|  |
| --- |
| Rapport de pentest |
| Vulnerability Report |
| Sunday, June 30, 2024 |

Table of Contents

[1. Introduction 3](#_Toc177396537)

[1.1. Contexte et objectifs de la mission 3](#_Toc177396538)

[1.2. Périmètre de la mission 3](#_Toc177396539)

[1.3. Méthodologie 3](#_Toc177396540)

[1.4. Déroulement des travaux 3](#_Toc177396541)

[1.5. Limites 4](#_Toc177396542)

[1.6. Remise en état 4](#_Toc177396543)

[2. Summary 5](#_Toc177396544)

[2.1. Executive summary 5](#_Toc177396545)

[2.1. Vulnérabilités et recommandations 6](#_Toc177396546)

[3. Constats détaillés 7](#_Toc177396547)

# Introduction

## Contexte et objectifs de la mission

IBC ask to carry out undefined. The objectives of these tests were to:

* Identify vulnerabilities in the web application and the mobile application (iOS for iPad);
* Identify the risks associated with the identified vulnerabilities;
* To propose prioritized corrective actions;
* Formalize these elements in a clear and understandable report for technical and business contacts.

## Périmètre de la mission

IBC has mandated us to perform security tests on the following scope:

* www.secure-bank.com
* www.sudo.co.

## Méthodologie

The engagement was based the following approaches:

* Phase 1: Information gathering & target mapping;
* Phase 2: Vulnerability scan & attack plan identification;
* Phase 3: Exploitation of identified vulnerabilities & elevation of privileges;
* Phase 4: Consolidation & results restitution.

## Déroulement des travaux

The testing activities were performed between 06/18/2024 and 07/02/2024.

## Limites

## Remise en état

# Summary

## Executive summary

## Vulnérabilités et recommandations

Following vulnerabilities have been discovered:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Vulnérabilités | Gravité | Recommandations | Priorité |
| V007 | (generic) SQL Injection | Critical | SQLi |  |
| V003 | XSS | High | XSS |  |
| V008 | Relative Path Traversal | High |  | 4 |

# Constats détaillés

The following details the vulnerabilities identified in the scope, together with an estimate of the impact of the exploitation of the vulnerability, the ease of exploitation of the vulnerability and its criticality. Also referenced are the business risks associated with the vulnerability in the context of your environment, as well as the recommendation for remediating the vulnerability and reducing the risks.

|  |  |  |
| --- | --- | --- |
| V007 | (generic) SQL Injection | Gravité |
| Critical |
| Libellé | Grâce à l'injection de code SQL (SQLi), il est possible de modifier la logique des instructions SQL et de permettre l'exécution de différents types de requêtes de base de données. En plus d'ajouter de nouveaux enregistrements ou de supprimer/modifier des enregistrements existants, il est possible d'obtenir des données de la base de données qui ne devraient, en théorie, pas être accessibles, telles que les utilisateurs, les mots de passe, les détails sur l'application elle-même, etc. Si l'utilisateur de la base de données dispose de plus de droits que nécessaire, il peut être en mesure d'accéder à d'autres bases de données ou même d'exécuter des commandes système. La principale cause de l'injection de code SQL (SQLi) est une validation incorrecte des paramètres d'entrée de l'application. | |
| Périmètre |  | |

|  |  |  |
| --- | --- | --- |
| Recommandation | | |
| E007 | SQLi | Priorité |
|  |
| Libellé | Validez les paramètres gérés par l'application, notamment lorsque l'utilisateur les soumet en tant que données d'entrée. Cette validation est toujours effectuée côté serveur et ne dépend jamais des validations côté client. | |

Injection de code Sql dans le champ de connexion :

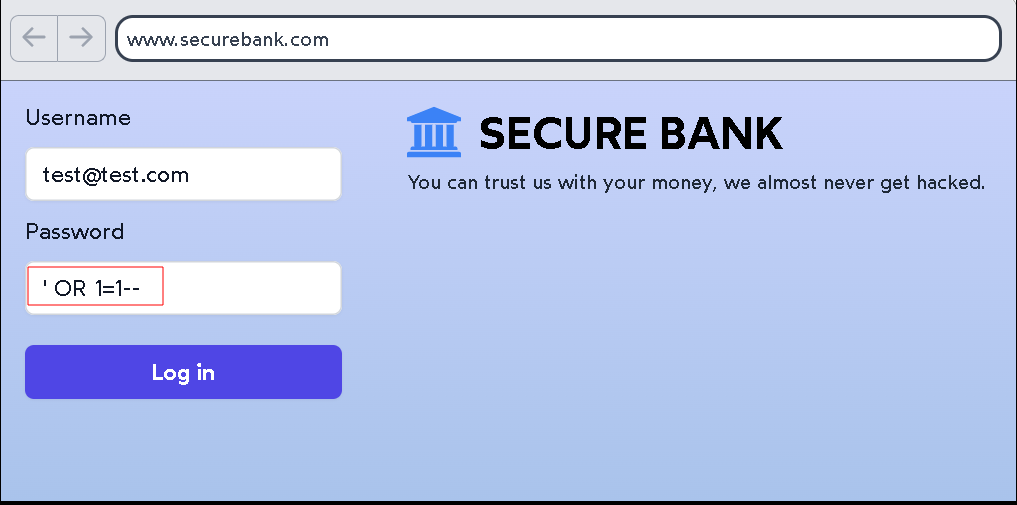


Image 1 –

Execution de code Sql reussie :

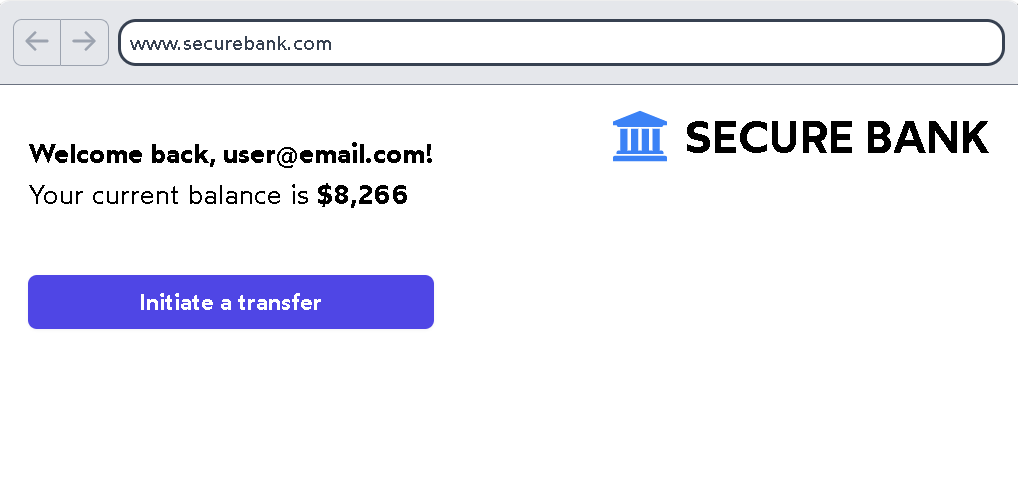


Image 1 –

|  |  |  |
| --- | --- | --- |
| V003 | XSS | Gravité |
| High |
| Libellé | La vulnérabilité « Cross-Site Scripting (XSS) » permet à des tiers d'injecter du code Java Script ou HTML dans des pages Web d'applications spécifiques. Ce faisant, il est possible de voler des sessions utilisateur, de compromettre la sécurité du système et d'obtenir des informations sensibles ou exclusives de l'application. Cette vulnérabilité est généralement provoquée par l'incapacité d'une application à valider les paramètres d'entrée et de sortie, ce qui se produit fréquemment avec les formulaires. | |
| Périmètre |  | |

|  |  |  |
| --- | --- | --- |
| Recommandation | | |
| E003 | XSS | Priorité |
|  |
| Libellé | Chaque paramètre géré par l'application doit être validé, notamment lorsqu'il s'agit de données d'entrée fournies par l'utilisateur. Cette validation est toujours effectuée côté serveur et ne repose JAMAIS sur des validations effectuées côté client. | |

Injection d'un code javascript dans un champ d'abonnement :

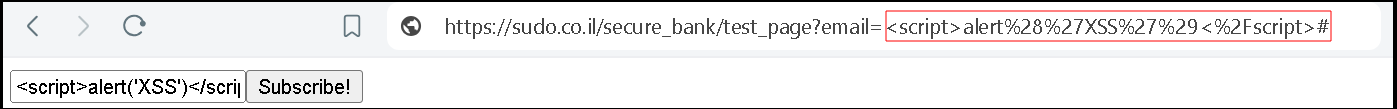


Image 1 –

Execution du code Xss reussie :

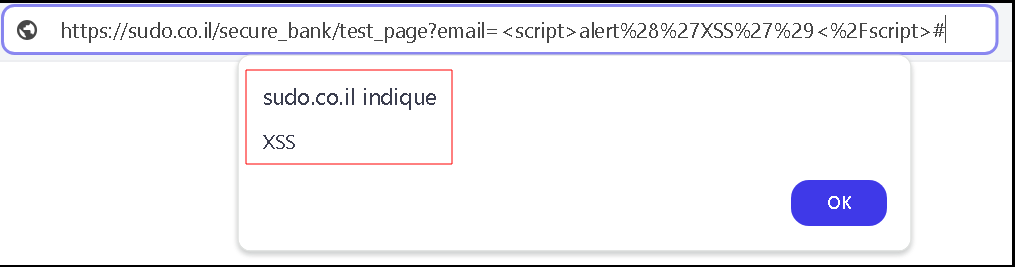


Image 1 –

|  |  |  |
| --- | --- | --- |
| V008 | Relative Path Traversal | Gravité |
| High |
| Libellé | Path Traversal est autorisé. Il est possible d'obtenir n'importe quel fichier du serveur distant. | |
| Périmètre |  | |

|  |  |  |
| --- | --- | --- |
| Recommandation | | |
| E008 |  | Priorité |
| 4 |
| Libellé | Partez du principe que toutes les entrées sont malveillantes. Utilisez une stratégie de validation des entrées « accepter les entrées connues comme bonnes », c'est-à-dire utilisez une liste blanche d'entrées acceptables strictement conformes aux spécifications. Rejetez toute entrée qui n'est pas strictement conforme aux spécifications ou transformez-la en quelque chose qui l'est. Lors de la validation des entrées, tenez compte de toutes les propriétés potentiellement pertinentes, notamment la longueur, le type d'entrée, la plage complète de valeurs acceptables, les entrées manquantes ou supplémentaires, la syntaxe, la cohérence entre les champs associés et la conformité aux règles métier. À titre d'exemple de logique de règle métier, « boat » peut être syntaxiquement valide car il ne contient que des caractères alphanumériques, mais il n'est pas valide si l'entrée ne doit contenir que des couleurs telles que « rouge » ou « bleu ». Ne vous fiez pas exclusivement à la recherche d'entrées malveillantes ou malformées (c'est-à-dire, ne vous fiez pas à une liste noire). Une liste noire est susceptible de manquer au moins une entrée indésirable, en particulier si l'environnement du code change. Cela peut donner aux attaquants suffisamment de marge pour contourner la validation prévue. Cependant, les listes noires peuvent être utiles pour détecter des attaques potentielles ou déterminer quelles entrées sont si mal formées qu'elles doivent être rejetées d'emblée. Lors de la validation des noms de fichiers, utilisez des listes blanches strictes qui limitent le jeu de caractères à utiliser. Si possible, n'autorisez qu'un seul caractère « . » dans le nom de fichier pour éviter les faiblesses telles que CWE-23, et excluez les séparateurs de répertoire tels que « / » pour éviter CWE-36. Utilisez une liste blanche d'extensions de fichiers autorisées, ce qui permettra d'éviter CWE-434. Ne vous fiez pas exclusivement à un mécanisme de filtrage qui supprime les caractères potentiellement dangereux. Cela équivaut à une liste noire, qui peut être incomplète (CWE-184). Par exemple, le filtrage de « / » constitue une protection insuffisante si le système de fichiers prend également en charge l'utilisation de « » comme séparateur de répertoire. Une autre erreur possible peut se produire lorsque le filtrage est appliqué d'une manière qui produit toujours des données dangereuses (CWE-182). Par exemple, si les séquences '../' sont supprimées de la chaîne '.../...//' de manière séquentielle, deux instances de '../' seraient supprimées de la chaîne d'origine, mais les caractères restants formeraient toujours la chaîne '../'. | |

Recuperation du fichier /etc/shadow

image

Image 1 –

Fichier shadow récupéré :



Image 1 –

utilisateur Secure\_Admin et son hash :

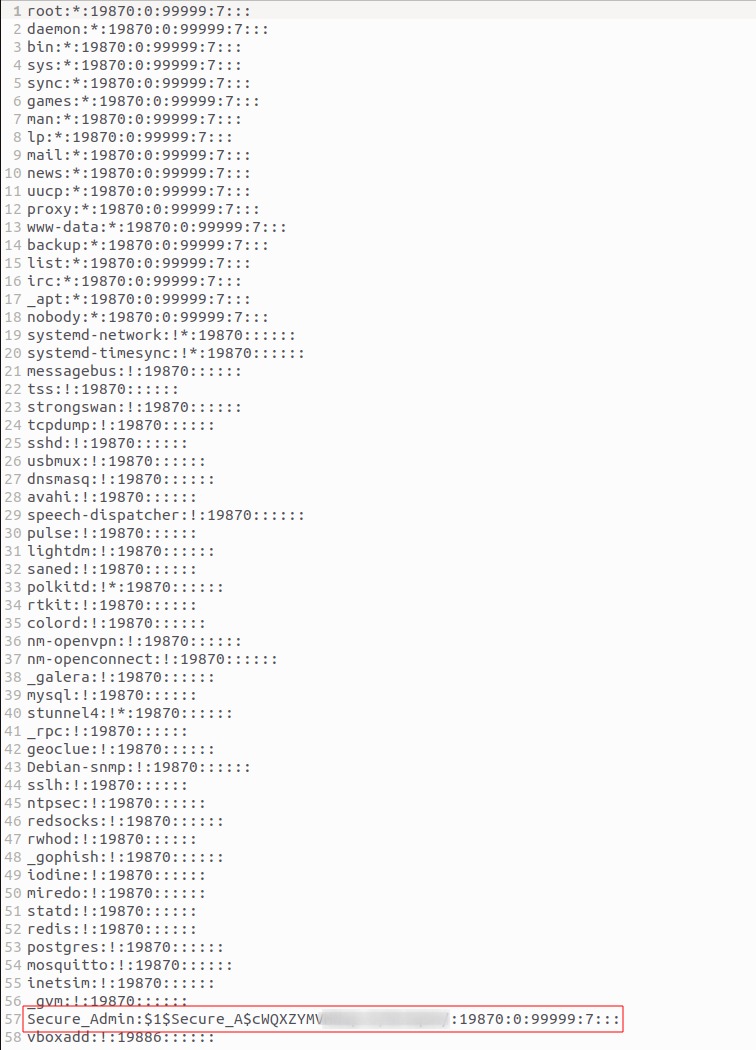


Image 1 – shadow.png