Penetratie Rapport

Groep 1:
Bram Jonkers
Leon Dijkstra
Lukas Cremers
Stefan Suk
Wiljan Siderius

27 maart 2024

Inhoudsopgave

1 Inleiding

Voor het project security en bcm is er een website gemaakt waarop mensen die klant zijn bij een energiebedrijf hun verbruik kunnen zien en waar medewerkers de nodige functies hebben om hun werk te doen. Om er voor te zorgen dat deze website veilig is, is het nodig om penetratie tests te doen en deze te documenteren in een penetratie rapport. Penetratietesten worden uitgevoerd om de kwetsbaarheden van het systeem te identificeren en te verhelpen voordat deze worden misbruikt door kwaadwillende actoren.

2 Doel van de test

Het doel van deze penetratietest is om de beveiliging van het systeem te verbeteren door potentiële zwakke punten bloot te leggen die kunnen leiden tot datalekken of hacking. Hierdoor worden de gegevens van gebruikers en werknemers beter beschermd en wordt de continuïteit van het bedrijf gewaarborgd. De test omvat zowel de website als de server zelf. Deze penetratietest houdt dus in dat de website uitvoerig wordt getest op bugs en/of fouten en op mogelijke zwaktes.

3 Methodologie

De testen worden uitgevoerd op verschillende manieren. Zo worden er dingen handmatig door de ICT studenten getest en worden andere dingen automatisch met behulp van tools getest.

De volgende dingen worden getest:

Black-box testen:

- SQL injectie.
- XSS (cross site scripting).
- command injectie.
- scannen met tools.

White-box testen:

- Code-analyse handmatig.
- Code-analyse geautomatiseerd met PhpStan.
- Inbreken op SSH connectie.

4 Ontdekkingen en kwetsbaarheden

Dit gedeelte beschrijft de gedetailleerde bevindingen van de penetratietest, waaronder alle geïdentificeerde kwetsbaarheden en beveiligingslekken.

4.1 SQL injectie

Bij verschillende onderdelen van de site is er een risico dat er een SQL injectie mogelijk is. Een aantal dingen, zoals de inlogpagina worden niet met SQL gedaan, maar via LDAP. Dit is al beveiligd en wordt geregeld door de Apache webserver. Op de pagina waar je je kunt aanmelden is wel een risico op SQL injectie, omdat hier wel informatie van de klant wordt opgeslagen in de database. Bij het verbinding maken met de database wordt er gebruikgemaakt van prepared statements waardoor er geen SQL injectie meer mogelijk is.

4.2 XSS (cross site scripting)

XSS is een vorm van injectie waarbij vanuit de browser een script kan worden uitgevoerd, dit kan voorkomen op elke plaats waar een gebruiker dingen in kan voeren. Omdat er geen goede bescherming tegen dit soort aanvallen is, is de kans op een XSS aanval vrij hoog.

4.3 Command injectie

Bij het testen voor de mogelijkheid om command injectie uit te voeren is er gebleken dat wij geen mogelijke punten hebben kunnen vinden waar dit mogelijk was, er is namelijk nergens in de website de mogelijkheid om een command line commando uit te voeren en zonder de juiste rol voor de URL waar je naar toe wilt, kan je ook niet parameter swappen in de URL’s

4.4 Scannen met tools

Rapport Nikto security scan:

Algemene informatie:

- Nikto versie: 2.5.0
- Doel IP: 192.168.243.41
- Doel Poort: 80,88,443
- Starttijd: 2024-03-21 11:44:36 (GMT 1)
- Eindtijd: 2024-03-21 11:44:55 (GMT 1)

Server informatie:

- Server: Apache/2.4.57 (Debian)
- Rootpagina: https://192.168.243.41/
- Apache versie: Verouderd (huidige is minstens 2.4.58)

Poort 80 bevindingen:

Geen CGI-directories gevonden.
SSL-gegevens tonen een certificaat met onderwerpen en alternatieve namen die mogelijk niet overeenkomen met de hostname.
Geen Strict-Transport-Security HTTP-header gedefinieerd.
Geen X-Content-Type-Options header ingesteld.
Content-Encoding header is ingesteld op "deflate", wat kan duiden op kwetsbaarheid voor de BREACH-aanval.
Webserver retourneert een geldige respons met onbekende HTTP-methoden.
/server-status onthult Apache-informatie.
Directory-indexering gevonden op /css/, /files/, /test/, /database/, /images/, /docs/.

Poort 443 bevindingen:

SSL-informatie: Onderwerp / Alternatieve namen, geen overeenkomst met certificaatnaam.
Geen Strict-Transport-Security HTTP-header gedefinieerd.
Geen X-Content-Type-Options header ingesteld.
Geen CGI-directories gevonden.
Apache-versie verouderd.
Content-Encoding header is ingesteld op "deflate".
/server-status onthult Apache-informatie.
Directory-indexering gevonden op /css/, /files/, /test/, /database/, /images/, /docs/.
Potentiële gevoelige bestanden onthuld, zoals test.php, .git-bestanden, README.md, en JetBrains-projectbestanden.

4.5 Handmatige code-analyse

Tijdens de handmatige code-analyse als onderdeel van de penetratietest zijn er een aantal dingen gevonden. Zo waren er stukken code waar sql injectie mogelijk was en stukken waar de code slordig en onoverzichtelijk was.

Potentiële zwakheden zijn geïdentificeerd in de input validatie van gebruikersinvoer, waardoor het risico op SQL-injectie en cross-site scripting (XSS) wordt vergroot.

Er zijn mogelijke kwetsbaarheden geconstateerd in het authenticatie- en autorisatiemechanisme van de applicatie, waaronder onvoldoende beveiligde sessiebeheer en toegangscontrole.

Enkele delen van de code tonen risico's met betrekking tot het onbedoeld blootstellen van gevoelige informatie, zoals het versturen van wachtwoorden in plain text.

De implementatie van beveiligings headers zoals Content Security Policy (CSP) en X-Content-Type-Options ontbreekt, waardoor de applicatie kwetsbaar kan zijn voor bepaalde aanvallen.

De foutafhandeling is inconsistent en kan mogelijk gevoelige informatie onthullen aan aanvallers, wat kan bijdragen aan het uitvoeren van aanvallen zoals SQL-injectie.
De handmatige code-analyse heeft verschillende potentiële kwetsbaarheden aan het licht gebracht die moeten worden aangepakt om de algehele beveiliging van de applicatie te verbeteren.

4.6 Geautomatiseerde code-analyse met PhpStan

PHPStan is een analysetool voor PHP-code die kan helpen bij het identificeren van potentiële fouten zoals bugs en kwetsbaarheden in de broncode van de applicatie. Door PHPStan te gebruiken in het ontwikkelproces kunnen we proactief beveiligingslekken identificeren en oplossen voordat de code in productie wordt uitgerold, PHPStan kan testen op veel standaard aanvallen en het is aangeraden om PHPStan vaker dan één keer te gebruiken.

hieronder zijn de overgebleven bugs na het meerdere keren gebruik van PHPStan, na het analyseren van deze bugs hebben wij ondervonden dat het risico niet hoog genoeg is om deze nog optelossen

intranet/klantenservice/index.php:
Regel 39: De functie search() heeft geen retourtype gespecificeerd. Je wilt mogelijk het retourtype specificeren als het de bedoeling is om iets terug te geven.
intranet/ldap_support.inc.php:
Regel 25: De functie ConnectAndCheckLDAP() retourneert geen false, dus dit kan worden verwijderd uit het retourtype. Zorg ervoor dat de documentatie van de functie overeenkomt met het gedrag.
Regel 54, 76, 97, 139, 171, 231, 261, 293, 315, 333: Diverse functies hebben ontbrekende retourtypen en parameter typen. Specificeer retourtypen en parameter typen voor betere code documentatie en typeveiligheid.
Regel 79, 80, 81, 139, 184, 187, 192, 242, 253, 278, 279, 300, 315, 323, 333, 341: Parameter type mismatch of toegang tot array-offsets op niet-array waarden. Controleer de datatypes en zorg voor een juiste afhandeling.
Regel 141: Er is aangegeven dat een if-voorwaarde altijd waar is. Controleer de logica van de voorwaarde.

Regel 231: Het retourtype mist een waarde type gespecificeerd in iterable type array. Je moet het waarde type specificeren of ervoor zorgen dat de functie altijd een array retourneert.
intranet/manager/createNewUser.php:
Regel 24, 98: Functies createuser() en checkUser() hebben geen retourtype gespecificeerd. Overweeg het specificeren van de retourtypen voor duidelijkheid.
intranet/manager/editUser.php:
Regel 34, 39, 57, 86: Parameter type mismatch voor LDAP functies. Zorg ervoor dat de juiste parameter typen worden doorgegeven.
intranet/manager/index.php:
Regel 17, 84, 89, 119: Variabelen zijn mogelijk niet gedefinieerd, en er is een parameter type probleem in regel 89. Zorg ervoor dat variabelen correct worden geïnitialiseerd en controleer de parameter typen.

4.7 Toegang verkrijgen via SSH

Het was mogelijk door middel van het verkrijgen van inloggegevens of bruteforcing om op afstand toegang te krijgen tot de server waarop de website zich bevindt. Dit is inmiddels voorkomen door het inloggen met wachtwoord compleet te verbieden en in plaats daarvan gebruik te maken van public key authentication. Het is echter nog wel mogelijk om toegang te verkrijgen tot de server als er toegang wordt verkregen tot een apparaat met toestemming. Dit kan alleen voorkomen worden door het minimaliseren van mogelijkheden voor een derde partij om een apparaat met toestemming te kunnen gebruiken, en het op tijd melden van diefstal.

4.8 HTTP

De website maakte gebruik van HTTP wat risico's veroorzaakt voor gebruikers van de website. Dit is inmiddels verbeterd en de website gebruikt nu HTTPS. Dit is gedaan door het genereren van een self-signed certificate met behulp van OpenSSL. Door het toevoegen van het .pem bestand waarmee het certificaat is gegenereerd aan de browser wordt het certificaat correct geaccepteerd. Dit is niet geschikt voor een publiek toegankelijke website, maar gezien de website nog niet publiek toegankelijk is, is het ook niet mogelijk om een publiek erkend certificaat te verkrijgen. Voor het verkrijgen van een echt certificaat kan gebruik worden gemaakt van Let's Encrypt en certbot wanneer de website publiek toegankelijk wordt, maar dit is nu nog niet mogelijk.

5 Risicobeoordeling

Tijdens de penetratie tests zijn er vrij veel risico's gevonden. De oorzaken die voor de risico's met de grootste consequenties zorgen zullen worden verholpen zodat de website en server beter beveiligd zijn.

Op alle plekken waar gebruikersinvoer mogelijk is, is het erg belangrijk om in de gaten te houden of er geen SQL injectie mogelijk is. Er is wel een controle op SQL injectie voor de connectie met de database gemaakt wordt waardoor er, als het goed is, geen risico is voor SQL injectie.

Om XSS te voorkomen is het implementeren van strikte validatie van gebruikersinvoer nodig om ervoor te zorgen dat scripts van kwaadaardige aard niet kunnen worden uitgevoerd.

Ook is het belangrijk om de code netjes en consistent te houden. Hierdoor is het makkelijker om eventuele fouten of kwetsbaarheden te vinden en te verbeteren. Doordat dit nu niet overal het geval is, is er een verhoogd risico op kwetsbaarheden.

Uit de code analyse van Nikto zijn niet zo zeer risico's gebleken. Wel zijn er een aantal aanbevelingen gevonden:
Update Apache: De huidige Apache-versie is verouderd, update naar minstens versie 2.4.58 om bekende kwetsbaarheden te verhelpen.
Configureer SSL-correcties: Zorg ervoor dat de SSL-certificaten overeenkomen met de hostname en configureer de Strict-Transport-Security HTTP-header.
Headers instellen: Implementeer de X-Content-Type-Options header om MIME-type sniffing aan te pakkekut flikker die onze code stuk maaktn.
Beveilig directory-indexering: Schakel directory-indexering uit om onbedoelde toegang tot gevoelige informatie te voorkomen.
Beveilig gevoelige bestanden: Beperk de toegang tot gevoelige bestanden en directories zoals /test.php en .git-bestanden.
Beperk Apache-informatie: Schakel Apache-status pagina's uit of beperk de toegang ertoe om het risico van informatie-uitlekken te verminderen.

Gebaseerd op de ontdekte kwetsbaarheden, worden de volgende aanbevelingen gedaan om de beveiliging van de PHP-website te verbeteren:

Implementeer een Content Security Policy om te bepalen welke bronnen de browser mag laden.
Voer regelmatig geautomatiseerde en handmatige beveiligingsscans uit om kwetsbaarheden op te sporen en te verhelpen.

6 Conclusie

Op basis van de uitgevoerde penetratietest kan worden geconcludeerd dat de PHP-website kwetsbaarheden vertoont voor verschillende aanvallen, waaronder SQL injectie en XSS. Het implementeren van de aanbevolen maatregelen kan de beveiliging van de website verbeteren en de risico's verminderen.

belangrijke bevindingen zijn het ontbreken van een goede input validatie terwijl er wel input validatie is gebruikt is er wel ondervonden dat het nog niet goed genoeg was. het versturen van de brief voor een goede authenticatie is niet mogelijk binnen het project, daardoor is op het moment de authenticatie niet volledig werkend wanneer er een account wordt aangemaakt.

aanbevolen maatregelen ter verbetering zijn onder meer het implementeren van striktere validatie voor het beveiligen tegen XSS en het beveiligen van database invoer om SQL injectie te voorkomen.

Door deze maatregelen te implementeren, zal de beveiliging van de website verbeterd worden en zullen de risico's een stuk verminderd zijn.

7 Referenties

CVE - CVE-2024-24041. (n.d.). https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2024-24041

CVE - CVE-2024-28746. (n.d.). https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2024-28746

CVE - CVE-2024-28752. (n.d.). https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2024-28752

Hasson, E. (2023, December 20). *What is Penetration Testing | Step-By-Step Process & Methods | Imperva*. Learning Center. https://www.imperva.com/learn/application-security/penetration-testing/

CVE - Search Results. (n.d.). https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=OpenLDAP+2.5.13

Cross Site Scripting (XSS) | OWASP Foundation. (z.d.). https://owasp.org/www-community/attacks/xss/#:~:text=Overview,to%20a%20different%20end%20user.