LOG8371- Ingénierie de la qualité en logiciel

Hiver 2023

Travaille pratique #3: Securité

Groupe Peel

2088821 – Laurent Quoc Hoa Nguyen

1995039 - Labib Bashir-Choudhury

1956802- Dawut Esse

1954607 - Victor Kim

2085085 - Yasser Kadimi

Soumis à : Armstrong Tita Foundjem

Date de Remise : 19 avril 2023

1. Application 1 (Vulnerable SAML App)

1.1. Introduction

SonarCloud est un service basé sur le cloud qui offre une analyse de code automatisée et une gestion continue de la qualité du code. Il aide les équipes de développement de logiciels à détecter et à résoudre les problèmes de code tôt dans le cycle de développement, ce qui conduit finalement à une meilleure qualité de code, à une fiabilité logicielle améliorée et à des cycles de publication plus rapides. SonarCloud offre une gamme de fonctionnalités, notamment l'analyse de code statique, les mesures de couverture de code, la détection de duplication de code et l'analyse de vulnérabilité. Il s'intègre aux outils de développement populaires tels que GitHub, GitLab et Bitbucket, ce qui permet aux équipes de l'intégrer facilement dans leurs flux de travail existants. Dans l'ensemble, SonarCloud est un outil puissant pour les développeurs qui cherchent à améliorer la qualité et la sécurité de leur code.

Vulnérable SAML App est une application qui contient plusieurs vulnérabilités. Le code est écrit en python et HTML. Pour déployer, tout ce qui faut faire c'est d'exécuter un fichier DockerFile avec la commande : *docker-compose up*.

L'application s'ouvre dans l'url suivante: http://127.0.0.1:8000. L'application a deux composants, l'idp et le sp. L'idp est le fournisseur d'identité, il est responsable de l'authentification. Le sp est le fournisseur de services. Ceci représente l'application dont l'utilisateur essaie d'accéder. Avec ces deux composantes, les développeurs peuvent bien observer leurs vulnérabilités.

1.2. l'analyse statique avec SonarCloud

Figure 1 : Analyse statique de Vulnerable SAML avec SonarCloud

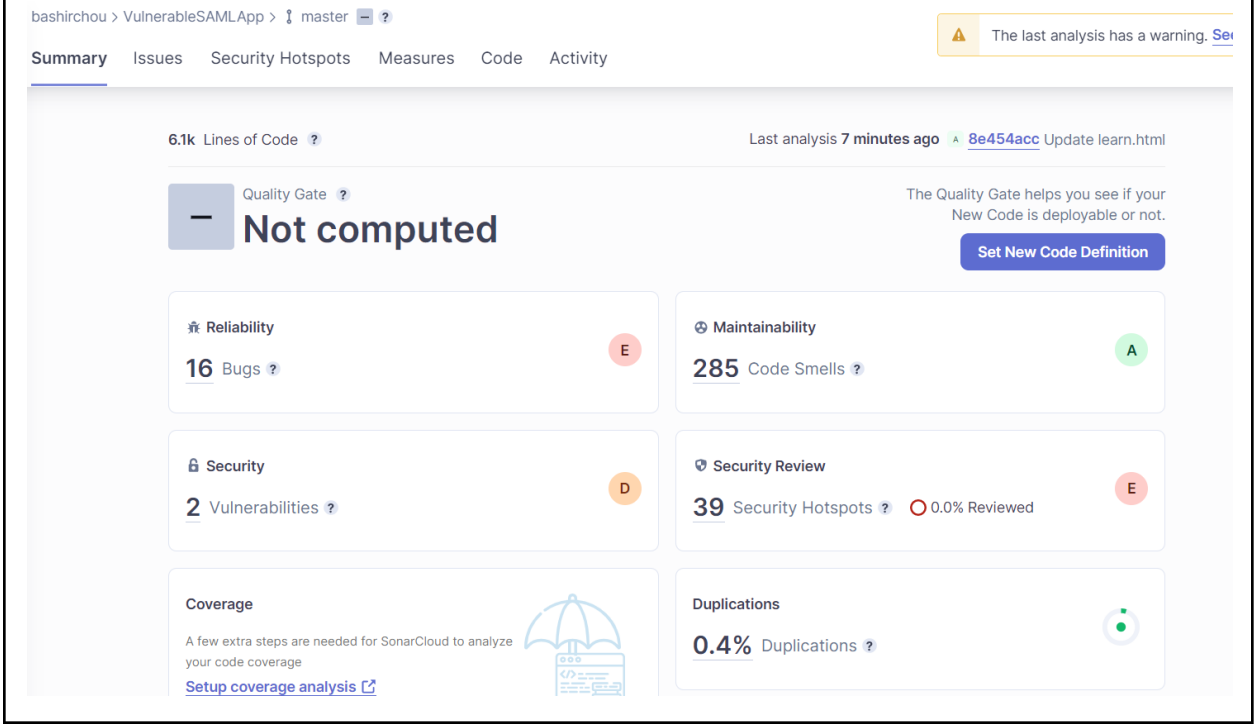

Tableau 2 : Vulnérabilités de Vulnerable SAML détectés par SonarCloud

Vulnérabilité	Nom du fichier	Severité	Type	Risque	Recommandation
1	vulnerablesp/src/onelogin/saml2/idp_metadata_parser.py	Critique	Cryptographie Faible, A3 - Sensitive Data Exposure (OWASP), A6 - Security Misconfiguration	Possiblement avoir des attaques de homme au milieu si les SSL/TLS n'est pas sécuritaire.	La validation de X.509 certificats est essentielle pour créer des sessions sécurisées de SSL/TLS.
2	vulnerablesp/Dockerfile	Moyen	Permission	Dans les dockerfiles, il y a des directives COPY et ADD qui sont utilisées de façon récursive. Il est possible que des fichiers inattendus s'ajoutent quand on	Nous pouvons arranger ce problème en limitant l'utilisation de COPY et ADD. Nous pouvons aussi éviter de copier tout le répertoire qui contient les fichiers.

				exécute cette dockerFile. Cela peut affecter des données confidentielles.	
3	vulnerablesp/yogiSP/vulnsp.py	Moyen	Cryptographie faible	L'utilisation des chiffres pseudo-aléatoires n'est pas sécuritaire. Quand un logiciel génère des valeurs prévisibles, il est possible que l'attaquant peut deviner la prochaine valeur. Cela peut permettre d'imiter un autre utilisateur.	Utiliser des générateurs des nombres aléatoires recommandés par OWASP. Ensuite, il faut s'assurer de seulement utiliser les nombres aléatoires générés une fois.
4	configure_platform.py	Basse	Encryption des données sensitives.	Cette application utilise des protocoles text clair comme ftp, telnet or http. Ces derniers ne chiffrent pas les données transportées. Les attaques peuvent lire, modifier ou corrompre le contenu qui est transporté.	Transférer les données avec un protocole TLS et SSH qui est authentifié et encrypté. SSH est une meilleure alternative de telnet.
5	vulnerablesp/src/onelogin/saml2/utils.py	Haut	Cross-Site Request Forgery	MD2, MD4, MD5, MD6, HAVAL-128, HMAC-MD5, DSA sont des algorithmes hash qui peuvent avoir des collisions, alors cela augmentera la chance de succès d'une attaque brute.	Il faut utiliser des algorithmes plus sécurisés comme SHA-256, SHA-512 ou SHA-3. Pour le hachage des mots de passeport comme bcrypt, scrypt, argon2 ou pbkdf2 pour ralentir les attaques brutes.

6	vulnerablesp/yogiSP/vulnsp.py	Haut	Cross-Site Request Forgery	L'application utilise GET, HEAD, OPTIONS, qui sont des méthodes HTTP sécurisées. Les méthodes non-sécurisées sont POST, PUT et DELETE. Utiliser des méthodes sécurisées et non sécurisées en même temps peut impacter la sécurité.	Pour toutes les routes et les contrôleurs d'une application, les méthodes HTTP autorisées sont explicitement définies.Les méthodes HTTP qui sont considérées comme sécuritaires peuvent seulement de performer des opérations de lecture.
7	vulnerableidp/Dockerfile	Moyenne	Permission	Quand des conteneurs sont exécutés avec un utilisateur privilégié peut nuire à leur sécurité d'exécution. Ceci peut permettre à n'importe quelle utilisateur de réaliser des actions administratives	Pour arranger ce problème, il faut créer un utilisateur par défaut comme postgresql ou zookeeper.
8	/configure_platform.py	Baisse	Encryption des données sensitives.	Cette application utilise des protocoles clair-text comme ftp, telnet or http. Ce-ci n'en crypte pas les données transportées. Les attaques peuvent lire, modifier ou corrompre le contenu du transport.	Transférer les données avec un protocole TLS et SSH qui est authentifié et encrypté. SSH est une meilleure alternative de telnet.
9	vulnerablesp/src/onelogin/saml2/constants.py	Baisse	Encryption des données sensitives.	Cette application utilise des protocoles clair-text comme ftp, telnet or http. Ce-ci n'en	Transférer les données avec un protocole TLS et SSH qui est authentifié et encrypté. SSH est une

				crypte pas les données transportées. Les attaques peuvent lire, modifier ou corrompre le contenu du transport.	meilleure alternative de telnet.
10	/vulnerablesp/src/onelogin/saml2/response.py	Baisse	Encryption des données sensitives.	Cette application utilise des protocoles clair-text comme ftp, telnet or http. Ce-ci n'en crypte pas les données transportées. Les attaques peuvent lire, modifier ou corrompre le contenu du transport.	Transférer les données avec un protocole TLS et SSH qui est authentifié et encrypté. SSH est une meilleure alternative de telnet.

1.2.1. Description des vulnérabilités

En termes d'OWASP top 10, les vulnérabilités détectées par **SonarCloud** sont **Sensitivity Data Exposure** et **Security Misconfiguration**. Le reste des vulnérabilités décrites dans le tableau précédent sont selon les **Security HotSpots**. Cette section est dédiée à décrire ces vulnérabilités de façon plus générale.

1.2.1.1. Cryptographie Faible

Cette vulnérabilité se produit quand l'application utilise des algorithmes de chiffrement qui ne sont plus considérés assez sécurisés pour protéger les données. Plusieurs attaques peuvent exploiter cette vulnérabilité comme des attaques force brute pour essayer de déchiffrer les messages chiffrés.

1.2.1.2. Permission

Cette vulnérabilité est due au fait que sans préciser les fichiers et les dossiers il est possible que certains fichiers avec des excès de permissions peuvent être copiés dans le conteneur et des attaques peuvent exploiter ces permissions pour avoir accès à des données sensibles.

1.2.1.3. Encryption des données

L'utilisation de certains protocoles qui ne chiffre pas les données transmises peut poser un problème, car on peut voir les données en texte clair. Les données sensibles peuvent facilement être volées, modifiées.

1.2.1.4. Cross-Site Request Forgery

Cette attaque est possible quand l'application ne bloque pas l'exécution des requêtes malveillantes des autres sites. Cette vulnérabilité peut permettre à un utilisateur de réaliser des opérations en tant qu' utilisateur légitime sans son consentement.

1.2.1.5. Sensitive Data Exposure

Cette attaque est possible quand les données ne sont pas bien encryptées. Les données comme les mots de passe peuvent être exposées.

1.2.1.6. Security Misconfiguration

Cette attaque est seulement possible quand l'application ou le système n'est pas bien configuré. Par exemple, des faibles mots de passe ou des ports ouvertes sont des causes de ce type de vulnérabilité.

1.3. Test d'intrusions avec ZAP

Figure 2 : Résultats des tests d'intrusions avec ZAP de Vulnerable SAML

Alert type	Risk	Count
Content Security Policy (CSP) Header Not Set	Medium	5 (71.4%)
Missing Anti-clickjacking Header	Medium	3 (42.9%)
Vulnerable JS Library	Medium	1 (14.3%)
Big Redirect Detected (Potential Sensitive Information Leak)	Low	1 (14.3%)
X-Content-Type-Options Header Missing	Low	7 (100.0%)
Information Disclosure - Suspicious Comments	Informational	1 (14.3%)
User Agent Fuzzer	Informational	24 (342.9%)
Total		7

1.3.1. Description des vulnérabilités

Cette section va décrire les différentes vulnérabilités que ZAP à détecter lors des tests d'intrusions. Vulnerable SAML présente 7 vulnérabilités, comme présenté à la figure 2, que nous allons décrire et rapporter selon les classements OWASP Top 10 en 2021, 2017 et 2013.

1.3.1.1. Content Security Policy (CSP) Header Not Set

L'alerte "Vulnerable SAML App : Content Security Policy (CSP) Header Not Set" indique que l'application SAML en question ne définit pas de manière appropriée la politique de sécurité du contenu (CSP) dans l'en-tête de la réponse HTTP. Cela peut permettre à des attaquants de mener des attaques de type cross-site scripting (XSS) et d'autres attaques basées sur l'injection de code malveillant dans les pages web. Il est donc recommandé de configurer correctement la politique de sécurité du contenu en définissant l'en-tête CSP approprié pour limiter les sources de contenu autorisées, les types de contenu autorisés et les politiques de gestion des erreurs. Cela peut aider à renforcer la sécurité de l'application et à prévenir les attaques basées sur l'injection de code malveillant. [7]

1.3.1.2. Missing Anti-clickjacking Header

Ça indique que l'application SAML en question ne définit pas de manière appropriée le header X-Frame-Options dans la réponse HTTP. Le header X-Frame-Options permet de se protéger contre les attaques de type Clickjacking en limitant la façon dont une page web peut être incorporée dans une iframe. Si ce header n'est pas défini, un attaquant peut utiliser une attaque de Clickjacking pour piéger un utilisateur en cliquant sur un élément invisible ou caché sur la page, ce qui peut permettre à l'attaquant de prendre le contrôle de la session de l'utilisateur ou de voler des informations confidentielles. Il est donc recommandé de configurer correctement le header X-Frame-Options pour limiter la façon dont une page peut être incorporée dans une iframe. Par exemple, vous pouvez définir la valeur de l'en-tête sur "DENY" pour empêcher toute incorporation de la page dans une iframe, ou sur "SAMEORIGIN" pour permettre uniquement l'incorporation de la page dans des iframes de même origine. Cela peut aider à renforcer la sécurité de l'application et à prévenir les attaques de Clickjacking. [8]

1.3.1.3. Vulnerable JS Library

L'alerte "Vulnerable JS Library" indique que l'application utilise une bibliothèque JavaScript qui présente une vulnérabilité connue, ce qui peut permettre à un attaquant de compromettre la sécurité de l'application. Il est important de vérifier régulièrement les bibliothèques utilisées et de les mettre à jour pour prévenir les vulnérabilités connues. [9]

1.3.1.4. Big Redirect Detected (Potential Sensitive Information Leak)

L'alerte "Big Redirect Detected (Potential Sensitive Information Leak)" indique qu'une redirection potentiellement dangereuse a été détectée dans l'application, ce qui peut mettre en

danger la sécurité des utilisateurs et des informations sensibles. Il est important de vérifier la redirection et de s'assurer qu'elle est légitime. [10]

1.3.1.5. X-Content-Type-Options Header Missing

L'alerte "X-Content-Type-Options Header Missing" indique que l'application ne définit pas correctement l'en-tête X-Content-Type-Options dans la réponse HTTP, ce qui peut exposer l'application à des attaques de type MIME-Sniffing. Il est important de définir cet en-tête pour indiquer aux navigateurs web de respecter le type de contenu déclaré. [11]

1.3.1.6. Information Disclosure - Suspicious Comments

L'alerte "Information Disclosure - Suspicious Comments" indique que des commentaires suspects ont été détectés dans le code source de l'application, qui peuvent divulguer des informations sensibles. Il est important de les vérifier et de les supprimer si nécessaire, ainsi que de sensibiliser les développeurs à l'importance de ne pas inclure d'informations confidentielles dans les commentaires du code. [12]

1.3.1.7. User Agent Fuzzer

L'alerte "User Agent Fuzzer" indique qu'une tentative de manipulation de l'agent utilisateur a été détectée, ce qui peut être utilisé par un attaquant pour tromper l'application ou masquer son identité. Il est important de détecter et de bloquer ces tentatives pour éviter les attaques potentielles. [13]

1.4. Comparaisons des résultats de Sonar Cloud et ZAP

Sonar Cloud et ZAP détectent des vulnérabilités différentes. Dans Sonar Cloud, la majorité des vulnérabilités concerne une faible encryption des données. En ZAP, nous avons plusieurs vulnérabilités différentes et il ne s'aligne pas avec celui de Sonar Cloud. Il n’a pas d’alternative en OWASP. La raison pour laquelle les deux logiciels donnent des résultats différents est à cause du fait qu’ils utilisent des différentes approches. ZAP simulent des attaques tandis que Sonar Cloud observe le code pour déduire quelles sont les vulnérabilités.

2. Application 2 (WackoPicko)

2.1. Introduction

WackoPicko est une application web conçue pour tester les outils de tests de sécurité, tels que SonarCloud et OWASP ZAP.[2] Elle permet donc la formation et la sensibilisation à la sécurité informatique. L'application a été développée en utilisant le langage de programmation PHP et utilise une base de données MySQL pour stocker les données de l'application. Elle est disponible en open source et peut être facilement installée et déployée sur un serveur web. Les développeurs de cette application ont aussi créé une image docker de cette application afin que les utilisateurs et testeurs puissent y déployer facilement. Dans le cadre de cette étude, nous avons rouler la commande suivante proposée par les auteurs du projet afin d'installer le projet et le déployer : *docker run -p 8080:80 -it adamdoupe/wackopicko.* Ainsi, un conteneur sera créé avec l'image de l'application, que nous pouvons accéder en *localhost,* au port 8080.

L'application WackoPicko contient une variété de vulnérabilités courantes, telles que des failles XSS, CSRF, Injection et autres. Les utilisateurs peuvent utiliser ces vulnérabilités pour apprendre à identifier les failles de sécurité et les corriger. Les prochaines sections qui vont suivre présenteront l'analyse statique de l'application avec SonarCloud, les tests d'intrusion avec OWASP ZAP et enfin une comparaison des résultats entre ces deux outils de tests de sécurité.

2.2. l'analyse statique avec SonarCloud

Figure 3 : Analyse statique de WackoPicko avec SonarCloud

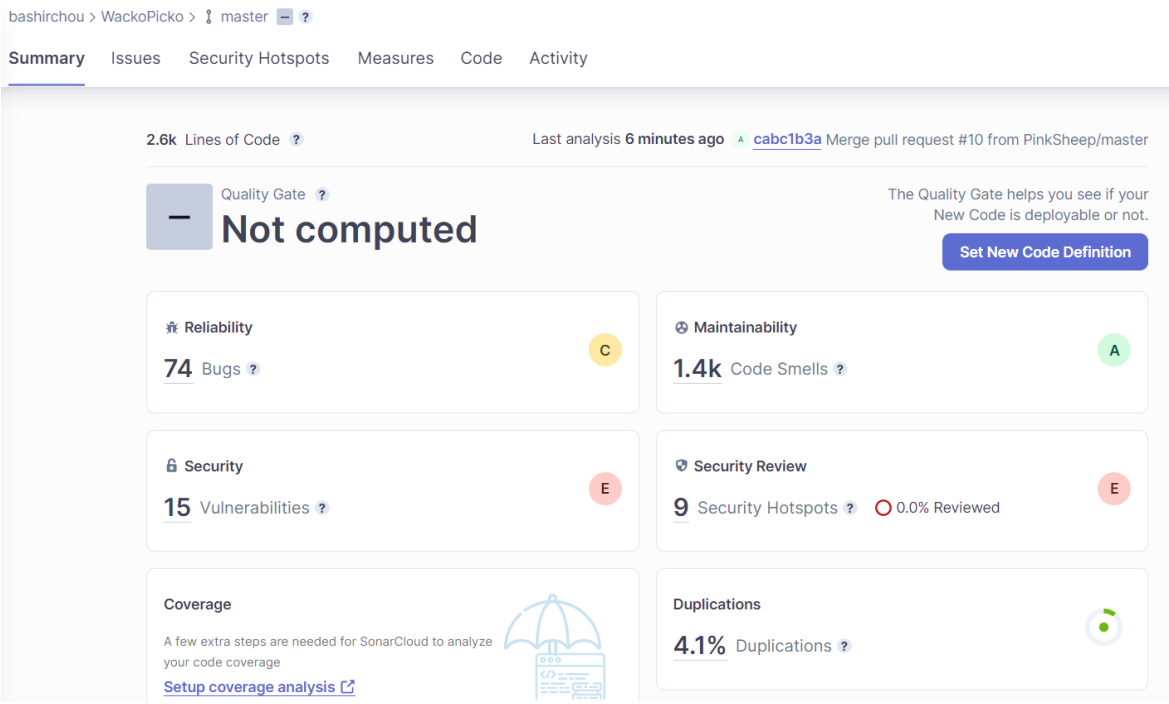

Tableau 2 : Vulnérabilités de WackoPicko détectés par SonarCloud

Vulnérabilité	Nom du fichier	Severité	Type	Risque	Recommandation
1	website/admin/index.php	Bloquante	Cross-Site Scripting (XSS)	L'utilisateur pourrait permettre à un attaquant de contrôler les fichiers inclus.	La stratégie de mitigation devrait être basée sur la liste blanche des valeurs autorisées ou la conversion en types sécurisés.
2	website/include/guestbook.php	Bloquante	Injection de SQL	Les attaquants peuvent exploiter cette vulnérabilité	Nous utilisons des requêtes paramétrées.

				pour exécuter des commandes malveillantes dans les bases de données.	
3	website/passcheck.php	Bloquante	Injection de commande	Un attaquant peut exécuter des commandes de système d'exploitation. Il peut compromettre l'application et prendre toutes les données.[1]	Une première suggestion est de ne pas utiliser les commandes OS. S'il n'est possible, nous pouvons utiliser des commandes pré-approuver.
4	website/users/register.php	Bloquante	Injection de SQL	Les attaques peuvent injecter des données dans un champ vulnérable pour ensuite exécuter des commandes malicieuses.	Nous utilisons des requêtes paramétrées.
5	website/users/login.php	Bloquante	Injection de SQL	L'attaque peut supprimer des données dans les bases de données.	Nous utilisons des requêtes paramétrées.
6	website/submitname.php	Bloquante	Cross-Site Scripting (XSS)	Usurpation d'identité	Le lien forgé peut injecter du code malicieux dans l'application web. Ceci peut causer une usurpation d'identité grâce aux vols de cookies.
7	website/pictures/upload.php	Bloquante	Injection de traversée de chemin	Il a un risque d'écraser et de supprimer les fichiers arbitraires.	La façon pour prévenir cette injection de chemin est de valider les chemins qui viennent des sources non fiables.

8	website/include/admins.php	Baisse	Configuration non sécurisée	Quand un cookie avec l'attribut de sécurité est mis à vrai, il ne peut pas envoyer une requête HTTP.	Il est recommandé d'utiliser HTTPS partout, alors en mettant le flag secure à vrai sera le comportement par défaut quand on crée des cookies.
9	Dockerfile	Moyenne	Permission	Quand des conteneurs sont exécutés avec un utilisateur privilégié, cela peut nuire à leur sécurité d'exécution. Ceci peut permettre à n'importe quelle utilisateur de réaliser des actions administratives	Pour arranger ce problème, il faut créer un utilisateur par défaut comme postgresql ou zookeeper.
10	website/include/pictures.php	Moyenne	Cryptographie faible	L'utilisation des chiffres pseudo-aléatoires sensibles à la sécurité. Quand un logiciel génère des valeurs prévisibles, il est possible que l'attaquant puisse deviner la prochaine valeur. Cela peut permettre d'imiter un autre utilisateur.	L'utilisation des fonctions qui utilise des générateurs qui produisent des chiffres random cryptographique.

2.2.1. Description des vulnérabilités

Selon SonarCloud, les vulnérabilités de OWASP Top 10 sont **Injection**, **Cryptographic Failures**, **Cross-site-Scripting** (XSS), **Security Misconfiguration** et **Broken Access Control**. Les autres du tableau 2 sont des vulnérabilités de **Sonar Source** ou ils sont des **Security Hotspots**.

2.2.1.1. Cryptographie Faible

Cette vulnérabilité se produit quand l'application utilise des algorithmes de chiffrement qui ne sont plus considérés assez sécurisés pour protéger les données. Plusieurs attaques peuvent exploiter cette vulnérabilité comme des attaques force brute pour essayer de déchiffrer les messages chiffrés.

2.2.1.2. Permission

Cette vulnérabilité est due au fait que sans préciser les fichiers et les dossiers il est possible que certains fichiers avec des excès de permissions peuvent être copiés dans le conteneur et des attaques peuvent exploiter ces permissions pour avoir accès à des données sensibles.

2.2.1.3. Injection SQL

Une vulnérabilité d'injection SQL se produit quand les entrées utilisateurs ne sont pas validées ou filtrées avant d'être incluses dans des requêtes SQL envoyées à la base de données. Les attaquants peuvent prendre avantage de cette vulnérabilité en envoyant des données malveillantes qui peuvent modifier ou supprimer des données.

2.2.1.4. Injection de Commande

Une vulnérabilité d'injection de commande produit lorsqu'un utilisateur peut injecter des commandes système malveillantes dans une application. Ils sont ensuite exécutés par le système d'exploitation sans vérification adéquate. Les attaquants peuvent prendre contrôle du système ou l'application.

2.2.1.5. Cross-Site Scripting

La vulnérabilité de Cross-Site Scripting (XSS) se produit lorsque des données entrantes ne sont pas validées. Les attaquants peuvent exploiter cette vulnérabilité en injectant du code JavaScript malveillant dans une page Web. Ce code sera exécuté par le navigateur web.

2.2.1.6. Configuration non-sécurisée

La vulnérabilité de configuration non sécurisée se produit lorsque le système n'est pas bien configuré. Alors, il n'est pas sécuritaire. Par exemple, les permissions d'accès aux fichiers seront affectées.

2.2.1.7. Injection de traversée de chemin

La vulnérabilité d'injection de traversée de chemin se produit lorsqu'un attaquant peut exploiter une application pour ajouter un fichier sur le système de fichiers distant en mettant un chemin de fichier malveillant dans un champ d'entrée d'une application web.

2.3. Test d'intrusions avec ZAP

Figure 4 : Résultats des tests d'intrusions avec ZAP de WackoPicko

Alert type	Risk	Count
Absence of Anti-CSRF Tokens	Medium	34 (261.5%)
Content Security Policy (CSP) Header Not Set	Medium	31 (238.5%)
Missing Anti-clickjacking Header	Medium	26 (200.0%)
Cookie No HttpOnly Flag	Low	2 (15.4%)
Cookie without SameSite Attribute	Low	2 (15.4%)
Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)	Low	43 (330.8%)
Server Leaks Version Information via "Server" HTTP Response Header Field	Low	69 (530.8%)
Timestamp Disclosure - Unix	Low	5 (38.5%)
X-Content-Type-Options Header Missing	Low	47 (361.5%)
Content-Type Header Missing	Informational	2 (15.4%)
Information Disclosure - Sensitive Information in URL	Informational	5 (38.5%)
Loosely Scoped Cookie	Informational	3 (23.1%)
User Controllable HTML Element Attribute (Potential XSS)	Informational	5 (38.5%)
Total		13

2.3.1. Description des vulnérabilités

Cette section va décrire les différentes vulnérabilités que ZAP à détecter lors des tests d'intrusions. WackoPicko présente 13 vulnérabilités, comme présenté à la figure 4, que nous allons décrire et rapporter selon les classements OWASP Top 10 en 2021, 2017 et 2013.

2.3.1.1. Absence of Anti-CSRF Tokens

Cette vulnérabilité détectée par ZAP est de risque moyen. Les tests de pénétration ont détectés 34 instances. Selon la documentation ZAP, cette vulnérabilité signifie que l'application WackoPicko ne dispose pas de jetons anti-CSRF pour protéger les utilisateurs contre les attaques de falsification de requêtes inter sites (CSRF). Pour remédier à la situation, une solution serait, durant la phase d'architecture et de conception, d'utiliser une bibliothèque ou un cadriciel qui prévient cette vulnérabilité, par exemple en utilisant des « packages » anti-CSRF tels que l'OWASP CSRFGuard. En considérant le risque de cette vulnérabilités et qu'elle est classée A01 - Broken Access Control du classement OWASP Top 10 en 2021 (A05 en 2017), il serait alors important que les développeurs de l'application appliquent les actions correctives. [14]

2.3.1.2. Content Security Policy (CSP) Header Not Set

Cette vulnérabilité détectée par ZAP est de risque moyen. L'analyse de l'application avec ZAP a détecté 31 instances de cette vulnérabilité. Cette vulnérabilité indique que l'application WackoPicko n'a pas de politique de sécurité de contenu (CSP) définie dans les en-têtes HTTP, ce qui peut permettre à un attaquant de charger et d'exécuter du contenu malveillant sur le site. Pour éliminer cette vulnérabilité, les développeurs pourraient s'assurer de configurer l'en-tête Content-Security-Policy dans les requêtes HTTP. En considérant le risque de cette vulnérabilités et qu'elle est classée dans A05 - Security Misconfiguration du classement OWASP Top 10 en 2021 (A06 en 2017 et A05 en 2013), il serait alors important que les développeurs de l'application appliquent les actions correctives, sans toutefois être une priorité élevée. [7]

2.3.1.3. Missing Anti-clickjacking Header

Cette vulnérabilité détectée par ZAP est de risque moyen. Selon les tests avec ZAP, cette vulnérabilité est apparue à 26 occasions. Cette vulnérabilité signifie que l'application WackoPicko ne dispose pas d'en-têtes de protection contre les attaques de « clickjacking », qui peuvent tromper les utilisateurs pour qu'ils cliquent sur des éléments malveillants ou indésirables. ZAP propose une solution qui consiste à configurer les en-têtes Content-Security-Policy et X-Frame-Options HTTP. En considérant le niveau de risque de cette vulnérabilité et que cette dernière est classée A05 - Security Misconfiguration du classement OWASP Top 10 en 2021 (A06 en 2017 et A05 en 2013), il serait alors pertinent que les développeurs de l'application appliquent les actions correctives requises, sans toutefois en faire une priorité. [8]

2.3.1.4. Cookie No HttpOnly Flag

Cette vulnérabilité détectée par ZAP est de faible risque. Selon les tests avec ZAP, cette vulnérabilité est apparue à seulement 2 occasions. Cette vulnérabilité indique que les cookies de l'application WackoPicko ne sont pas marqués avec le drapeau HttpOnly, ce qui peut permettre à un attaquant de voler des cookies et de voler des informations de session, par un script JavaScript notamment. Une solution proposée par ZAP consiste à s'assurer que le flag est correctement assigné au moment de la création du cookie. En considérant le niveau de risque de cette vulnérabilité et que cette dernière est classée A05 - Security Misconfiguration du classement OWASP Top 10 en 2021 (A06 en 2017 et A05 en 2013), ce n'est pas alors une priorité pour les développeurs d'apporter immédiatement les actions correctives. [15]

2.3.1.5. Cookie without SameSite Attribute

Cette vulnérabilité détectée par ZAP est un risque de faible niveau. Selon les tests avec ZAP, cette vulnérabilité est apparue à seulement 2 occasions. Cette vulnérabilité signifie que les cookies de l'application WackoPicko n'ont pas l'attribut SameSite défini, ce qui peut permettre aux cookies d'être exploités dans des attaques de vol de session telles que le détournement de clic. L'action corrective à apporter au code serait de s'assurer que l'attribut SameSite est défini sur « lax » ou « strict » pour tous les cookies. En considérant le niveau de risque de cette vulnérabilité et que cette dernière est classée A01 - Broken Access Control du classement OWASP Top 10 en 2021 (A05 en 2017), il serait important que l'action corrective proposée, ou une autre, soit appliquée sans trop de délai, afin que le niveau de risque de cette vulnérabilité n'augmente pas. [16]

2.3.1.6. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Cette vulnérabilité détectée par ZAP est de faible risque. Selon les tests avec ZAP, cette vulnérabilité est apparue à 43 occasions. Cette vulnérabilité indique que l'application WackoPicko divulgue des informations sur les technologies utilisées sur le serveur via l'en-tête HTTP X-Powered-By, ce qui peut permettre à un attaquant de cibler des vulnérabilités connues dans ces technologies. Une solution qui pourrait régler cette vulnérabilité serait de s'assurer que le serveur soit configuré pour supprimer les en-têtes « X-Powered-By ». Cette vulnérabilité est classée A01 - Broken Access Control du classement OWASP Top 10 en 2021 et A03 - Sensitive Data Exposure dans le classement de 2017. Étant donné que la vulnérabilité est apparue à 43 occasions et en considérant le niveau de risque, il faudrait que la prochaine priorité des développeurs soit de corriger cette vulnérabilité. [17]

2.3.1.7. Server Leaks Version Information via "Server" HTTP Response Header Field

Cette vulnérabilité détectée par ZAP est un risque de faible niveau. Selon les tests avec ZAP, cette vulnérabilité est apparue à 69 occasions. Cette vulnérabilité signifie que l'application WackoPicko divulgue des informations sur la version du serveur via l'en-tête HTTP Server, ce qui peut permettre à un attaquant de cibler des vulnérabilités connues dans cette version. Une solution qui pourrait régler cette vulnérabilité serait de s'assurer que le serveur soit configuré pour supprimer les en-têtes « Server ». En considérant le niveau de risque de cette vulnérabilité, le nombre de fois qu'elle a été détectée et que cette dernière est classée A05 - Security Misconfiguration du classement OWASP Top 10 en 2021 (A06 en 2017 et A05 en 2013), il faudrait que la prochaine priorité des développeurs soit de corriger cette vulnérabilité. [18]

2.3.1.8. Timestamp Disclosure - Unix

Cette vulnérabilité détectée par ZAP est de faible risque. Selon les tests avec ZAP, cette vulnérabilité est apparue à seulement 5 occasions. Cette vulnérabilité indique que l'application WackoPicko divulgue des informations sur l'horodatage (timestamp) Unix via des messages d'erreur, ce qui peut aider un attaquant à mieux comprendre le fonctionnement de l'application. Pour contrer cette vulnérabilité, une solution serait de s'assurer que les informations de timestamp ne sont pas sensibles et ne peuvent pas être agrégées. Cette vulnérabilité est classée A01 - Broken Access Control du classement OWASP Top 10 en 2021 et A03 - Sensitive Data Exposure dans le classement de 2017. Étant donné du niveau de risque de la vulnérabilité, il n'est donc pas une priorité pour les développeurs d'apporter immédiatement des actions correctives. [19]

2.3.1.9. X-Content-Type-Options Header Missing

Cette vulnérabilité détectée par ZAP est un risque de faible niveau. Selon les tests avec ZAP, cette vulnérabilité est apparue à 47 occasions. Cette vulnérabilité indique que l'application WackoPicko ne dispose pas de l'en-tête X-Content-Type-Options défini, ce qui peut permettre à un attaquant de tromper le navigateur pour qu'il exécute du contenu malveillant. La solution serait donc de s'assurer que le Content-Type soit bien défini dans les en-têtes des requêtes. En considérant le niveau de risque de la vulnérabilité, du nombre de fois qu'elle a été rencontrée et le fait qu'elle soit classée A05 - Security Misconfiguration du classement OWASP Top 10 en 2021 (A06 en 2017 et A05 en 2013), il serait donc important que la prochaine priorité des développeurs soit de régler cette faille de sécurité. [11]

2.3.1.10. Content-Type Header Missing

Cette vulnérabilité détectée par ZAP est un risque de niveau informationnel. Cela signifie que cette dernière n'est pas considérée comme une vulnérabilité ou une menace en soi, mais plutôt comme un sujet de préoccupation ou une faiblesse potentielle qui peut ne pas avoir d'impact immédiat sur la sécurité, mais qui le pourrait dans le futur. Selon les tests avec ZAP, cette

vulnérabilité est apparue seulement 2 fois lors de l'attaque. Cette vulnérabilité indique que l'application WackoPicko ne dispose pas l'en-tête Content-Type, ce qui peut permettre à un attaquant de tromper le navigateur pour qu'il exécute du contenu malveillant. La solution proposée par ZAP serait de s'assurer que l'en-tête est bien défini. En considérant le niveau de risque de cette vulnérabilité et que cette dernière est classée A05 - Security Misconfiguration du classement OWASP Top 10 en 2021 (A06 en 2017 et A05 en 2013), il n'est pas nécessaire d'apporter l'action corrective proposée, mais plutôt d'y porter attention afin d'analyser la cause qui mène à cette vulnérabilité afin de s'assurer que ce risque ne devient pas plus important dans le futur. [20]

2.3.1.11. Information Disclosure - Sensitive Information in URL

Cette vulnérabilité détectée par ZAP présente un risque de niveau informationnel. Cette faiblesse n'est pas une menace en soit dans le cas de l'application WackoPicko, car le risque est informationnel, tel qu' expliqué dans la vulnérabilité précédente. Selon les tests avec ZAP, cette vulnérabilité est apparue à seulement 5 occasions. Cette vulnérabilité signifie que l'application WackoPicko transmet des informations sensibles dans l'URL, telles que les identifiants d'utilisateur ou les mots de passe, ce qui peut permettre à un attaquant de récupérer ces informations en surveillant le trafic réseau ou en accédant aux journaux du serveur. La solution serait de ne pas passer les données sensibles dans les URL. Cette vulnérabilité est classée A01 - Broken Access Control du classement OWASP Top 10 en 2021 et A03 - Sensitive Data Exposure dans le classement de 2017. Bien que le risque est informationnel et que la vulnérabilité n'est pas une menace actuelle, il faut toutefois que les développeurs portent très attention à celle-ci, car elle peut rapidement devenir une menace sérieuse. [21]

2.3.1.12. Loosely Scoped Cookie

Cette vulnérabilité détectée par ZAP présente un risque de niveau informationnel. Suite aux tests de pénétration avec ZAP, cette vulnérabilité est apparue à seulement 3 occasions. Cette vulnérabilité indique que les cookies de l'application WackoPicko sont trop largement définis en termes de portée, ce qui peut permettre à un attaquant d'accéder à des informations de session de différents utilisateurs. La solution proposée pour corriger la vulnérabilité serait de limiter les cookies à un nom de domaine complètement qualifié (FQDN). Cette vulnérabilité est classée A08 - Software and Data Integrity Failures du classement OWASP Top 10 en 2021 (A06 en 2017). En considérant le niveau de risque de celle-ci et du classement OWASP, il n'est pas une priorité pour les développeurs d'investiguer sur la cause de cette dernière. [22]

2.3.1.13. User Controllable HTML Element Attribute (Potential XSS)

Cette vulnérabilité détectée par ZAP présente un risque de niveau informationnel. Selon les tests avec ZAP, cette vulnérabilité est apparue à seulement 5 occasions. Cette vulnérabilité indique que les éléments HTML de l'application WackoPicko sont vulnérables aux attaques de cross-site scripting (XSS) en raison de la présence d'attributs contrôlables par l'utilisateur. Cela peut

permettre à un attaquant d'exécuter du code malveillant dans le navigateur d'un utilisateur. ZAP propose une solution qui consiste à valider toutes les entrées et sorties avant d'écrire dans des attributs HTML. Cette vulnérabilité est classée A03 - Injection du classement OWASP Top 10 en 2021 (A01 en 2017 et 2013). En considérant le niveau de risque de celle-ci et du classement OWASP, il serait nécessaire que les développeurs s'assurent de prendre connaissance de cette vulnérabilité et de prévoir un moment pour apporter les corrections nécessaires. [23]

2.4. Comparaisons des résultats de SonarCloud et ZAP

En analysant les résultats des tests, nous pouvons remarquer que certaines vulnérabilités détectées par SoncarCloud ont aussi été détectées par ZAP. Les types de vulnérabilités détectées par SonarCloud sont : cryptographie faible, injection SQL, injection de commande, Cross-Site Scripting, configuration non-sécurisée, permission et injection de traversée de chemin. Ces vulnérabilités se rapporte respectivement aux catégories du classement OWASP Top 10 en 2021 A02 - Cryptographic Failures pour cryptographie faible, A03 - Injection pour injection SQL et injection de commande, A05 - Security Misconfiguration pour configuration non-sécurisée, A01 - Broken Access Control pour permission et injection de traversée de chemin et enfin A07 - Cross-Site Scripting (XSS) dans le classement de 2017 pour Cross-Site Scripting. En comparaison avec les tests faits par ZAP, la vulnérabilité d'injection SQL est l'équivalent de User Controllable HTML Element Attribute (Potential XSS). Pour cette vulnérabilité commune détectée par les deux outils de tests de sécurité, SonarCloud a détecté que la vulnérabilité est à un niveau de sévérité critique, c'est-à-dire le niveau le plus élevé, tandis que ZAP a détecté que le niveau de risque n'est qu'informationnel. La différence entre les résultats des tests peut s'expliquer par le fait que les deux outils de test n'utilisent pas la même approche. SonarCloud se concentre sur l'analyse statique du code source, tandis que ZAP utilise une approche d'analyse dynamique en testant les applications en cours d'exécution. Cela peut aussi expliquer que SonarCloud ne détecte pas certaines vulnérabilités, telles que les vulnérabilités Missing Anti-clickjacking Header et Loosely Scoped Cookie.

3. Références

1. https://portswigger.net/web-security/os-command-injection#:~:text=OS%20command%20injection%20(also%20known,application%20and%20all%20its%20data
2. yogisec. (s.d.). VulnerableSAMLApp. Github. https://github.com/yogisec/VulnerableSAMLApp
3. adamdoupe. (s.d.). WackoPicko. Github. https://github.com/adamdoupe/WackoPicko
4. OWASP. (s.d.). Welcome to the OWASP Top 10 - 2021. https://owasp.org/Top10/
5. OWASP. (s.d.). OWASP Top Ten 2017. https://owasp.org/www-project-top-ten/2017/Top_10
6. OWASP. (s.d.). OWASP Top 10 - 2013. https://owasp.org/www-pdf-archive/OWASP_Top_10_-_2013.pdf
7. ZAP. (s.d.). Content Security Policy (CSP) Header Not Set. https://www.zaproxy.org/docs/alerts/10038-1/
8. ZAP. (s.d.). Anti-clickjacking Header. https://www.zaproxy.org/docs/alerts/10020/
9. ZAP. (s.d.). Vulnerable JS Library. https://www.zaproxy.org/docs/alerts/10003/
10. ZAP. (s.d.).Big Redirect Detected (Potential Sensitive Information Leak). https://www.zaproxy.org/docs/alerts/10044/
11. ZAP. (s.d.). X-Content-Type-Options Header Missing, https://www.zaproxy.org/docs/alerts/10021/
12. ZAP. (s.d.). Information Disclosure - Suspicious Comments. https://www.zaproxy.org/docs/alerts/10027/
13. ZAP. (s.d.). User Agent Fuzzer. https://www.zaproxy.org/docs/alerts/10104/
14. ZAP. (s.d.). Absence of Anti-CSRF Tokens. https://www.zaproxy.org/docs/alerts/10202/
15. ZAP. (s.d.). Cookie No HttpOnly Flag. https://www.zaproxy.org/docs/alerts/10010/
16. ZAP. (s.d.). Cookie without SameSite Attribute. https://www.zaproxy.org/docs/alerts/10054/
17. ZAP. (s.d.). Server Leaks Information via 'X-Powered-By' HTTP Response Header Field(s). https://www.zaproxy.org/docs/alerts/10037/
18. ZAP. (s.d.). Server Leaks Version Information via 'Server' HTTP Response Header Field. https://www.zaproxy.org/docs/alerts/10036-2/
19. ZAP. (s.d.). Timestamp Disclosure. https://www.zaproxy.org/docs/alerts/10096/
20. ZAP. (s.d.). Content-Type Header Missing. https://www.zaproxy.org/docs/alerts/10019/
21. ZAP. (s.d.).Information Disclosure - Sensitive Information in URL. https://www.zaproxy.org/docs/alerts/10024/
22. ZAP. (s.d.). Loosely Scoped Cookie. https://www.zaproxy.org/docs/alerts/90033/

23. ZAP. (s.d.). User Controllable HTML Element Attribute (Potential XSS). https://www.zaproxy.org/docs/alerts/10031/