KASINTU

Web Collection Game

Security Report

Semester 3 - Individual Project

Airell Rasendriya Bachtiar
airell.bachtiar@student.fontys.nl

Table of Contents

1. Version

Version	Date	Description
0.1	03-06-2022	Security Risk Report first draft
0.2	10-06-2022	Fixed security table
1.0	16-06-2022	Version 1.0 of security report

2. Introduction

2.1. Purpose

The purpose of this document is to review the security measure using top 10 OWASP most common security risks.

2.2. Definitions, Acronyms, and Abbreviations

- Gacha: A method inspired by toy vending machine where you can get a toy randomly from what the vending machine provide. Instead of toy vending machine, here it is turned into an application game where you can get an item, in this project we called a creature, randomly with a set number of chances.
- Summon or Pull: The action performed when you are getting a creature from the gacha.
- Banner: The place where you summon or pull creature. Banner contains a list of creatures in which the player can obtained and a chance or percentage of how many chances you can obtain a specific creature.

3. System Overview

3.1. Description

This game is called Kasintu which means bird. Kasintu is a collection-based game where player can collect as much as they want. What they will collect is a different type of birds that is real and fictional thus the meaning of Kasintu is bird, a game where you collect birds. From now on these birds will be called creature.

3.2. Main User Activities

The main feature of this game is called a gacha system. Gacha system is where player can get a chance to receive a virtual item using in game currency. This is where player mainly get a new creature that will be release or has been released by the developer. They called this action of obtaining new creature as a summoning or pulling. In this case we will call this action as summon or summoning. As the where they summoning these creatures is called a banner. A banner contains a certain amount or all the creature available that can be obtained by the player who summoned on that banner.

For a future feature, Kasintu will also include a marketplace and breeding system. Marketplace is where player can but, sell or trade creatures from the other player. Breeding system is where player can breed their own creature to make new creature which may become rarer that the previous creature.

3.3. Project Goal

The goal of this project is to have a game that will entertain user by collecting creatures and to collect everything the game provides. For better user experience, this game will have to has a fast user interface to make user does not need to wait long in between action or input and a secure application so that data from user cannot be tracked or stolen by a third party.

4. Security Risk

This data is taken from the top 10 OWASP most common security risks. (Stock, Glas, Smithline, & Gigler, 2021)

Type	Likelihood	Impact	Risk	Action	Planned
A01: 2021-Broken Access Control	High	Severe	High	Implements access control mechanisms	No, out of scope
A02: 2021-Cryptographicc Failures	Moderate	Moderate	Moderate	Sensitive data is only stored in database and does not pass around or stored anywhere in the system	Yes
A03: 2021-Injection	Low	Severe	Moderate	Parameterized queries, JPA criteria API, and user data sanitization	No
A04: 2021-Insecure Design	Low	Mild	Low	Unit and integration tests	Yes
A05: 2021-Security Misconfiguration	High	Severe	High	Check all the system configuration settings	No, out of scope
A06: 2021-Vulnerable and Outdated Components	Low	Mild	Low	N/A	Yes
A07: 2021-Identification and Authentication Failures	High	Severe	High	Implements multi-factor authentication and check password strength	Yes
A08: 2021-Software and Data Integrity Failures	Low	Severe	Moderate	Keep application private, changing code can only be done by the owner	Yes
A09: 2021-Security Logging and Monitoring Failures	Moderate	Moderate	Moderate	Log every important action done by the user	No
A10: 2021-Server-Side Request Forgery	Low	Severe	Moderate	N/A	No

Broken Access Control is a risk where attacker can access API with no authorization. It can be solved by implementing access control mechanisms such as Cross-Origin Resource Sharing (CORS). CORS is a HTTP-header based mechanism that allows a server to indicate any origins other than its own from which browser should permit loading resources. (contributors, 2022) This action will not be performed due to time constraint in the project thus it is out of scope.

Cryptographic failures are a data that is sensitive and needs protection that is not encrypted so that attackers can access those sensitive data. The first action needed to be done is encrypted the data by using cryptographic algorithm such as SHA256. To add

more layer of protection, sensitive data are not passed around redundantly in web application and not saved anywhere other than the database.

Injection is an attack that injects SQL statement and data can be leaked or changed by the statement. To prevent it we can perform multiple action, parameterized queries, JPA criteria API, and user data sanitization. (Sevestre, 2022) Parameterized queries are a method where the SQL statement for execution is prepared with the question mark placeholder whenever user-supplied value is needed. JPA criteria API makes creating complex statement/query services more straightforward and safer. User data sanitization is a technique of applying a filter to user-supplied data so it can be safely used by other part of the application. These methods can be applied when application has an SQL statement. Fortunately, the application doesn't have any SQL statement so the injection prevention will not be applied.

Security design is a flaw in the design of the application and making the flawed design can be exploited. The action that can be taken is making unit and integration tests. This test will be added in the application.

Security configuration can happen when configuration in the application is used incorrectly and unnecessary features that are not used are installed. From there attackers can abuse the misconfiguration of the security to access the application. To prevent this, security configuration settings can be checked and verified if that is being used. It is out of scope in this project so any risk will be accepted.

Vulnerable and outdated components is a risk where components that are included in the application is outdated/old, or the version are not known. This can be prevented by checking every component and verify the version and date. This risk has been taken into account from the beginning and every component used in the project are verified, trusted, and not outdated.

Identification and authentication failures is a failure where the user's identity, authentication, and session manager are not fully protected. This can be prevented by having a multi factored authentication and check for password strength.

Software and data integrity failures relate to code and infrastructure that does not protect against integrity violation. This include, having a library or components from untrusted source and application can be access/change/updated by random sources. It can be prevented by making the application private and verify that library and component is trusted.

Security logging and monitoring failures is a failure in logging the actions taken from the users. By logging the action performed by the users, it can help detect, escalate, and respond to active breaches. This has been partly implemented by the system when application is accessing the database.

Server-side request forgery flaws occur whenever the web application is fetching remote resource without validating the user-supplied URL and make attackers can send a crafted request. This can be easily handled by enforcing "deny by default" firewall

policies or network access control rules to block all but essential intranet traffic. This risk will be accepted because it is out of scope for this project.

5. Conclusion

Most of the risks listed are not planned due to time constraint for the project and it is not planned for a public access web application and used by hundreds of users. Therefore, an attack to the web application is low and any attack will be accepted because it doesn't contain many sensitive data and saved data are only related to the application.

6. References

contributors, M. (2022, 05 21). Retrieved from MDN web docs: https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

Sevestre, P. (2022, 05 21). Retrieved from Baeldung: https://www.baeldung.com/sql-injection

Stock, A. v., Glas, B., Smithline, N., & Gigler, T. (2021, 09 24). Retrieved from OWASP: https://owasp.org/Top10