Florida Gulf Coast University

Template based on the Centers for Medicare & Medicaid Services, Information Security & Privacy Management's Assessment

Security Assessment Report

Version N.0

January 1, 1900

Table of Contents

1. Summary

Executive Summary Here: Describe the overall goal, method, and major findings/ recommendations here. (it's the TLDR)

The overall goal for completing this security assessment for the project I chose was to highlight vulnerabilities in the program and identify those which would be advantageous to implement to improve system security. The methods used to complete the security assessment are broad in scope and cover a variety of topics which were covered in the course over the semester, such as risk/threat assessment, implementation of Software Development Lifecycle (SDLC), using Access Controls and Authorization, looking at Network security, deperimeterization, penetration testing, etc. For recommendations, on both this and future projects, a significant take away is to plan the SDLC of the project from start to finish, conduct a thorough risk analysis for all/many possible threats, and implement Access Controls to a further degree.

1. Assessment Scope

What tools, platforms, OSes, Browsers, and software (including your own) was tested or used in testing?

- macOS
- Command Line
- Unity
- CLion
- Chrome

Major Limitations:

- Different OS testing unavailable
- Solo development/testing

2. Summary of Findings

Of the findings discovered during our assessment, 3 were considered Emergency Risks, 3 were considered Major Risks, 2 were considered Moderate Risks, 2 were considered Minor Risks, and 0 were considered Negatable Risks. The SWOT used for planning the assessment are broken down as shown in Figure 2.

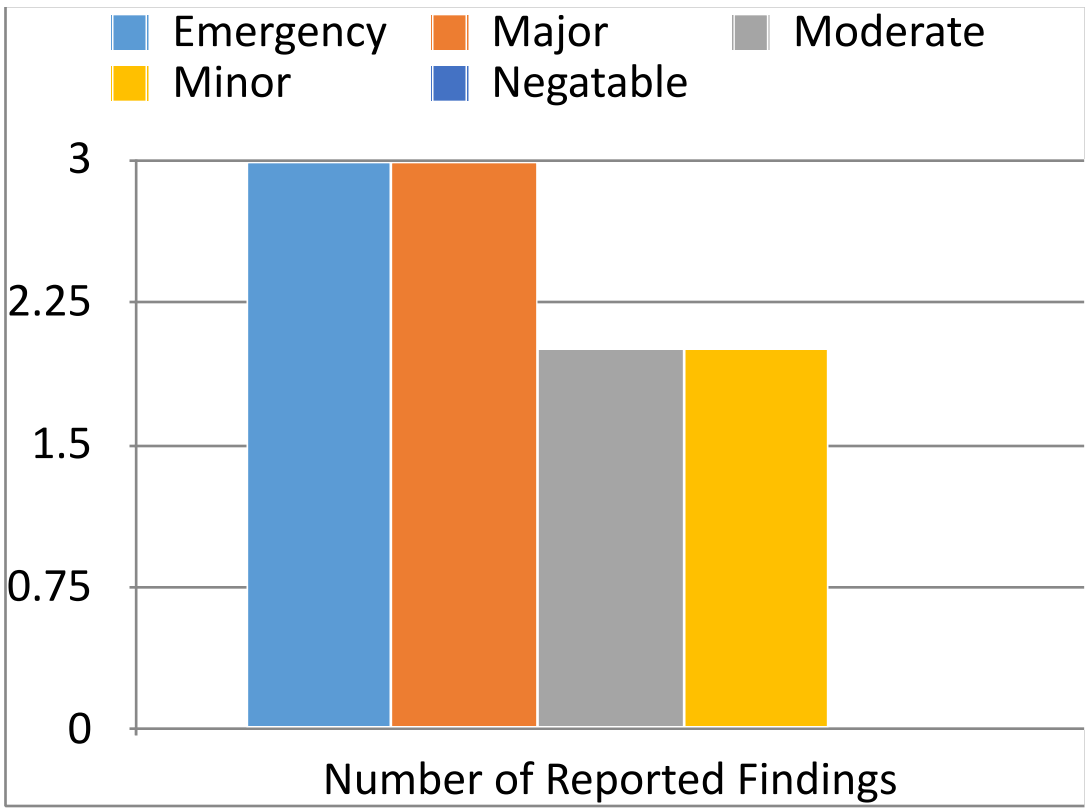


Figure 1. Findings by Risk Level

Explain above and link to full table of explanation of top risks like Figure 3.

SWOT Diagram: Driving Simulator Game (Updated)

<u>Hacking</u>: ’T’

1. User data - user information is compromised and used by unauthorized parties ’T'

   S - users may have the opportunity to be compensated

   O - minimize the amount of user information needed for account setup

2. Modifying the game code - players systems get attacked from malware written to game files ’T'

   S - repositories may help track the traffic of users attempting to develop the files

   O - implement access controls for developers who want to alter the game code

3. Development plans leaked - plans in the code for future content/DLC are leaked ’T'

   S -

   O - use secure communications channels and/or store DLC content elsewhere

4. Stealing Game Code - the code used for the game is leaked/stolen ‘W’

   S - the opportunity to copyright your work may help counteract this problem

   O - adding an encryption to the main files could be a layer of protection

5. Developers Info Stolen - dev. Credentials are stolen, could give access to authorized developers and/or release content w/ malware ’T'

   S -

   O - minimize the amount of sensitive information needed for devs and add additional authentication layers.

<u>Cheating</u>: ’T’

6. Cheats Installed & Used - players download and use a cheating program from the Internet ‘W’

   S -

   O - Implement an anti-cheat software into the project.

7. Code is altered - users alter the files in ways to make the game easier/play as unintended ‘W’

   S -

   O - Add encryption needed to access the files as well as checking for authenticated devs

8. Cheats developed and sold - users develop and sell a program to give players advantage ’T'

   S -

   O - an effective anti-cheat program could counteract individuals attempting to profit from ‘ruining’ the game

9. Scripting - macro’s are used by players to make the game easier by bypassing user inputs needed ’T'

   S - |

   O - have code to check conditions which match those that would happen when users run scripts instead of actually playing.

10. Boosting services - other users get paid by individuals who want their account leveled up ‘W’

   S - checking the IP address from where users login to play could help highlight this

   O - we can ban those users who violate the Terms of Service agreement

We can compare the severity of the risks using the basic risk assessment template. Out of the risks found in the original project, there were 3 considered to be Emergency; these include Modifying the game code, User/dev data being stolen, and Developers information being hacked; also, there were 3 risks in the Major section, being Development plans leaked, Cheats for the game are developed and sold, and Stealing Game Code. The other 4 risks fall in the lower risk categories of Moderate, Minor, and Negatable.

Figure 2. SWOT

Explain which issues were used from above SWOT (which are addressed in this assessment).

3. Summary of Recommendations

One of the more significant changes made to the project is the addition of a database to store and keep track of users; we can see both who logs on to play and at what time and permit different controls to users depending on their credentials/progression. Also, the development of future content/plans for the game has been moved to a different project; this allows for increased security for this part and less of a chance that this content is leaked before the release date, and can simply be inserted into the project when completed. There are still features which need to be included, such as file encryption for the game scripts, an anti-cheat software for the release version of the game, a developed user profile system, etc.

2. Goals, Findings, and Recommendations

1. Assessment Goals

The purpose of this assessment was to do the following:

- Highlight vulnerabilities in the projects design/code
- Identify features that would be advantageous to implement into the game
- Overall improve the project security

2. Detailed Findings

Ensure each vulnerability is thoroughly explained, specific risks to the continued operations are identified, and the impact of each Threat or Weakness is analyzed as a business case. Ensure these are linked to Table 1 when describing the Risk Value. This is not the fixes – it's the description of the problems found. The fixes go in the next section (for ease of lookup using TOC) - build this off your checklist, SWOT, and risk assessments.

1. Tracking users - Unidentified users, a.k.a. anonymous or unauthenticated users, can pose a number of security issues for a system or application; firstly, unidentified users can potentially gain access to sensitive or confidential information, as they are not subject to any access controls or authentication checks. They can engage in a range of malicious activities, such as attempting to exploit vulnerabilities in the system, injecting malicious code, or stealing sensitive data. Unidentified users can also launch DoS attacks on a system or application, which can cause the system to crash or become unavailable to legitimate users and even cause data breaches by accessing or manipulating sensitive information, such as personally identifiable information or financial data.

2. User data leaked - There are a number of ways that user data can be put at risk from video games; first of all, game companies may experience data breaches that result in the theft of user data. This can occur when hackers gain access to the game company's servers or when employees mishandle user data. Hackers may attempt to breach the game's servers or the player's device to gain access to user data. They may exploit vulnerabilities in the game's code or use social engineering tactics to trick players into providing their login credentials or other sensitive information. Phishing tactics may occur when attackers send players fake emails or messages that appear to be from the game company, asking for login credentials or other sensitive information. If players fall for this scam, their data may be compromised. Also, some games may use third-party services, such as social media, payment processors, and web tools that have access to user data. If these services experience a breach, user data may be leaked.

3. Modifying the code - If hackers alter program code, it's no surprise that this can have serious consequences. Hackers can inject malicious code into a program, which can then be used to perform unauthorized actions, steal data, or damage the system. They can also alter program code to exploit vulnerabilities in the program, which can be used to gain unauthorized access to a system or network. Altering program code can lead to system crashes or other unexpected behavior, which can result in data loss, downtime, or other disruptions to operations. If events like these occur, it can damage the reputation of the company/individuals who created the program, leading to loss of trust from customers and backers alike.

4. Cheats - The use of cheats not only pose a risk to the user who is using them to gain an unfair advantage, but even to the other players of the game. For example, some cheats require players to download and install software from unofficial sources. This software may contain malware or viruses that can compromise the player's device and steal sensitive data. Some cheats may exploit vulnerabilities in the game's code, which can be used to gain unauthorized access to the game or the player's device. Cheats can also affect the security of the game's network; for instance, cheats that allow players to modify network traffic can disrupt game servers and affect the game's performance. Lastly, in-game economies can be affected by

cheats that allow players to generate or obtain large amounts of in-game currency or items, which disrupt the game's economy and can negatively impact other players' experiences.

5. Stolen Code - When developer code is stolen, there are several consequences; Intellectual property theft occurs when stolen code can be used to create copycat products or to exploit the ideas and innovations of the original developer, leading to lost revenue and intellectual property theft. If stolen developer code is used by a competitor, it can give that competitor an unfair advantage in the marketplace, as they may be able to produce similar products at a lower cost or more quickly than the original developer. Stolen developer code may contain sensitive information, such as passwords, user data, or payment information, which can be used to breach the security of the original developer's systems or those of its customers.

6. Memory leaks - Memory leaks are a common type of software bug that occur when a program fails to free up memory that is no longer needed. In other words, memory leaks occur when a program continuously allocates memory but never deallocates it, leading to a gradual loss of available memory. Several negative effects can occur as a result; as more memory is allocated and not released, the program will gradually slow down, as it has less memory to work with; if a program continues to allocate memory without releasing it, it may eventually run out of available memory, causing the program or system to crash. In some cases, memory leaks can create security vulnerabilities, as an attacker may be able to exploit the leaked memory to gain access to sensitive data or execute arbitrary code. (*Understanding memory leaks in Java*. Baeldung.)

7. Backup - Not having a backup can lead to several security issues; without a backup, if data is lost due to hardware failure, human error, or cyberattacks such as ransomware, the data cannot be recovered, leading to permanent data loss. Without a backup, if a cyberattack occurs, such as a ransomware attack, the victim may be forced to pay the ransom to recover their data. Additionally, if backups are not in place, attackers may be able to infiltrate and compromise systems undetected. Depending on the nature of the data, certain legal and regulatory compliance requirements may require the need for regular backups to be taken, and failure to do so can lead to legal penalties and fines.

8. Buffer Overflow - Buffer overflow is a type of software vulnerability that occurs when a program writes more data into a buffer, which is a temporary storage area in memory, than it can handle. This extra data can overwrite adjacent memory locations, causing the program to behave unexpectedly or crash. In other words, when a program receives input, it stores that input in a buffer, which has a certain size limit. If an attacker sends more data than the buffer can hold, the extra data can overflow into adjacent memory locations and overwrite data that was not intended to be modified. This can allow the attacker to execute arbitrary code or gain control of the program. (*What is buffer overflow? attacks, types & vulnerabilities*. Fortinet.)

3. Recommendations

Here's where your fixes go (ensure you reference Table 2 for your ease of fix evaluation and explain why it matches that category).

1. User profile system - There are several security benefits of having a user profile system in a video game; User profile systems can provide authentication and authorization features, ensuring that only authorized users have access to the game and its features. They can use encryption to protect communication between the game client and server, preventing unauthorized access and data theft. User profile systems can protect sensitive user data, such as personal information, login credentials, and payment information, from theft or unauthorized access, and may provide account recovery features, allowing users to regain access to their accounts in the event of a lost password or other issue.

2. Implementing encryption - Adding encryption to a project can help improve security by protecting sensitive data from unauthorized access and data theft. Encryption can help protect sensitive data, such as passwords, personal information, and payment information, from theft or unauthorized access. Even if an attacker gains access to the encrypted data, they will not be able to read it without the decryption key. Encryption can also help secure communication channels between the client and server, preventing eavesdropping and man-in-the-middle attacks. This is particularly important for projects that involve transmitting sensitive data over the internet. In addition, some industries and organizations have legal or regulatory requirements for data encryption. Adding encryption to a project can help ensure that you comply with these requirements and avoid potential legal or financial consequences.

3. Anti-cheat software - Anti-cheat software is designed to detect and prevent cheating in video games. Anti-cheat software helps ensure that all players are competing on a level playing field, without any unfair advantages gained through cheating. Cheating can negatively impact the overall game experience for all players; anti-cheat software can help create a more enjoyable and engaging game experience by preventing cheating. Some cheats are designed to exploit vulnerabilities in the game client or server, which can create security risks for both the game and its users. Anti-cheat software can help identify and prevent these exploits, improving the overall security of the game.

4. Handle memory leaks - Fixing memory leaks can improve project security by reducing the likelihood of exploits that could be used by attackers to compromise the system. When a memory leak is fixed, the program will no longer consume unnecessary memory, reducing the

likelihood of crashes or unexpected behavior. Additionally, fixing memory leaks can help reduce the likelihood of memory-related exploits, making it more difficult for attackers to exploit vulnerabilities in the program. Although fixing memory leaks alone may not be sufficient to provide complete security, developers can still improve the overall stability and reliability of their program, as well as its security.

5. Loss prevention methods - Protecting the code of a project is essential to maintaining the integrity and security of the project. One method is implementing access controls; limiting access to the code repository and associated tools, such as build systems and deployment servers, can help prevent unauthorized access to the code. Access control measures can include password-protected accounts, multi-factor authentication, and role-based access control. Code signing is another one of these methods, and involves digitally signing the code with a cryptographic key to ensure that it has not been modified or tampered with. This can help prevent unauthorized modifications to the code and provide assurance that the code is authentic. Regularly testing the code for vulnerabilities and weaknesses can help identify potential security issues before they can be exploited. Security testing measures can include vulnerability scanning, penetration testing, and code review.

6. Fix buffer issues - Fixing buffer overflow vulnerabilities can involve a range of strategies, depending on the specific nature of the vulnerability. Some common strategies include input validation to prevent data from being written beyond the bounds of a buffer, using safer programming practices to avoid buffer overflows, and using security-focused programming languages or libraries that are designed to prevent buffer overflow vulnerabilities. When a buffer overflow vulnerability is fixed, the program will no longer write data beyond the bounds of the buffer, reducing the likelihood of memory corruption and exploitation. This can help improve the overall security of the program by reducing the attack surface and making it more difficult for attackers to exploit vulnerabilities.

7. Backup - There are numerous reasons as to why we want to have backed up data for the projects we work on; Backups can prevent data loss due to hardware failures, software bugs, or user errors. If your project data is lost or corrupted, you can restore it from a backup and minimize the impact on your work. In the event of a disaster such as a fire, flood, or theft, backups can be used to restore your project data and resume work as quickly as possible. They can be used to maintain a version history of your project. If you need to revert to an earlier version of your project, you can use a backup to restore the older version. Also, backups can be used to share your project with collaborators, such as team members or clients. By providing them with a backup, you can ensure that they have access to the most up-to-date version of your project.

3. Methodology for the Security Control Assessment

3.1.1 Risk Level Assessment (delete this text: you don't have to change 3.1.1)

Each Business Risk has been assigned a Risk Level value of High, Moderate, or Low. The rating is, in actuality, an assessment of the priority with which each Business Risk will be viewed. The definitions in Table 1 apply to risk level assessment values (based on probability and severity of risk). While Table 2 describes the estimation values used for a risk's "ease-of-fix".

Table 1 - Risk Values

Rating	Definition of Risk Rating
High Risk	Exploitation of the technical or procedural vulnerability will cause substantial harm to the business processes. Significant political, financial, and legal damage is likely to result
Moderate Risk	Exploitation of the technical or procedural vulnerability will significantly impact the confidentiality, integrity and/or availability of the system, or data. Exploitation of the vulnerability may cause moderate financial loss or public embarrassment to organization.
Low Risk	Exploitation of the technical or procedural vulnerability will cause minimal impact to operations. The confidentiality, integrity and availability of sensitive information are not at risk of compromise. Exploitation of the vulnerability may cause slight financial loss or public embarrassment
Informational	An "Informational" finding, is a risk that has been identified during this assessment which is reassigned to another Major Application (MA) or General Support System (GSS). As these already exist or are handled by a different department, the informational finding will simply be noted as it is not the responsibility of this group to create a Corrective Action Plan.
Observations	An observation risk will need to be "watched" as it may arise as a result of various changes raising it to a higher risk category. However, until and unless the change happens it remains a low risk.

Table 2 - Ease of Fix Definitions

Rating	Definition of Risk Rating
Easy	The corrective action(s) can be completed quickly with minimal resources, and without causing disruption to the system or data
Moderately Difficult	Remediation efforts will likely cause a noticeable service disruption • A vendor patch or major configuration change may be required to close the vulnerability • An upgrade to a different version of the software may be required to address the impact severity • The system may require a reconfiguration to mitigate the threat exposure • Corrective action may require construction or significant alterations to the manner in which business is undertaken
Very Difficult	The high risk of substantial service disruption makes it impractical to complete the corrective action for mission critical systems without careful scheduling • An obscure, hard-to-find vendor patch may be required to close the vulnerability • Significant, time-consuming configuration changes may be required to address the threat exposure or impact severity • Corrective action requires major construction or redesign of an entire business process
No Known Fix	No known solution to the problem currently exists. The Risk may require the Business Owner to: • Discontinue use of the software or protocol • Isolate the information system within the enterprise, thereby eliminating reliance on the system In some cases, the vulnerability is due to a design-level flaw that cannot be resolved through the application of vendor patches or the reconfiguration of the system. If the system is critical and must be used to support on-going business functions, no less than quarterly monitoring shall be conducted by the Business Owner, and reviewed by IS Management, to validate that security incidents have not occurred

3.1.2 Tests and Analyses

SEE 'RISK ASSESSMENT' IN FIGURES:

Accurate:

- Altered game code
- Use of cheats
- Changing game functionality
- User data
- Stolen project/code
- Cheats developed and distributed
- Development plans leaked

Inaccurate:

- Boosting services
- Scripting
- Dev's Info Stolen

3.1.3 Tools

This was completed using <list and describe any tools used for testing (include Linux Command Line commands>.

1. Penetration testing - Penetration testing involves evaluating the security of a system or application by attempting to exploit vulnerabilities or weaknesses in its security controls. The goal of penetration testing is to identify and address potential security issues before they can be exploited by attackers. Penetration testing typically involves a series of steps, including planning, reconnaissance, enumeration, vulnerability scanning, exploitation, post-exploitation, and reporting. The testing can be performed from both internal and external perspectives, and can be conducted in various ways, including black box, white box, and gray box testing. Penetration testing is an important component of a comprehensive information security program, as it can help organizations identify and remediate vulnerabilities before they can be exploited by attackers. Penetration testing can also provide valuable insights into the effectiveness of existing security controls and can help organizations make informed decisions about where to invest in future security improvements. (*Penetration testing*. What is Penetration Testing? | Core Security.)

2. White box/Black box testing - Black box testing and white box testing are two different approaches to testing the security of a system or application. Black box testing involves testing the system or application from the perspective of an external attacker, without knowledge of the internal workings of the system. The tester is typically only given information about the system's inputs and outputs and is not provided access to the underlying code or architecture. The goal of black box testing is to identify vulnerabilities that could be exploited by an external attacker. White box testing, on the other hand, involves testing the system or application with full knowledge of the internal workings, including access to the source code, architecture, and system design. The tester is able to identify vulnerabilities that could be exploited by both external and internal attackers. Each approach has its own advantages and disadvantages. Black box testing is more realistic and can identify vulnerabilities that would be visible to an external attacker, but it may miss some vulnerabilities that are not obvious from the outside. White box testing, on the other hand, can

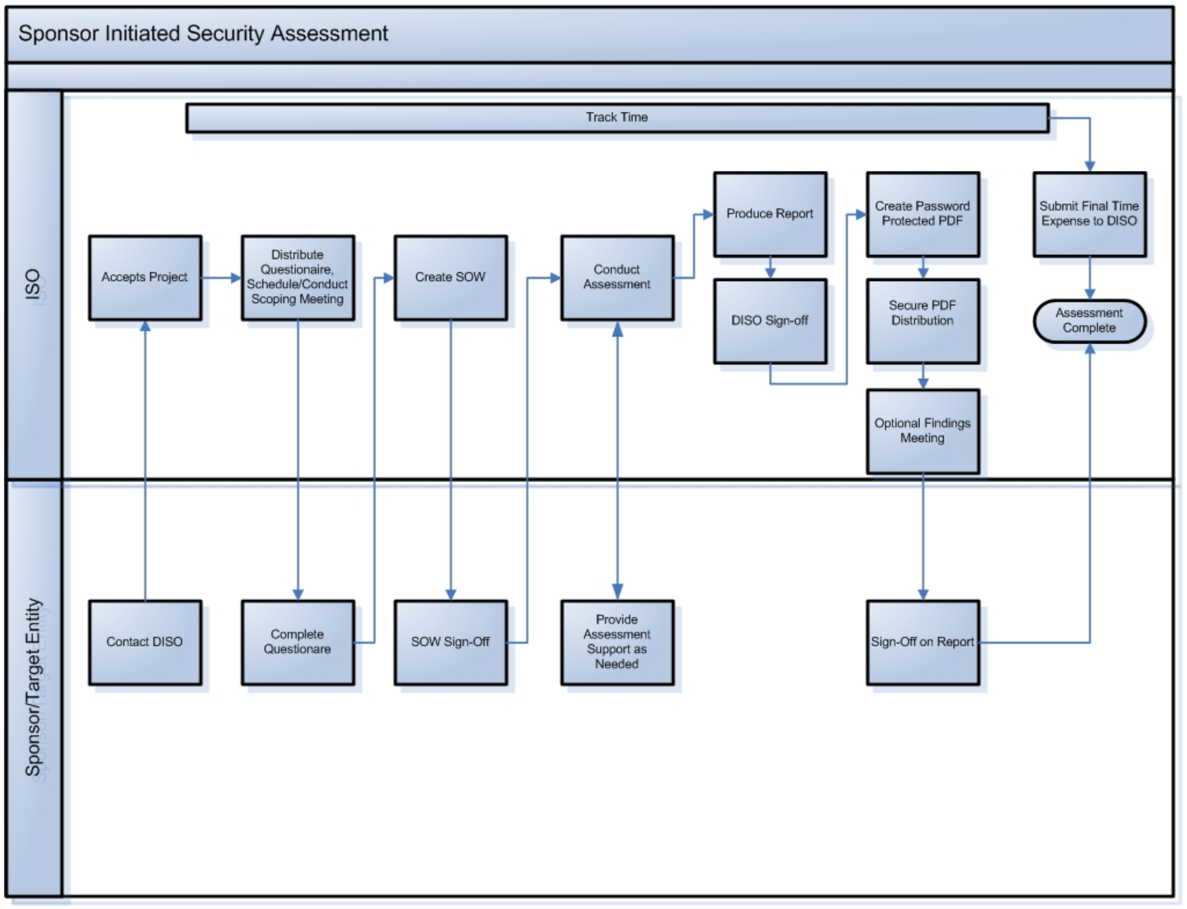
provide more comprehensive coverage and can identify vulnerabilities that would not be visible to an external attacker, but it may be more expensive and time-consuming. (*Black box vs. white box testing: Understanding 3 key differences*.)(*Differences between black box testing vs white box testing*.)

3. Access Controls and Authorization - Access controls and authorization are two important concepts in information security. Access controls are mechanisms used to limit or control access to a system or application. Access controls can be physical, technical, or administrative, and are typically used to ensure that only authorized users are able to access sensitive or confidential information. Authorization, on the other hand, is the process of granting or denying access to specific resources or functionality within a system or application. Authorization is typically based on a user's identity, role, or other attributes, and is used to ensure that users are only able to access the resources or functionality that they are authorized to use. Both access controls and authorization are important components of a comprehensive information security program. Access controls can help prevent unauthorized access to a system or application, while authorization ensures that users are only able to access the resources or functionality that they need to do their job. (*What is access control? | authorization vs authentication | cloudflare*.)

4. Figures and Code

Insert any pictures here (including of major code issues or code that was used as a tool – can just screenshot and add link to github). This section must include at least 4 figures or code portions:

1.1. Process or Data flow of System (this one just describes the process for requesting), use-cases, security checklist, graphs, etc.

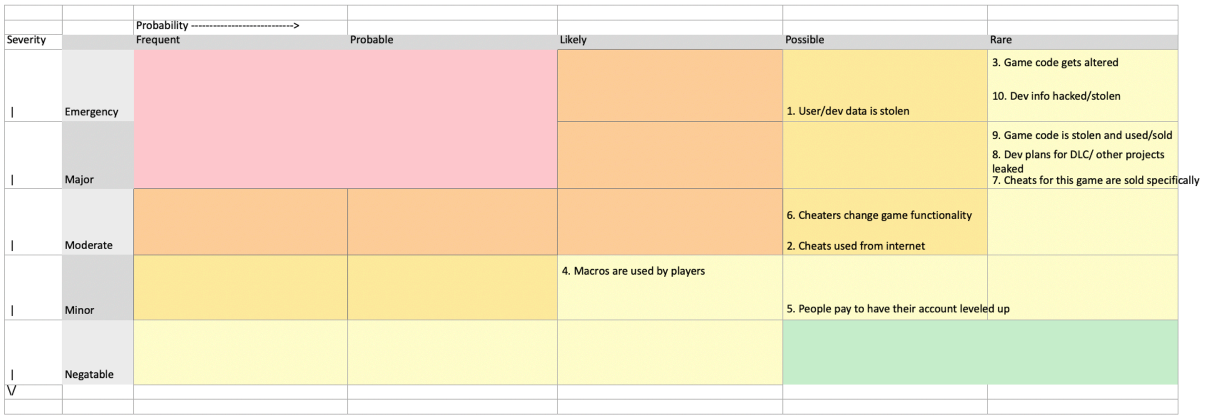

Sponsor Initiated Security Assessment
ISO
Sponsor/Target Entity
Track Time
Accepts Project
Distribute Questionaire, Schedule/Conduct Scoping Meeting
Create SOW
Conduct Assessment
Produce Report
DISO Sign-off
Create Password Protected PDF
Secure PDF Distribution
Optional Findings Meeting
Submit Final Time Expense to DISO
Assessment Complete
Contact DISO
Complete Questionare
SOW Sign-Off
Provide Assessment Support as Needed
Sign-Off on Report

Describe the process flow here.

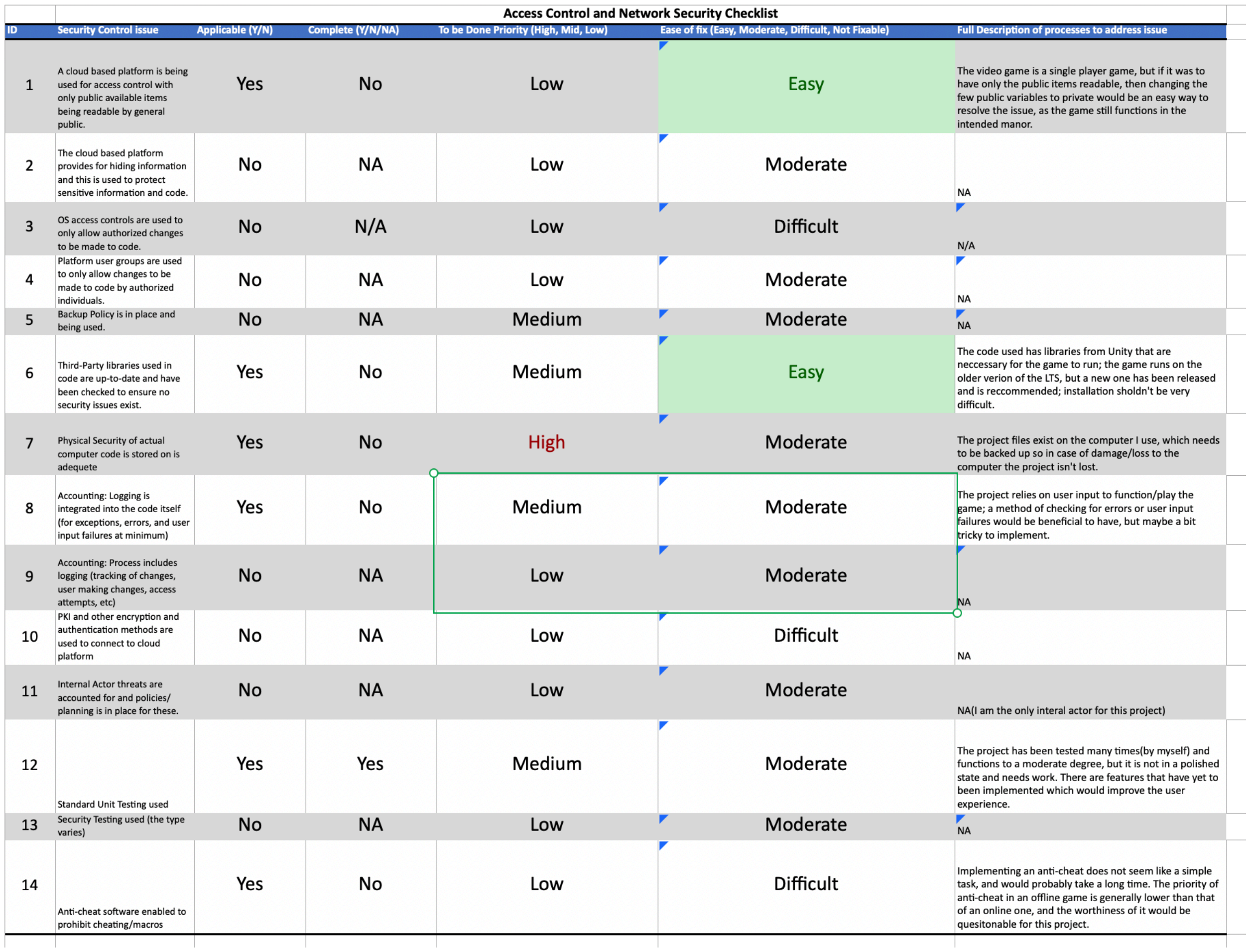
1.2. Other figure of code

HERE

Project Risk Assessment:



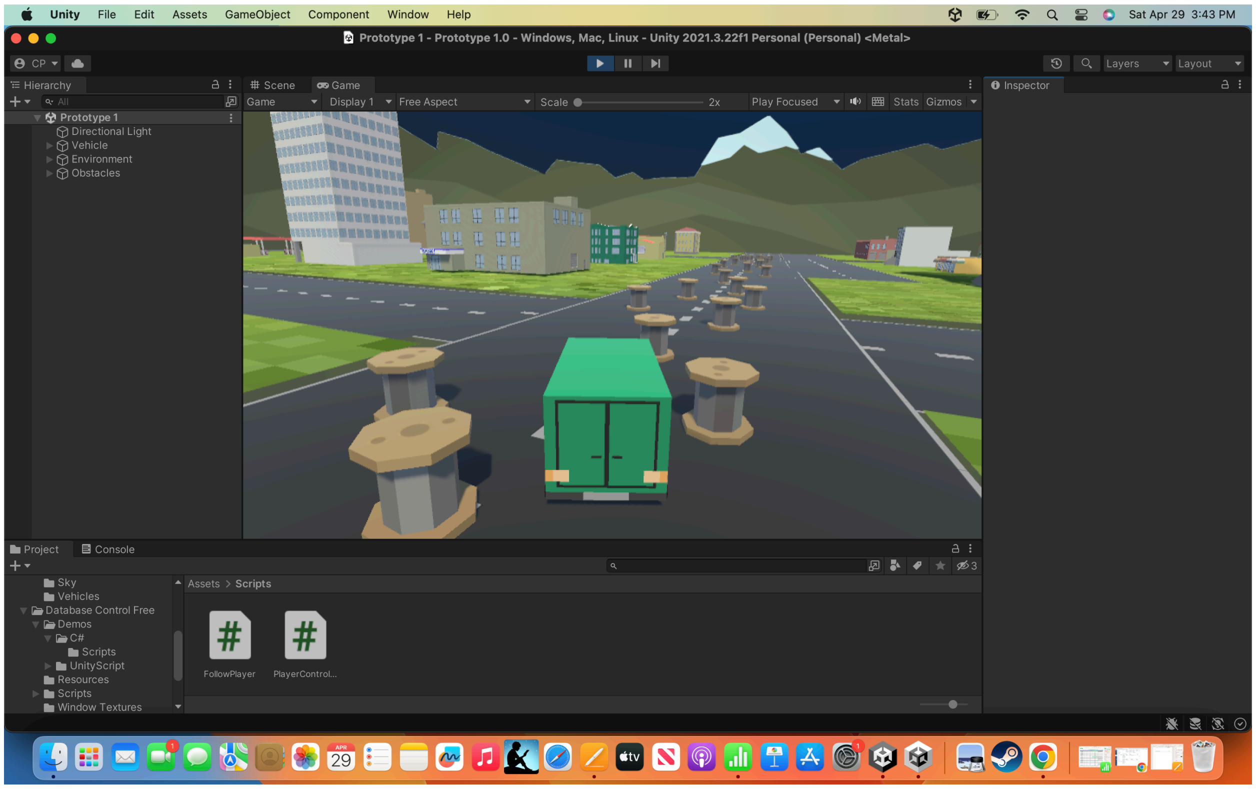
Severity		Probability ----------------------------> Frequent	Probable	Likely	Possible	Rare
|	Emergency				1. User/dev data is stolen	3. Game code gets altered 10. Dev info hacked/stolen
|	Major					9. Game code is stolen and used/sold 8. Dev plans for DLC/ other projects leaked 7. Cheats for this game are sold specifically
|	Moderate				6. Cheaters change game functionality 2. Cheats used from internet	
|	Minor			4. Macros are used by players	5. People pay to have their account leveled up	
|	Negatable					
V						



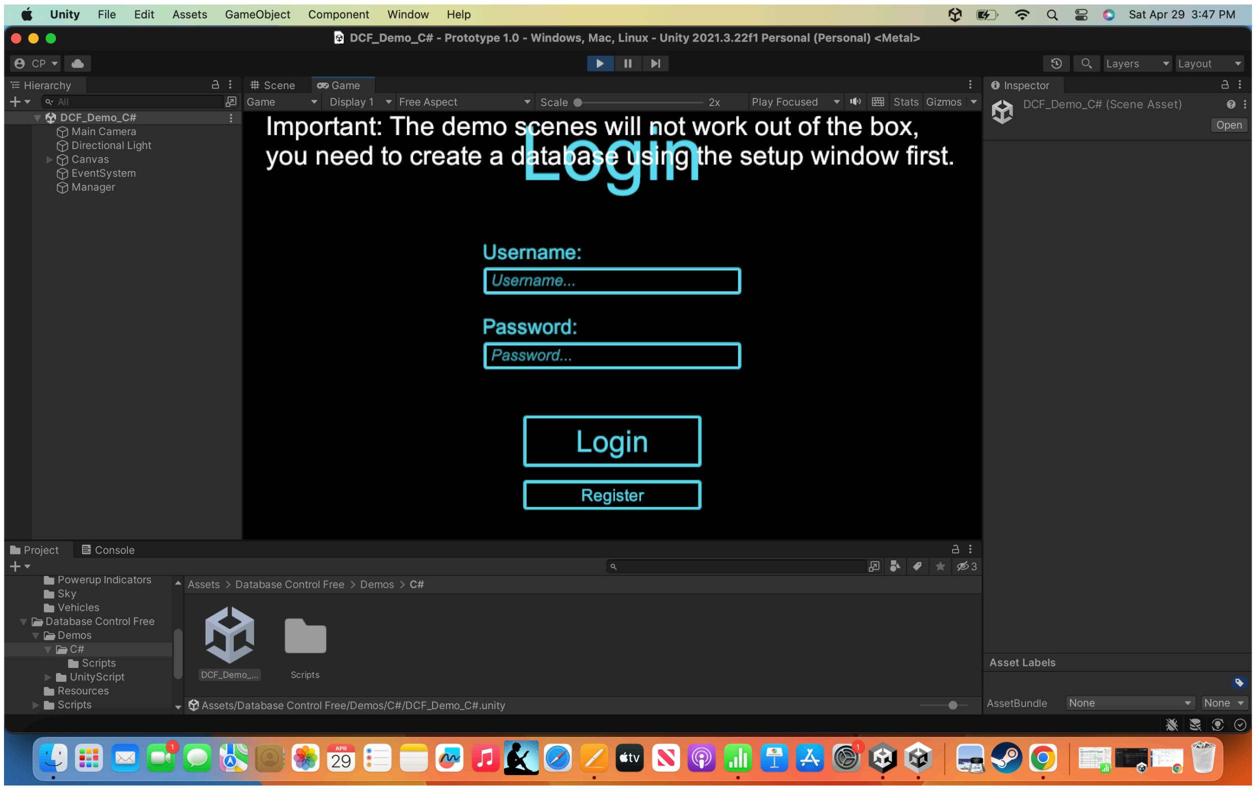
Access Control and Network Security Checklist

ID	Security Control issue	Applicable (Y/N)	Complete (Y/N/NA)	To be Done Priority (High, Mid, Low)	Ease of fix (Easy, Moderate, Difficult, Not Fixable)	Full Description of processes to address issue
1	A cloud based platform is being used for access control with only public available items being readable by general public.	Yes	No	Low	Easy	The video game is a single player game, but if it was to have only the public items readable, then changing the few public variables to private would be an easy way to resolve the issue, as the game still functions in the intended manor.
2	The cloud based platform provides for hiding information and this is used to protect sensitive information and code.	No	NA	Low	Moderate	NA
3	OS access controls are used to only allow authorized changes to be made to code.	No	N/A	Low	Difficult	N/A
4	Platform user groups are used to only allow changes to be made to code by authorized individuals.	No	NA	Low	Moderate	NA
5	Backup Policy is in place and being used.	No	NA	Medium	Moderate	NA
6	Third-Party libraries used in code are up-to-date and have been checked to ensure no security issues exist.	Yes	No	Medium	Easy	The code used has libraries from Unity that are neccessary for the game to run; the game runs on the older verion of the LTS, but a new one has been released and is reccommended; installation shoudn't be very difficult.
7	Physical Security of actual computer code is stored on is adequete	Yes	No	High	Moderate	The project files exist on the computer I use, which needs to be backed up so in case of damage/loss to the computer the project isn't lost.
8	Accounting: Logging is integrated into the code itself (for exceptions, errors, and user input failures at minimum)	Yes	No	Medium	Moderate	The project relies on user input to function/play the game; a method of checking for errors or user input failures would be beneficial to have, but maybe a bit tricky to implement.
9	Accounting: Process includes logging (tracking of changes, user making changes, access attempts, etc)	No	NA	Low	Moderate	NA
10	PKI and other encryption and authentication methods are used to connect to cloud platform	No	NA	Low	Difficult	NA
11	Internal Actor threats are accounted for and policies/ planning is in place for these.	No	NA	Low	Moderate	NA(I am the only interal actor for this project)
12	Standard Unit Testing used	Yes	Yes	Medium	Moderate	The project has been tested many times(by myself) and functions to a moderate degree, but it is not in a polished state and needs work. There are features that have yet to been implemented which would improve the user experience.
13	Security Testing used (the type varies)	No	NA	Low	Moderate	NA
14	Anti-cheat software enabled to prohibit cheating/macros	Yes	No	Low	Difficult	Implementing an anti-cheat does not seem like a simple task, and would probably take a long time. The priority of anti-cheat in an offline game is generally lower than that of an online one, and the worthiness of it would be quesitonable for this project.

Project Running:



Database in Project:



Important: The demo scenes will not work out of the box, you need to create a database using the setup window first.

5. Works Cited

There are no sources in the current document.

baeldung, W. by: (2022, July 14). *Understanding memory leaks in Java*. Baeldung. Retrieved April 29, 2023, from https://www.baeldung.com/java-memory-leaks

GeeksforGeeks. (2023, April 19). *Differences between black box testing vs white box testing*. GeeksforGeeks. Retrieved April 29, 2023, from https://www.geeksforgeeks.org/differences-between-black-box-testing-vs-white-box-testing/

Penetration testing. What is Penetration Testing? | Core Security. (n.d.). Retrieved April 29, 2023, from https://www.coresecurity.com/penetration-testing

What is access control? | authorization vs authentication | cloudflare. (n.d.). Retrieved April 29, 2023, from https://www.cloudflare.com/learning/access-management/what-is-access-control/

What is buffer overflow? attacks, types & vulnerabilities. Fortinet. (n.d.). Retrieved April 29, 2023, from https://www.fortinet.com/resources/cyberglossary/buffer-overflow#:~:text=Also%20known%20as%20a%20buffer,the%20data%20in%20those%20locations.

Writer, H. A. T., Ashtari, H., Writer, T., 29, L. U. S., Hossein Ashtari Technical Writer. (2022, September 29). *Black box vs. white box testing: Understanding 3 key differences*. Spiceworks. Retrieved April 29, 2023, from https://www.spiceworks.com/tech/devops/articles/black-box-vs-white-box-testing/