deritzona, hau da, erabiltzen diren sistemak, framework-ak eta bertsioak identifikatzea. Goiburu honek beraz, erasoaren lehen fasea errazten du eta OWASP 2021-en “Security Misconfiguration” kategorian sartzen da: konfigurazio lehenetsiak informazio-ihes txikiak sortzen ditu eta horiek erabilita ingurunea hobeto ulertu eta eraso sofistikatuagoak egin daitezke.

Gainera, “Server” goiburua beste goiburu batzuekin (adibidez X-Powered-By) edo HTML banner elementuekin batera agertzen bada, informazio horiek guztiak uztartuz erasotzaileak ingurune teknikoaren deskribapen zehatza osa dezake.

Neurri teoriko nagusia da zerbitzariak bidaltzen duen informazioa murriztea edo ezkutatzea. Horretarako:

- Apache konfigurazioan (apache2.conf edo httpd.conf fitxategietan) `ServerTokens Prod` eta `ServerSignature Off` lerroak gehitu behar dira. Lehenengoak erantzunetan soilik “Apache” hitza erakusten du, bertsio edo sistema eragilearen xehetasunik gabe; bigarrenak berriz, errore-orrietan informazio gehigarria ezkutatzen du.
- Proxy edo balio erdiko geruza bat (adibidez, Nginx) erabil daiteke goiburu hori ezkutatzeko edo erantzuna aldatzeko, kanpora informazio neutro soilik bidaliz.
- Log eta konfigurazio azterketa egitea gomendatzen da: `curl -I` bidez erantzun bakoitzaren goiburuak ikuskatzea eta informazio-ihesak gertatzen diren endpoints identifikatzea.
- Hardening orokorra aplikatzea: Apache bertsioak eguneratu, osagai kalteberak egiaztatu eta behar ez diren moduluak desgaitu.



Irudia 12: Server goiburua (zerbitzariaren informazio-ihesa)

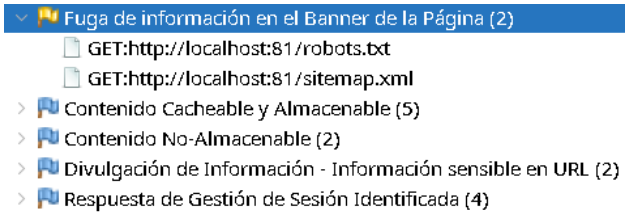
8. Banner-raren edo Footer-aren informazio-ihesa

Aplikazioaren azterketan zehar hauteman da zerbitzariaren erantzunetan eta errore-orrietan informazio tekniko gehigarria bistaratzen dela banner modura. Eduki hori ez da aplikazioaren HTML kodearen parte, baizik eta Apache-ren konfigurazio lehenetsiek automatikoki gehitzen dutena. Horrelako orrietan ohikoa da mezuak agertzea, adibidez: `Apache/2.4.25 (Debian) Server at localhost Port 80`. Horrek zerbitzariaren izena, bertsioa eta ingurune teknikoa agerian uzten ditu. Nahiz eta aplikazioak berak ez duen informazio hori esplizituki erakusten, erantzunetan dagoen banner honek informazio-ihesa sor dezake eta sistemaren konfigurazioaren xehetasunak kanpora filtratu.

Banner horietan agertzen den informazio teknikoak (Apache bertsioa, sistema eragilea edo ataka) ingurunearen egitura teknikoa identifikatzeko pista baliotsuak eskaintzen dizkio erasotzaileari. Informazio hori erabilita, erasotzaile batek zein software bertsio zehatz erabiltzen den jakin dezake eta ondoren CVE publikoetan bilatu dezake zer zaurgarritasun aprobetxa daitezkeen bertsio horretarako. Gainera, horrelako banner-ak beste informazio-ihes batzuekin (adibidez, Server edo X-Powered-By goiburuak) uztartzen badira, ingurune osoaren mapa lortu daiteke eta horrek eraso sofistikatuagoak prestatzea errazten du. OWASP 2021 sailkapenean hau Security Misconfiguration/Information Disclosure kategoriatan sartzen da.

Neurri teoriko nagusia da aplikazioak erakusten duen informazio teknikoa ezabatzea edo ezkutatzea. Horretarako:

- Apache-ren konfigurazio fitxategietan ezarri behar da: `ServerSignature Off` eta `ServerTokens Prod` lerroak gehitzea. Horrela, zerbitzariak ez du bertsioaren edo sistemaren informaziorik erakutsiko orrialdeetan edo errore-mezuetan.
- Errore-orri pertsonalizatu bat sortu behar da (`ErrorDocument 404 /errorea.html`), testu txiki batekin, erabiltzaileari ez erakusteko zerbitzariaren xehetasunik.
- Azken egiaztapena egitea gomendatzen da OWASP ZAP edo `curl` bezalako tresnen bidez, eduki informazio-ihesik ez dela gertatzen egiaztatzeko.



Irudia 13: Banner-raren edo Footer-aren informazio-ihesa

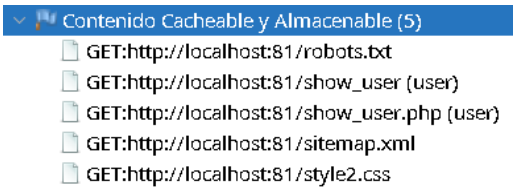
9. Cache-an Gorde Daitezken Datuak

Webguneak informazio baliagarria cache-an gordetzen uzten du, eta ez litzateke horrela izan beharko. Erabiltzailearen pasahitzak edo bestelako datuak cachean gordetzen badira, eta pertsona desberdinek gailu bera partekatzen badute, orduan aurretik sartu den erabiltzailearen informazioa eskuratu daiteke.

ZAP-ek aplikazioak zeinbait orrialde edo erantzun cachean gordetzeko baimena ematen duela detektatu du. Honek esan nahi du nabigatzaileak edo bitarteko sistemek (proxy-ak, CDN-ak…) informazio hori gordetzeko eta berrerabiltzeko aukera dutela. Arazoa sortzen da cacheatutako edukia erabiltzailearen datu pertsonalak, saio-informazioa edota pasahitzak barne hartzen baditu, izan ere, informazio hori beste erabiltzaile batek gailu berdina erabiliz eskuratu ahal izango luke.

Cachean informazio sentikorra gordetzeak baimenik gabeko sarbideak eta pribatutasun-hausturak eragin ditzake. Adibidez, erabiltzaile batek kontu pertsonal batean saioa hasi ondoren gailua partekatzen badu hurrengo erabiltzaileak cachean gordetako edukia erabiliz aurrekoaren informazioa eskuratu dezake. Horrek datu konfidentzialen ihesa, saioaren berrerabilera eta identitatearen ordezkapena eragin ditzake.

Eduki sentikorra dute orrialdeetan cachea desgaitu behar da. Horretarako, HTTP erantzun goiburu egokiak gehitu behar dira, hala nola, Cache-Control, no-cache edota no-store. Horrez gain, gomendagarria da erabiltzailearen saioarekin lotutako informazioa ez gordetzea eta erantzun dinamikoak ez cacheatzea. Horrela, ziurtatuko da erabiltzaile bakoitzak bere datu propioak soilik ikusten ditu eta ez beste norbaitenak.



Irudia 124: Cachean gorde daitezkeen datuak

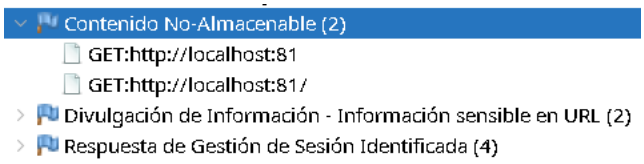
10. Cache-an Gorde Ezin Daiteken Informazioa

Aurrekoaren kontrakoa da hau, nahiz eta segurtasun-arazoa ez izan, errendimendu aldetik ez da hoberena. Errekurtso bera behin eta berriro erabiltzen eta kargatzen da, gure webgunea motelduz.

ZAP-ek adierazi du aplikazioak errekurtso batzuk cachean gordetzeko aukera ez duela gaituta. Horrek esan nahi du erabiltzaileak orrialde edo elementu bera bistaratzen duenean nabigatzaileak eduki hori berriro zerbitzaritik eskuratu behar duela. Ondorioz, webgunearen karga-denbora handiagoa izan daiteke eta zerbitzariaren baliabideak gehiago kontsumitu daitezke.

Arazo honek ez du zuzenean segurtasun arriskurik sortzen baina errendimendu-arazoak bai. Eduki estatikoak (irudiak, CSS edo JavaScript fitxategiak) cachean gordetzen ez badira, aplikazioaren erantzun-denbora luzatu eta erabiltzailearen esperientzia okertu daitezke. Gainera, trafiko gehiago sortzen da zerbitzariaren eta bezeroaren artean eta horrek baliabide gehiago kontsumitu ditzake sarean eta zerbitzarian.

Arazoa konpontzeko gomendagarria da memoria cacherako politika egoki bat ezartzea. Horretarako, HTTP cache goiburuak (adibidez, *Cache-Control*) erabiltzea komeni da, nabigatzaileak errekurtsoak lokaletan gordetzeko eta berrerabiltzeko aukera izan dezan. Horrela, webgunea arinago kargatuko da, zerbitzariaren karga txikiagoa izango da eta erabiltzailearen esperientzia hobea izango da.



Irudia 135: Cache-an gorde ezin daiteken informazioa

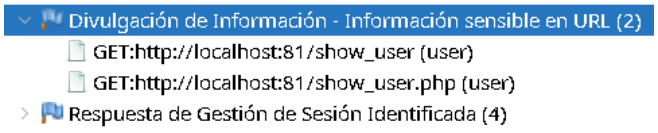
11. Informazioaren espozizioa – URL-tan informazio sentikorra

ZAP-ek antzeman du aplikazioak URL-tan informazio pribatu edo sentikorra bistaratzen duela, hala nola, tokenak, erabiltzailearen Idak edo bestelako datu pertsonalak. Kasu honetan, arazoa bi eskaeratan hauteman da: *GET http://localhost:81/show_user* eta *GET http://localhost:81/show_user.php.* Bi kasuetan erabiltzailearen informazioa URL barruan transmititzen da. Honek esan nahi du, datuak URL bidez bidaltzen direla. Ondorioz, nabigatzailearen historialean edota partekatutako esteketan informazioa gorde daiteke.

Erasotzaile batek URL horiek eskuratuz gero, datu pribatuak ikusi edo erabili ditzake eta horrek baimenik gabeko sarbideak, kontu-lapurreta edo informazio-ihesak eragin ditzake. Gainera, informazio hori cache-memorietan geratzen bada, beste erabiltzaile edo sistemek eskuratu ahal izango dute.

Arrisku hau ekiditeko hainbat gauza hartu behar ditugu kontuan. Ez da gomendagarria informazio sentikorra URL bidez bidaltzea, izan ere, nabigatzailearen historialean edo proxyetan gorde ohi dira. Horren ordez, datu horiek post eskaeren gorputzean edo goiburu seguruetan (secure headers) bidali behar dira, beti HTTPS konexio segurua erabiliz, informazioaren transmisioa zifratuta egon dadin.

Gainera, parametro sentikorrak garbitu, enkriptatu edo maskaratu behar dira zerbitzariaren aldetik eta ez dira inoiz testu garbian erakutsi behar. Nabigatzailearen historialak eta cache-memoria sistemak berrikusi eta konfiguratu behar dira datu horiek ez gordetzeko. Bestalde, komeni da token edo datu pribatuen iraupena mugatzea eta saioaren amaieran informazio hori automatikoki ezabatzea.



Irudia 146: Informazioaren espozizioa-URL-tan informazio sentikorra

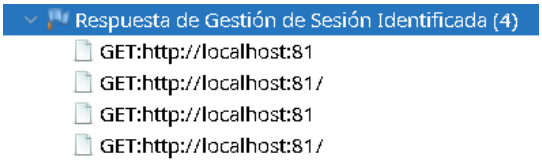
12. Saio-kudeaketaren erantzun identifikatua

Aplikazioaren erantzunetan saio-kudeaketarako token bat agertu da. Token horiek erabiltzaileen saioak identifikatzeko erabiltzen dira; beraz, behar bezala babestuta ez badaude, saioaren lapurreta edo berrerabilera gertatu daiteke.

Horrez gain, tokenak erantzunetan, sistemako erregistro-fitxategietan edo transmisio ez-seguruetan agertzeak informazio sentikorra arriskuan jartzen du. Egoera horrek baimenik gabeko sarbideak edota datu-filtrazioak eragin ditzakete.

Egoera horrek aukera ematen dio erasotzaile bati saio-tokena eskuratzeko eta ondorioz beste erabiltzaile baten saioa bereganatzeko edo saioaren jarraipena manipulatzeko. Horrelako kasuetan, autentifikazio-mekanismoak saihestu daitezke eta aplikazioaren segurtasuna ahuldu daiteke. Ondorioz, erabiltzaileen datuen konfidentzialtasuna eta sistemaren osotasuna arriskuan gera daitezke.

Arazo hau saihesteko, tokenak ez lirateke erantzunen eduki publikoetan agertu behar eta ez dira URLen bidez transmititu behar. Horren ordez, gomendagarria da tokenak cookie seguruetan gordetzea, Secure eta HttpOnly atributuak erabiliz edota goiburu pribatuen bidez bidaltzea. Gainera, saio bakoitzari identifikatzaile bakarra esleitu behar zaio eta saioa amaitzean token hori ezabatu edo beste bat sortu behar da. Saioak denbora muga egokia izan behar du eta inaktibitatearen ondoren automatikoki iraungi behar du. Azkenik, HTTPS konexio segurua beti erabili beharrezko gauza da, datuak zifratuta transmititzeko eta trafikoaren interzepzioa saihesteko.



Irudia 157: Saio kudeaketaren erantzun identifikatua

Beste segurtasun hobekuntzak

1. Pasahitzen Babesa

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

```
INSERT INTO usuarios (nombre, nan, telefonoa, jaiotze_data,
email, pasahitza)
VALUES ('$izena', '$nan', '$telefonoa', '$data', '$email',
'$pasahitza')
```

Lerro horretan `$pasahitza` aldagaia erabiltzaileak formularioan idatzitako balio berdina da; ez da inolako prozesu kriptografikorik aplikatzen. Horrek esan nahi du datu-basean pasahitzak testu arruntean gordetzen direla eta ondorioz, datu-basearen kopia gertatuz gero erabiltzaileen pasahitz guztiak zuzenean ikus daitezke.

Datu-basea konprometituko balitz (adibidez, SQL injekzio baten, zerbitzariaren errore baten edo baimen-haustura baten bidez), erasotzaileak pasahitz guztiak eskuratuko lituzke. Pasahitz berdinak erabiltzen dituzten beste webguneetan sartzeko aukera izango luke eta kontu pertsonalak lapurtu edo identitateak ordezkatu. Gainera, datu-baseak beste informazio pertsonal batzuk gordetzen baditu (emailak, NAN-ak, telefonoak), arrisku hori handitzen da, erabiltzailearen identifikazioa errazteko eta phishing edo datu-abusua egiteko.

Localhost ingurunean arriskua txikia da, ez baitago Internetera irekita, baina benetako zerbitzarian edo sare konpartitu batean arrisku altua izango litzateke, datu-basearen sarbidea edo fitxategien baimenak aldatzen badira. Horregatik, nahiz eta kasu honetan simulazio hutsa izan praktika txarra da eta benetako proiektu batean arrisku oso handia izango luke.

Errorea, lehen azaldu den moduan, `register.php` fitxategian agertzen da `INSERT INTO` komandoaren inguruan. Balio arrunta (`$pasahitza`) erabiltzaileak sartutako pasahitza da eta ez dago `password_hash()`, `hash()` edo antzeko funtziorik; hortik ondorioztatzen da pasahitza zifratu gabe gordetzen dela.

Arazo hau saihesteko, hash algoritmo seguru bat erabili beharko litzateke, salt eta pepper tekniken laguntzarekin edo seed batekin egin. Horrela, datu-basea edo komunikazioak filtratuko balira ere pasahitzak ez lirateke eskuragarriak izango.

2. HTTP Segurua

Aplikazioaren konfigurazioan ikus daiteke komunikazioa ez dela HTTPS bidez egiten, soilik HTTP portuan. Zehazki, `docker-compose.yml` fitxategian `ports: - "81:80"` lerroak erakusten du Apache zerbitzaria 80 portuan ari dela entzuten eta ez duela 443 porturik gaituta. Horrez gain, Dockerfile fitxategian ez da TLS ziurtagiririk instalatzen ezta Apache-ren SSL modulua (`a2enmod ssl`) aktibatzen. Ondorioz, garapen-ingurunean aplikazioa `http://localhost:81` bidez exekutatzen da eta datuak (login-ak, cookie-ak, token-ak...) sarean testu arruntean bidaltzen dira.

HTTP bidezko komunikazioan, datu guztiak enkriptatu gabe bidaltzen dira. Horrek esan nahi du sare berean dagoen erasotzaile batek (adibidez, Wi-Fi konexio partekatuan) erabiltzailearen kredentzialak, cookieak edo saio token-ak entzuteko aukera duela. Horrela, Man-in-the-Middle erasoak egin daitezke.

Egia esan, garapen-ingurunean (localhost) arrisku praktikoa txikia da baina arrisku teorikoa existitzen da: kontainerak sare publikoko ingurune batean exekutatuz gero, datuak ikusgai izango lirateke. Produktuzioko edo partekatutako test-inguruneetan HTTP soilik erabiltzeak arrisku oso handia dakar, batez ere saio-cookie edo tokenen transmisioa ez baita segurua.

Garraio-segurtasuna bermatzeko, zerbitzaria TLS bidez zerbitzatu behar da bai garapen-ingurunean bai produkzioan. Horrek eskatzen du Apache-ren SSL modulua aktibatzea eta ziurtagiri baliozkoak erabiltzea: garapenerako, self-signed edo CA lokal baten bidez sortutakoak eta produkziorako, autoritate ziurtatzaile (CA) batek sinatutako ziurtagiriak erabili behar dira.

Docker konfigurazioan, hau islatzen da:

- `Dockerfile`-ean, SSL modulua aktibatzea eta beharrezko ziurtagiriak kopiatzea.
- `docker-compose.yml`-ean, 443. ataka mapeatzea (eta ez soilik 80).

Halaber, zerbitzariak automatikoki bideratu behar du HTTP eskaera guztiak HTTPSra eta cookie-ei Secure, HttpOnly eta SameSite atributuak ezarri behar zaizkie.

Amaitzeko, garrantzitsua da konfigurazio-aldaketak egiaztatzea:

- Ziurtagirien baliozkotasuna eta TLS bertsio egokiak (TLS 1.2/1.3) erabiltzen direla.
- Protokolo zaharrak (SSLv2, SSLv3, TLS 1.0) desaktibaturik daudela.
- Eskaeren eta header-en bidez konfigurazioa zuzen egiaztatzea.

Erreferentziak

Cómo utilizar el encabezado de política de permisos - really simple SSL (no date) Scribd. Eskuragarri: https://es.scribd.com/document/856005916/Como-utilizar-el-encabezado-de-politica-de-permisos-Really-Simple-SSL (2025eko azaroaren 7an aipatuta).

MozDevNet (no date) Web, MDN Blog RSS. Eskuragarri: https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Headers/X-Content-Type-Options (2025eko azaroaren 7an aipatuta).

MozDevNet (no date a) *Tecnología web para desarrolladores*, *MDN Blog RSS*. *Eskuragarri*: https://developer.mozilla.org/es/docs/Web/HTTP/Reference/Headers/X-Content-Type-Options (*2025eko azaroaren 7an aipatuta*).

MozDevNet (no date b) *Web*, *MDN Blog RSS*. *Eskuragarri: https:/developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Headers/Content-Security-Policy* (*2025eko azaroaren 7an aipatuta*).

MozDevNet (no date b) *Web*, *MDN Blog RSS*. *Eskuragarri*: https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Headers/X-Frame-Options (*2025eko azaroaren 7an aipatuta*).

MozDevNet (no date a) *Tecnología web para desarrolladores*, *MDN Blog RSS*. Eskuragarri: https://developer.mozilla.org/es/docs/Web/HTTP/Reference/Headers/Set-Cookie(*2025eko azaroaren 7an aipatuta*).

Cookie without SameSite attribute (no date) *ZAP By Checkmarx*. Eskuragarri: https://www.zaproxy.org/docs/alerts/10054-1/(*2025eko azaroaren 7an aipatuta*).

Password storage cheat sheet¶ (no date) *Password Storage - OWASP Cheat Sheet Series*. Eskuragarri:https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html(*2025eko azaroaren 7an aipatuta*).

A02:2021 – Cryptographic Failures (no date) *A02 Cryptographic Failures - OWASP Top 10:2025 RC1*. Eskuragarri: https://owasp.org/Top10/A02_2021-Cryptographic_Failures/ (*2025eko azaroaren 7an aipatuta*).

HTTP security response headers cheat sheet¶ (no date) *HTTP Headers - OWASP Cheat Sheet Series*. Eskuragarri: https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Headers_Cheat_Sheet.html (*2025eko azaroaren 7an aipatuta*).

A05:2021 – security misconfiguration (no date) *A05 Security Misconfiguration - OWASP Top 10:2025 RC1*. Eskuragarri: https://owasp.org/Top10/A05_2021-Security_Misconfiguration/ (*2025eko azaroaren 7an aipatuta*).

K, A.P. (no date) *Apache web server hardening*. Eskuragarri: https://beaglesecurity.com/blog/article/apache-web-server-hardening.html (*2025eko azaroaren 7an aipatuta*).

WSTG - latest (no date) *OWASP Foundation*. Eskuragarri: https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/01-Information_Gathering/05-Review_Web_Page_Content_for_Information_Leakage (*2025eko azaroaren 7an aipatuta*).

Password_hash - manual (no date) *PHP*. Eskuragarri: https://www.php.net/manual/en/function.password-hash.php (*2025eko azaroaren 7an aipatuta*).

Escaneo de Vulnerabilidades Automático Con Owasp Zap (no date) *Seguridad Cero Academy*. Eskuragarri: https://academy.seguridadcero.com.pe/blog/escaneo-vulnerabilidades-autom%C3%A1tico-OWASP-ZAP (*2025eko azaroaren 7an aipatuta*).