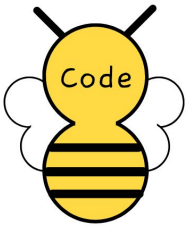
Risk Analysis

CodeBee



Hongqing Cao, 400053625
Jason Nagy, 400055130
Shivaansh Prasann, 400100351
Sida Wang, 400072157
Weidong Yang, 400065354
Yachen Wu, 400131465

1. Introduction

1.1 Purpose

The purpose of this document is to analyze the various subsystems of CodeBee and their potential risks and examine how to either reduce or eliminate these risks. The goal is to identify solutions to reduce risks to an acceptable level based on the risk severity and likelihood. The risk assessment shall be performed at each different stage of the software development life cycle to ensure that the robustness and security of our application is ensured. Different aspects of the potential risks will be considered including Vulnerability and Threat.

1.2. Scope of risk analysis

This risk analysis highlights the different vulnerabilities that can possibly affect the system as a whole, as well as threats and their likelihoods which may exploit system vulnerabilities. We shall not address issues that might arise by user error, such as erroneous programs, or problems stemming from the user's infrastructure, such as local system power loss. We consider all the scenarios which could affect the application as well as the supporting infrastructure.

The web application is implemented to achieve the purpose of teaching students aged 10-16 how to program throughout playing enjoyable programming-related computer games. The user will be able to overcome plenty of games with corresponding difficulty level and achievement sub-system will display game progress and extra rewards to the user.

2. Risk Analysis Approach

In this risk analysis, we use the following online risk model[1] as a guide with modifications being made to the order in which the model is implemented. The main focus of this risk analysis is to assess availability and security risks rather than organizational risks, as a result of which we are only considering technical risks to the availability and integrity of our application. The impact of the vulnerabilities are implicitly mentioned within the descriptions, and the severity of each vulnerability gives an indication of the impact.

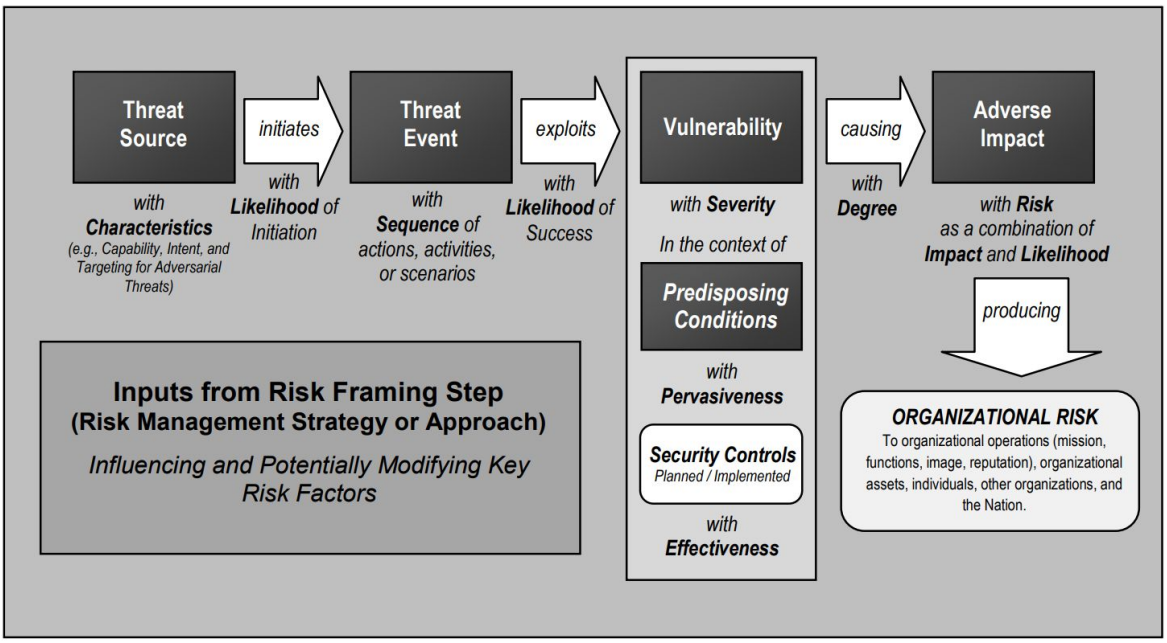


FIGURE 3: GENERIC RISK MODEL WITH KEY RISK FACTORS

[1] https://csrc.nist.gov/publications/detail/sp/800-30/rev-1/final

3. System Characterization

3.1 Technology components

Component	Description
Applications	AWS, Firebase
Networks & Protocols	HTTP protocol allows two-way transmission of data between users' computers and the server hosting the application.
User Subsystem	Manages all user account operations of the system
Level Subsystem	Controls user progression through the system by determining which level files a user has access to
Game Subsystem	Loads level files for the user to work on and complete. This component is responsible for execution of the main game screen
Achievement Subsystem	Interfaces with the user and level subsystems to determine which level files are available to a user depending on their progress

3.2 Data Used By System

Data	Description
User Database	Stores account and login information about users. Managed by user subsystem
Level Files	Contains instructions, expected output, available code blocks and other data required to complete a single level. Managed by the Level Subsystem and executed by the game subsystem.
Achievement Data	Includes the progress of the game, which level the users have accomplished

3.3 Users

Users	Description
Non-Account Users	Does not own an account in the user database. Progression within CodeBee is not saved on system exit.
Account User	Owns an account in the user database and uses it to log into CodeBee. Level progression is saved.
System Administrators	Have access to the AWS hosting service and database, with special permissions to make modifications to the entire system.

4. Vulnerability & Threats

4.1 Vulnerability Statement

Vulnerabilities were considered with each subsystem and possible supporting systems in mind. Each vulnerability affects one or more parts of CodeBee. Possible solutions to each vulnerability are proposed to remove or mitigate the possibility of an attack through each vector.

Vulnerability	Description	Solutions	Severity
SQL injection to the User Login Input	The user login system potentially has the vulnerability of SQL injection, such that the attacker could interfere with the queries that CodeBee makes to its database.	A reliable third party authentication service like Firebase. All data inserted into the database should be cleaned to prevent SQL code being entered as data.	Critical
Packet Sniffing	HTTP protocol can be insecure.	Deploy the web application using firebase hosting which comes with a free SSL certificate which will prevent security issues. Use HTTPS protocol instead of HTTP.	Minor
Denial-of-Service (DoS)	The web server being attacked by other parties and crashes.	Allow minimum user error and practicing Network Security include constructing complex passwords and security firewalls. Leverage the Cloud technology, rather than relying solely on on-premises architecture.	Critical
Malicious Data	Level and code block data is read by the system and this data is only produced by system developers. If malicious levels or code blocks are made,	Use of a data verification technique to determine origin or creator. Data can be stored on servers to prevent breaches to local data. Before executing code, the local	Major

	unverified code will be run on the system.	copy could be compared to a server 'master' copy.	
Privilege escalation	By gaining access to a system administrator's credentials, an attacker could modify the contents of the challenges provided to players and render the application incorrect and ineffective. In extreme cases, it could also cause the application to crash.	Two-factor authentication is provided by AWS and this should be leveraged to prevent unwanted access to the application as an administrator.	Critical
Unauthorized user access	The user's login credentials could be leaked / discovered by someone who could then use the user login to affect the user's progress in the application.	The user password can have restrictions which make it more secure. Two factor authentication can be used to ensure that unintentional access to the application is not allowed.	Minor
AWS unavailability	In the rare event that AWS servers are not accessible, the application would not be available to users.	AWS servers have a backup functionality, however in case these backups are also unavailable, HA/DR systems can be looked into.	Minor
Unavailability attack	Submitted code can cause infinite loops, requiring special logic which may be intentionally exploited to reduce server performance.	Limits to the amount of requests a single user can make in a given time period. Special logic which flags users suspected of system attacks.	Moderate
Cracking the password on administrator account	The attackers can use brute-force methodology to try every possible password to hack the administrator account.	Practicing Network Security include constructing complex passwords and security firewalls	Moderate

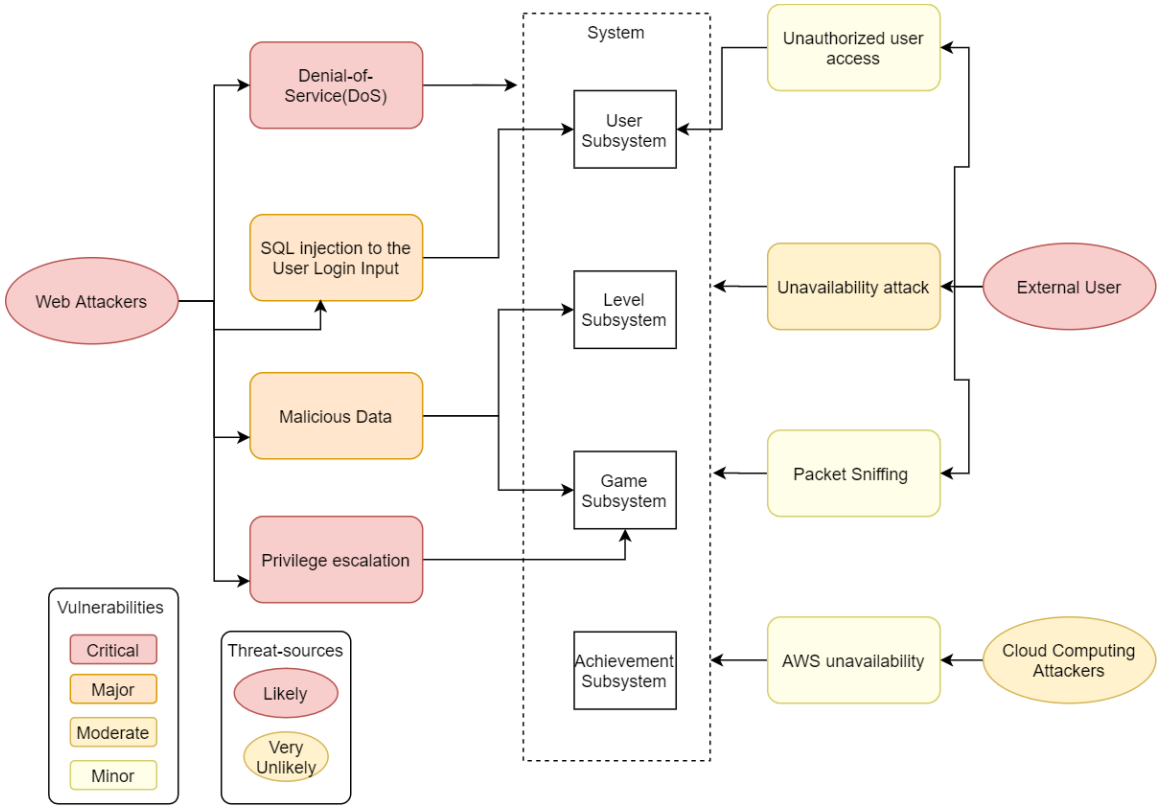
4.2 Threat Statement

Here we highlight the different possible sources that can be a potential threat to our system. These sources are responsible for exploiting the potential vulnerabilities of our system, either intentionally or unintentionally.

Threat-Source	Threat	Likelihood of Success
External User	Gain access to administrator accounts.	likely
External User	Gain all user account data information.	likely
Web Attackers	Make the application server traffic busy and unavailable (possibly use a bot to create as many requests).	likely
Cloud Computing Attackers	Make the third-party cloud server unavailable.	very unlikely

4.3 Flow Diagram

The following diagram shows how the threat-sources and vulnerabilities would affect our system and subsystems.



5. User Hazard Statement

The following table mentions how the application could potentially be harmful to the users, as well as potential solutions to mitigate said risks.

Hazard	Solutions	likelihood
Flashlights displayed may cause photosensitive epilepsy(PSE).	Provide a warning before user entering the game and use less offensive visual effects	very unlikely
Contents might be offensive to users with some cultural backgrounds.	Use content not specific to any cultural backgrounds	very unlikely
High volume or high frequency sound may be harmful for users' hearing.	Use stock sounds available on the internet	very unlikely

6. Conclusion on Risk Analysis

We conclude that the major risks concerning our application pertain to the security of our infrastructure - preventing unexpected access to the application or supporting infrastructure and applications such as AWS and Firebase, as well as ensuring the safety of our users' data privacy will be given a high priority at all stages of development. These measures shall improve the overall reliability of our application, ensuring high availability and professional-grade security measures which can be trusted by the users.

Minor issues that can arise due either due to user error or third party service failures, being unlikely, will not be addressed directly by our team, however care will be taken during the design stages to minimize the possibility of such errors as much as possible.