Assignment 1: Requirements and Design

A. Introduction

1. Team Name: UHM Cyber Defense
2. Team Members:
    - Michael Boyle
    - Kenneth Yamaguchi-Townsend
    - Joshua Ducey
    - Pauline Wu
3. Application Title: Network Defender

    Description: Game - Endless Horde Defense

    Functional Requirements Specification:
    - User accounts
    - Scoreboard file
    - Uses UNITY game engine
        - All modules implemented by students
    - Arena/battlefield to defend
    - Player character entity
    - Enemy AI entities - multiple types
    - 3 player weapons
        - Possible expansion - more weapons and/or powerups
    - 3D models, textures, animations
    - Sounds
4. Type of program: Desktop-based
5. Development Tools
    - Programming language: C#
    - IDE: Unity

B. Requirements

1. <u>Security Requirements</u>:
    - Authentication:
        - The program will correctly identify and verify users before allowing access.
        - The program will use a username and password combination for accounts.
    - Data Integrity
        - The program will ensure data is not intentionally corrupted (e.g., fake scores).
        - The program will provide anti-cheat measures (e.g., health and damage value validation).
        - The program will sanitize player text inputs to prevent code execution attacks.

<u>Privacy Requirements</u>:

- Data Confidentiality
    - The program will not allow unauthorized users access to any stored data (e.g., the scoreboard).
- Anonymity
    - The program will not store personal information about the users.

<u>System of keeping track of security flaws</u>:

- Manual source code reviews and testing.
- Risk analysis repeated iteratively throughout the software development process.
- When security flaws are discovered, they will be analyzed to determine whether they originated from the requirements, design, or implementation.
- All security flaws will be tracked by the team and corrected to remove the risk associated with them.

2. Quality Gates (or Bug Bars):
    - The level of security of our program would be moderate. Because we are going to make a game which would not contain much important information. The users would need to properly install the program before they could use it. In addition, they will need authentication by email or some other ways before they could sign up for an account. No one will have admin access unless they have the authorization.
    - The level of privacy is also moderate because since it is a desktop-based program, it may involve transferring data from one computer to another one. Which may result in sharing data with other computers. Although the PII is stored locally on each computer that has this program, the share of files and some sensitive data will happen when transferring.

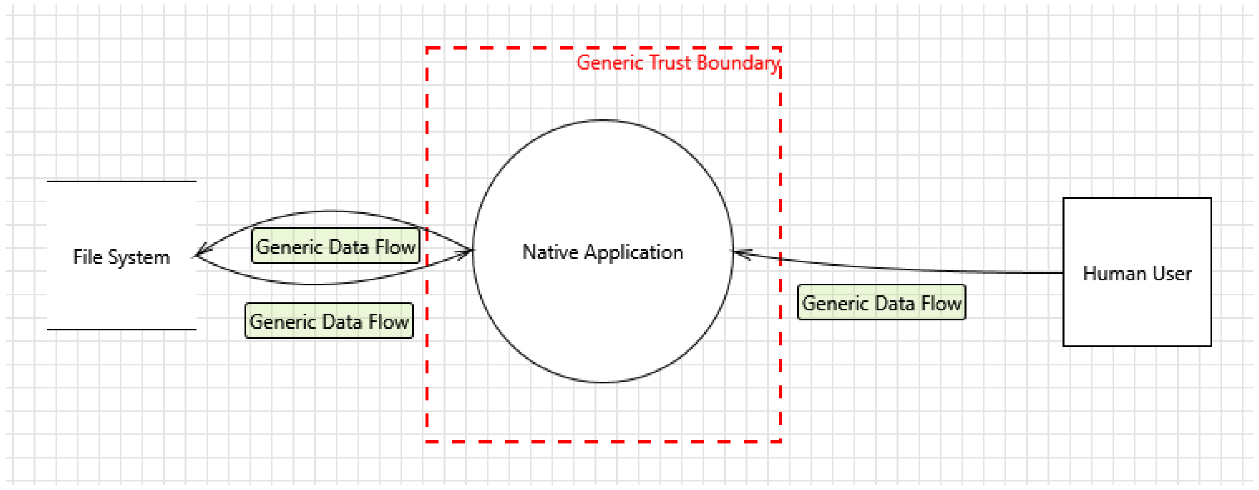
3. Risk Assessment Plan for Security and Privacy:
    - Everyone on the team will be responsible for security and privacy. It is everyone's obligation to constantly monitor and investigate new ways to lower overall security and privacy risk.
    - The Privacy Impact Rating of the program is P1 solely because it is a single-player game that provides an experience that may be attractive to children
    - No personally identifiable information (PII) of any kind will be stored on the user's computer or transferred from the user's computer to the program.
    - The parts of the program that will need threat modelling are the entry points from the user and the file system.
    - The dataflow from the user and the file system will need security reviews.

C. Design

1. Design Requirements:
   - Implement only the basic features of UNITY Engine needed
     - Example: Single player so do not implement MP netcode
   - Additional features made by team - Starting with Basic req. Only.
     - Defining Game Types
     - Defining Entities
     - Defining Menus, Options
   - Art and Sound Assets - Open Licensed or Team Made
   - Player Account and Login Functionality
   - Anti-Cheat Code - Either built in checks or external addition
   - Scoreboard Encryption and validation checks
   - Program will only write its own outputs with no input from players
     - Exception is data entry for account creation
     - Inputs will be sanitized before addition to player record

2. Attack Surface Analysis and Reduction:
   - User level privileges only.
     - Program will have no user accessible debug console
   - Possible Attack Vectors
     - Player Account Authentication Email System
     - Player Information Entry
     - Any existing UNITY Engine flaws
     - Player information I/O
     - Scoreboard I/O
     - Memory/Buffer Overflows in team made features

3. <u>Threat Modeling</u>:



Human User: Other than controls for the game itself, the user only inputs data for account creation. As stated above, this text entry will be sanitized. Controls within the game will not be text-based and do not present an obvious risk.

File System: The game then interacts directly with the file system to store user data and the scoreboard files. These files will be encrypted and can only be modified by the native application.

Native Application: The game cannot trust the File System or the Human User by default. The files need to be checked for unauthorized changes (checksum validation) and the inputs of the user need to be sanitized to prevent injections and cheating (e.g., using macros to provide unfair advantage).

Assignment 2: Implementation

A. Approved Tools

Mac OS Requirements

Compiler/ Tool	Minimum Required Version and Switches/Options	Optimal/Recommended Version and Switches/Options	Comments
Game Development/Code Compiler	Unity Personal Edition 2017.1	Unity Personal Edition 2019.3	
C# Editor	C# editor of choice	Visual Studio for Mac 8.4.6	Must use C# v2.0 or later
FxCop	Version 1.32	Most recent version	
2D Art	GIMP 2.8	GIMP 2.10.14	

Win64 Requirements

Compiler/ Tool	Minimum Required Version and Switches/Options	Optimal/Recommended Version and Switches/Options	Comments
Game Development/Code Compiler	Unity Personal Edition 2017.1	Unity Personal Edition 2019.3	
Visual Studio	Visual Studio 2008	Visual Studio 2019	Must use C# v2.0 or later
FxCop	Version 1.32	Most recent version	
3D Art	Autodesk Softimage XSI 5.0 Adv.	Autodesk Softimage XSI 7.0 Adv.	Newer 3D graphics application can be used
2D Art	GIMP 2.8	GIMP 2.10.14	

Content Disclaimer

This documentation is not an exhaustive reference on the SDL process as practiced at UHM Cyber Defense. Additional assurance work may be performed by product teams (but not necessarily documented) at their discretion. As a result, this example should not be considered as the exact process that UHM Cyber Defense follows to secure all products.

B. Deprecated/Unsafe Functions

Old Unity Versions on Windows: While the version of Unity does not need to be upgraded (i.e. 2017 to 2019), the patched version needs to be installed to protect against remote code execution.

Lightmapping using Enlighten: Enlighten is used for global illumination. However, it is deprecated and will be removed soon. Instead, we will either use Progressive GPU Lightmapper or Progressive CPU Lightmapper.

Unity Remote 3: Unity Remote 3 is now a deprecated tool used when you want to quickly test a project on an iOS device without completely building and deploying the game after a small change. The new version has simply been named Unity Remote and now works for iOS, Android, and tvOS for quick testing.

Obsolete API warnings and automatic updates: Among other messages, Unity shows warnings about the usage of obsolete API calls in your code. So, for example, you could access a Rigidbody on the object using code like:

```
// The "rigidbody" variable is part of the class and not declared in the user script.
   Vector3 v = rigidbody.velocity;
```

These shortcuts have been deprecated, so you should now use code like:

```
  // Use GetComponent to access the component.
   Rigidbody rb = GetComponent<Rigidbody>();
   Vector3 v = rb.velocity;
```

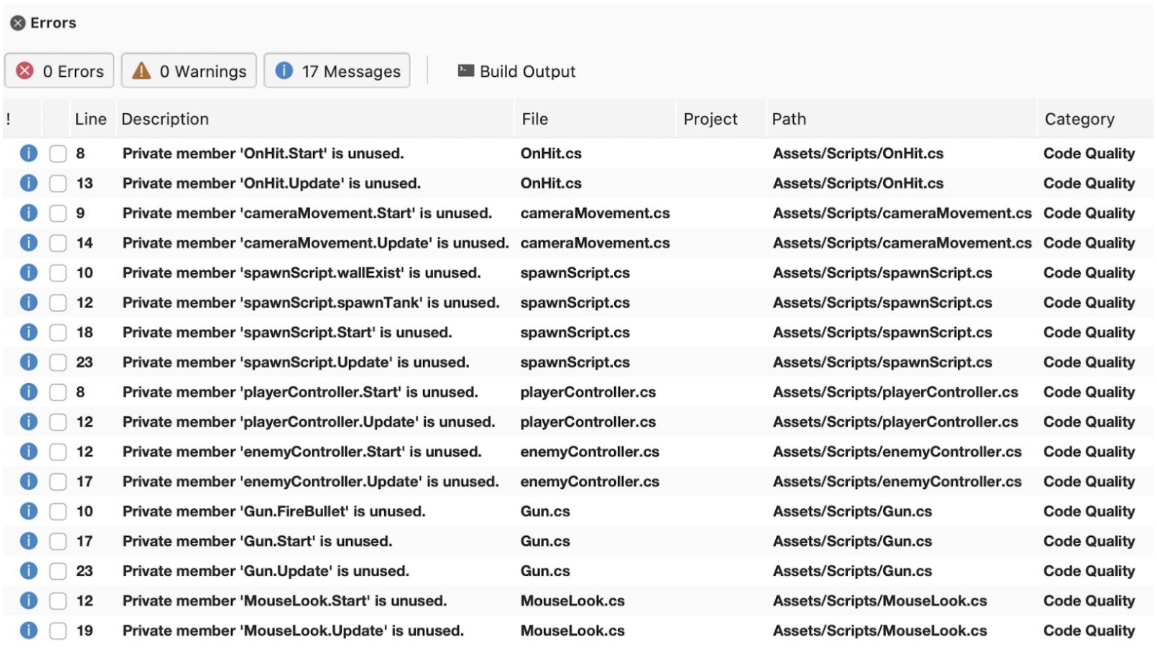
Source: https://docs.unity3d.com/Manual/Console.html

C. Static Analysis

After searching about the static analysis tools that we could use for C#, we decided to use FxCop. Because it has been integrated into Visual Studio 2012 and later as Code Analysis, and the program we are making involves using Visual Studio. Which makes it convenient for us to use this tool every time when we are coding and trying to compile the code.

In order for us to use FxCop in the Visual Studio, we will need to use it as an external tool to analyse our code and it gives a report after the analysis is done. There is a lot of useful documentation online which we may need later as we continue to use this tool while developing our program.

Below is a screen shot of the result after the analysis:



!		Line	Description	File	Project	Path	Category
●	☐	8	Private member 'OnHit.Start' is unused.	OnHit.cs		Assets/Scripts/OnHit.cs	Code Quality
●	☐	13	Private member 'OnHit.Update' is unused.	OnHit.cs		Assets/Scripts/OnHit.cs	Code Quality
●	☐	9	Private member 'cameraMovement.Start' is unused.	cameraMovement.cs		Assets/Scripts/cameraMovement.cs	Code Quality
●	☐	14	Private member 'cameraMovement.Update' is unused.	cameraMovement.cs		Assets/Scripts/cameraMovement.cs	Code Quality
●	☐	10	Private member 'spawnScript.wallExist' is unused.	spawnScript.cs		Assets/Scripts/spawnScript.cs	Code Quality
●	☐	12	Private member 'spawnScript.spawnTank' is unused.	spawnScript.cs		Assets/Scripts/spawnScript.cs	Code Quality
●	☐	18	Private member 'spawnScript.Start' is unused.	spawnScript.cs		Assets/Scripts/spawnScript.cs	Code Quality
●	☐	23	Private member 'spawnScript.Update' is unused.	spawnScript.cs		Assets/Scripts/spawnScript.cs	Code Quality
●	☐	8	Private member 'playerController.Start' is unused.	playerController.cs		Assets/Scripts/playerController.cs	Code Quality
●	☐	12	Private member 'playerController.Update' is unused.	playerController.cs		Assets/Scripts/playerController.cs	Code Quality
●	☐	12	Private member 'enemyController.Start' is unused.	enemyController.cs		Assets/Scripts/enemyController.cs	Code Quality
●	☐	17	Private member 'enemyController.Update' is unused.	enemyController.cs		Assets/Scripts/enemyController.cs	Code Quality
●	☐	10	Private member 'Gun.FireBullet' is unused.	Gun.cs		Assets/Scripts/Gun.cs	Code Quality
●	☐	17	Private member 'Gun.Start' is unused.	Gun.cs		Assets/Scripts/Gun.cs	Code Quality
●	☐	23	Private member 'Gun.Update' is unused.	Gun.cs		Assets/Scripts/Gun.cs	Code Quality
●	☐	12	Private member 'MouseLook.Start' is unused.	MouseLook.cs		Assets/Scripts/MouseLook.cs	Code Quality
●	☐	19	Private member 'MouseLook.Update' is unused.	MouseLook.cs		Assets/Scripts/MouseLook.cs	Code Quality

This result shows our code has many unused functions. Which is acceptable since we just started implementing our program. This analyzer is still new to us. We will need to do more research on its usage for different areas. Overall it is pretty good.

Assignment 3: Implementation & Verification

A. Dynamic Analysis

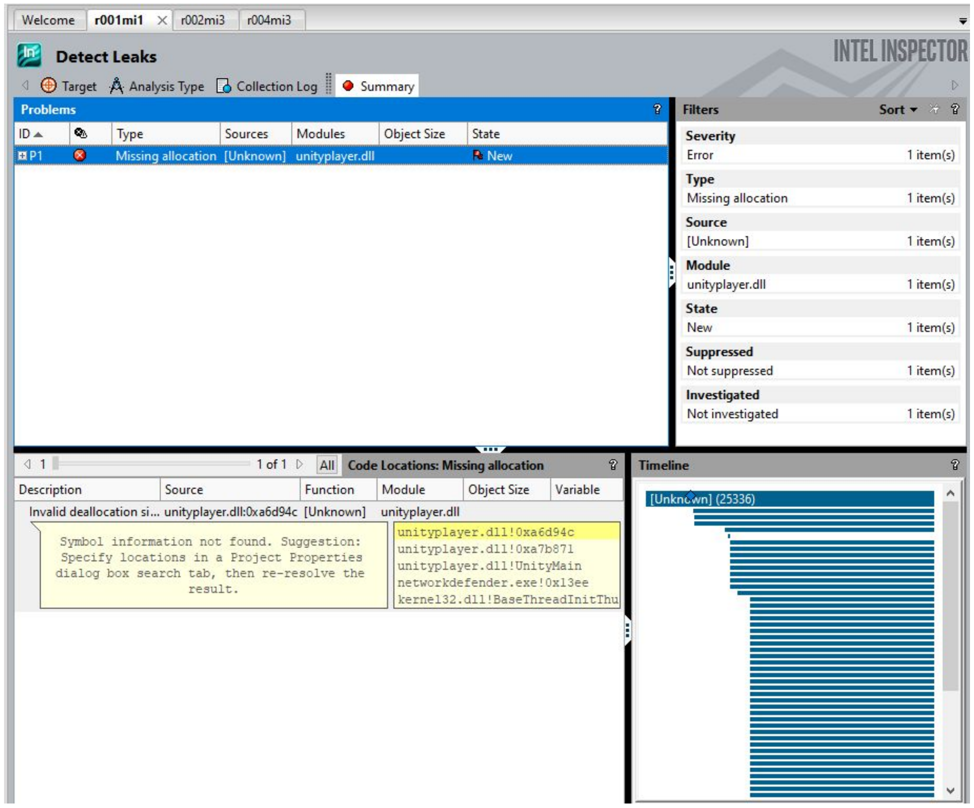
Tool: Intel Inspector

- Memory Analysis
    - Memory Leaks
    - Threading Issues
    - Deadlocks
    - Race Conditions
    - Redundant Cache Flushes
    - Persistent Memory Implementations
- Full GUI or IDE Integration
- Standalone usage
    - No special builds or commands needed

Usage Experience:

Simple to setup with standalone installer and easy project start and management. Analysis of the program is as simple as pointing Inspector to its location. Primary three options are to test for leaks, memory problems and deadlocks/races. Each test is run as its own report and saved under its own report code. Analysis with the tool takes additional CPU and Memory usage which slows down execution of the application being tested. Each report is broken up into several sections that outline what was found in an easy to read fashion.

Detect Memory Leaks



Problems

ID		Type	Sources	Modules	Object Size	State
P1		Missing allocation	[Unknown]	unityplayer.dll		New

Filters

Sort	
Severity	
Error	1 item(s)
Type	
Missing allocation	1 item(s)
Source	
[Unknown]	1 item(s)
Module	
unityplayer.dll	1 item(s)
State	
New	1 item(s)
Suppressed	
Not suppressed	1 item(s)
Investigated	
Not investigated	1 item(s)

1 1 of 1 All Code Locations: Missing allocation

Description	Source	Function	Module	Object Size	Variable
Invalid deallocation si...	unityplayer.dll:0xa6d94c	[Unknown]	unityplayer.dll		

```
Symbol information not found. Suggestion:
Specify locations in a Project Properties
dialog box search tab, then re-resolve the
result.
```

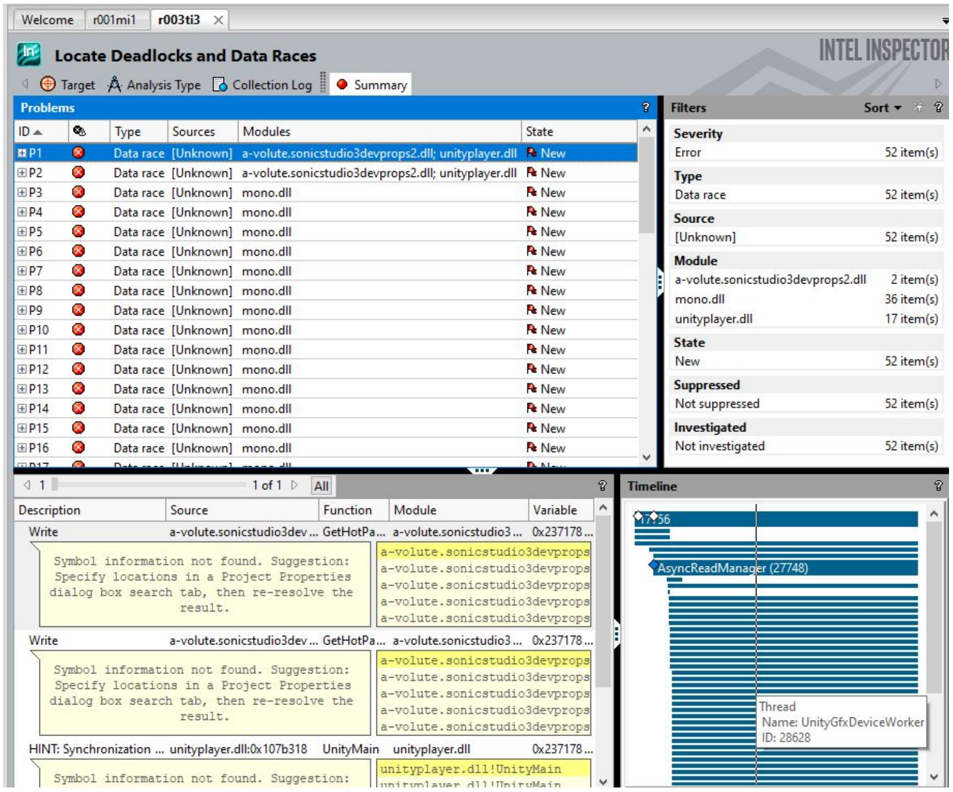
```
unityplayer.dll!0xa6d94c
unityplayer.dll!0xa7b871
unityplayer.dll!UnityMain
networkdefender.exe!0x13ee
kernel32.dll!BaseThreadInitThu
```

Timeline

[Unknown] (25336)

This report shows that our latest run has no real issues except for a missing allocation. However it specifies that the issue lies with unityplayer.dll which is not too helpful since a game built by Unity requires that .dll to run. The error could either be in our scripting and code or a non critical issue in Unity itself.

Memory Deadlocks/Races



Welcome | r001mi1 | r003ti3

Locate Deadlocks and Data Races

INTEL INSPECTOR

Target | Analysis Type | Collection Log | Summary

Problems

ID		Type	Sources	Modules	State
P1	⊗	Data race	[Unknown]	a-volute.sonicstudio3devprops2.dll; unityplayer.dll	New
P2	⊗	Data race	[Unknown]	a-volute.sonicstudio3devprops2.dll; unityplayer.dll	New
P3	⊗	Data race	[Unknown]	mono.dll	New
P4	⊗	Data race	[Unknown]	mono.dll	New
P5	⊗	Data race	[Unknown]	mono.dll	New
P6	⊗	Data race	[Unknown]	mono.dll	New
P7	⊗	Data race	[Unknown]	mono.dll	New
P8	⊗	Data race	[Unknown]	mono.dll	New
P9	⊗	Data race	[Unknown]	mono.dll	New
P10	⊗	Data race	[Unknown]	mono.dll	New
P11	⊗	Data race	[Unknown]	mono.dll	New
P12	⊗	Data race	[Unknown]	mono.dll	New
P13	⊗	Data race	[Unknown]	mono.dll	New
P14	⊗	Data race	[Unknown]	mono.dll	New
P15	⊗	Data race	[Unknown]	mono.dll	New
P16	⊗	Data race	[Unknown]	mono.dll	New

Filters Sort

Severity	
Error	52 item(s)
Type	
Data race	52 item(s)
Source	
[Unknown]	52 item(s)
Module	
a-volute.sonicstudio3devprops2.dll	2 item(s)
mono.dll	36 item(s)
unityplayer.dll	17 item(s)
State	
New	52 item(s)
Suppressed	
Not suppressed	52 item(s)
Investigated	
Not investigated	52 item(s)

1 | 1 of 1 | All

Description	Source	Function	Module	Variable
Write	a-volute.sonicstudio3dev ...	GetHotPa...	a-volute.sonicstudio3 ...	0x237178 ...
Write	a-volute.sonicstudio3dev ...	GetHotPa...	a-volute.sonicstudio3 ...	0x237178 ...
HINT: Synchronization ...	unityplayer.dll:0x107b318	UnityMain	unityplayer.dll	0x237178 ...

```
Symbol information not found. Suggestion:
Specify locations in a Project Properties
dialog box search tab, then re-resolve the
result.
```

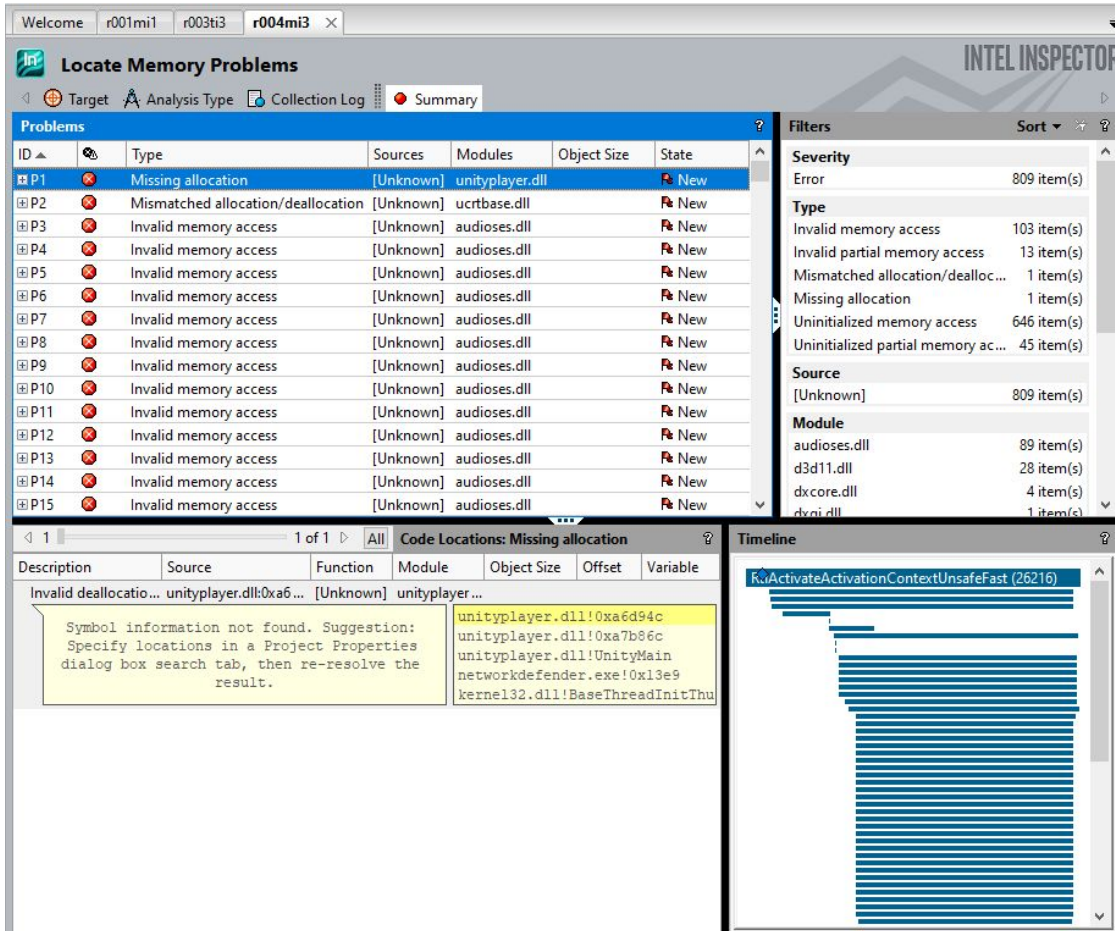
```
a-volute.sonicstudio3devprops
a-volute.sonicstudio3devprops
a-volute.sonicstudio3devprops
a-volute.sonicstudio3devprops
a-volute.sonicstudio3devprops
```

```
unityplayer.dll!UnityMain
```

Timeline

This check for deadlocks/races showed multiple data races dealing with unityplayer.dll and mono.dll. Sources unknown error is from one or more threads in our application accessing the stack of another thread. The Inspector can be set to watch for data races on stack accesses and then run again which we will do.

Memory Problems


Welcome | r001mi1 | r003ti3 | r004mi3

Locate Memory Problems

Target | Analysis Type | Collection Log | Summary

Problems

ID		Type	Sources	Modules	Object Size	State
P1	⊗	Missing allocation	[Unknown]	unityplayer.dll		New
P2	⊗	Mismatched allocation/deallocation	[Unknown]	ucrtbase.dll		New
P3	⊗	Invalid memory access	[Unknown]	audioses.dll		New
P4	⊗	Invalid memory access	[Unknown]	audioses.dll		New
P5	⊗	Invalid memory access	[Unknown]	audioses.dll		New
P6	⊗	Invalid memory access	[Unknown]	audioses.dll		New
P7	⊗	Invalid memory access	[Unknown]	audioses.dll		New
P8	⊗	Invalid memory access	[Unknown]	audioses.dll		New
P9	⊗	Invalid memory access	[Unknown]	audioses.dll		New
P10	⊗	Invalid memory access	[Unknown]	audioses.dll		New
P11	⊗	Invalid memory access	[Unknown]	audioses.dll		New
P12	⊗	Invalid memory access	[Unknown]	audioses.dll		New
P13	⊗	Invalid memory access	[Unknown]	audioses.dll		New
P14	⊗	Invalid memory access	[Unknown]	audioses.dll		New
P15	⊗	Invalid memory access	[Unknown]	audioses.dll		New

Filters

Severity	
Error	809 item(s)
Type	
Invalid memory access	103 item(s)
Invalid partial memory access	13 item(s)
Mismatched allocation/dealloc...	1 item(s)
Missing allocation	1 item(s)
Uninitialized memory access	646 item(s)
Uninitialized partial memory ac...	45 item(s)
Source	
[Unknown]	809 item(s)
Module	
audioses.dll	89 item(s)
d3d11.dll	28 item(s)
dxcore.dll	4 item(s)

1 of 1 | All | Code Locations: Missing allocation

Description	Source	Function	Module	Object Size	Offset	Variable
Invalid deallocatio...	unityplayer.dll:0xa6...	[Unknown]	unityplayer...			

```
Symbol information not found. Suggestion:
Specify locations in a Project Properties
dialog box search tab, then re-resolve the
result.
```

```
unityplayer.dll!0xa6d94c
unityplayer.dll!0xa7b86c
unityplayer.dll!UnityMain
networkdefender.exe!0x13e9
kernel32.dll!BaseThreadInitThu
```

Timeline

RtlActivateActivationContextUnsafeFast (26216)

Shows multiple invalid memory accesses from multiple library files. Some might be issues with Unity itself.

While easy to use with easy to read results, the primary problem with this tool is that it shows what is going on at an execution level and pinpoints libraries that seem to be the cause of the problem. Most other tools will do the same more than likely and the root issue there is probably how the work is being done in the Unity SDK and the fact that their library functions are being used. This makes narrowing down any issues difficult as it has to be determined if the issue is native to Unity or if it has been introduced by bad practices or code in the project itself. One easy way to narrow this down and curtail current and future issues is to engage in good coding practices as a team with unified formatting and naming.

Then code reviews to eliminate bloat in the form of unused variables, functions, legacy code and depreciated code. This tool while non-specific in how it shows vulnerabilities for our project is a success in terms of highlighting how important it is to be aware of limitations when working with SDKs and their function libraries and how to work around them and plan for them in making secure and quality software. Our group will have to compile a listing of known issues with Unity so further dynamic testing can be narrowed down to issues of our own creation.

B. Attack Surface Review

There has not been any major development changes or security patches in any of our approved tools. As a result, no new attack vectors have been created that may require review and mitigation including threat models.

Here is a list of trivial updates to our approved tools:

Mac OS Requirements

Compiler/ Tool	Old Version	Current Version	Notes
Visual Studio 2019 for Mac	8.4.6	8.4.8	Addresses some accessibility issues.
GIMP	2.10.14	2.10.18	Usability improvements and bug fixes.

Win64 Requirements

Compiler/ Tool	Old Version	Current Version	Notes

Visual Studio 2019	16.4.5	16.4.6	Performance improvements and bug fixes.
GIMP	2.10.14	2.10.18	Usability improvements and bug fixes.

Assignment 4: Verification

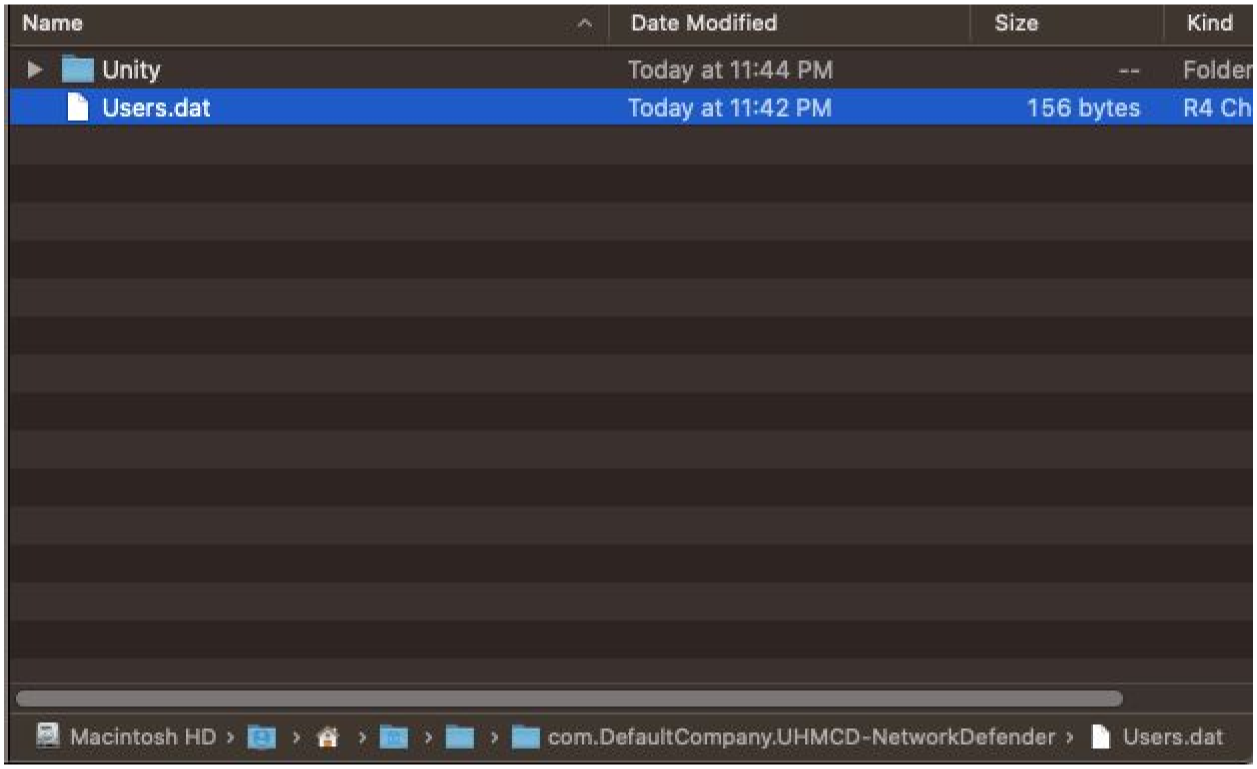
A. Fuzz Testing

Base Steps for All Attacks

Step 1: Locate the Users.dat file.

Currently, the Users.dat file is stored within the file system of the application. This means that a user has potential to find and manipulate the file. The first step was to locate the file.

Note: File location is OS dependent due path command Application.persistentDataPath used. This location is MacOS specific. Windows 10 location is: C:\Users\UserName\AppData\LocalLow\DefaultCompany\UnityProjectName



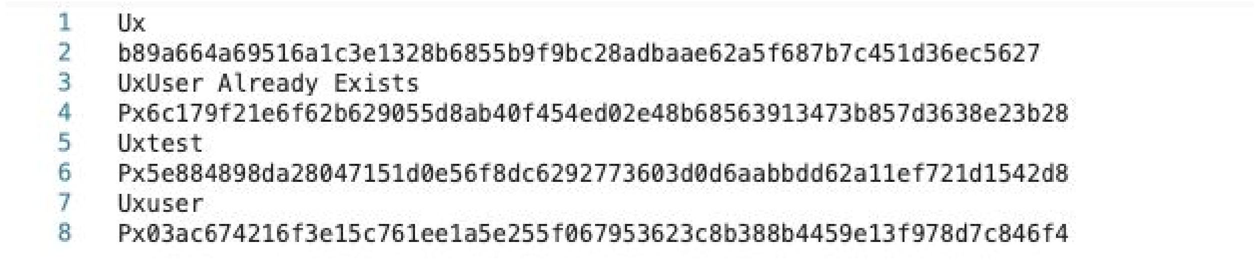
Name	Date Modified	Size	Kind
Unity	Today at 11:44 PM	--	Folder
Users.dat	Today at 11:42 PM	156 bytes	R4 Ch

Macintosh HD › › › › › com.DefaultCompany.UHMCD-NetworkDefender › Users.dat

The file sits in the base directory of the application, so it is not hard to find.

Step 2: Open Users.dat with an Editor

I used Visual Studio in this instance.

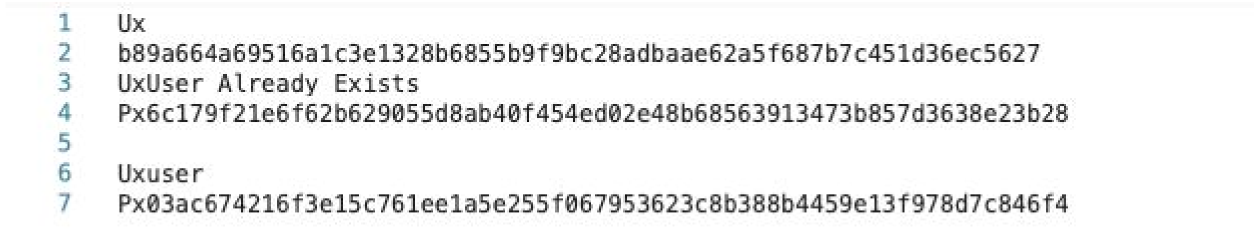


```
Ux
b89a664a69516a1c3e1328b6855b9f9bc28adbaae62a5f687b7c451d36ec5627
UxUser Already Exists
Px6c179f21e6f62b629055d8ab40f454ed02e48b68563913473b857d3638e23b28
Uxtest
Px5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8
Uxuser
Px03ac674216f3e15c761ee1a5e255f067953623c8b388b4459e13f978d7c846f4
```

The contents of the file are as follows. As someone who worked on the project, I know how the file is structured. However, it would not be hard for someone to guess that Ux prefixes the users and Px prefixes the respective password. This is important for later. As a note, Ux and Px are holdovers from the first implementation of the user account creation and are now deprecated and can be removed to increase security. Now I will cover the possible attacks using this vector.

Deleting a User's Account/Invalidating Their Login

This one is by far the simplest. In order to delete a user's account, you simply need to delete the lines that contain their information.



```
Ux
b89a664a69516a1c3e1328b6855b9f9bc28adbaae62a5f687b7c451d36ec5627
UxUser Already Exists
Px6c179f21e6f62b629055d8ab40f454ed02e48b68563913473b857d3638e23b28

Uxuser
Px03ac674216f3e15c761ee1a5e255f067953623c8b388b4459e13f978d7c846f4
```

In this screenshot, I deleted the account that belongs to 'test'. They would now need to create an account again. The attacker could also use this to steal the other player's name.

Taking Over Another User's Account

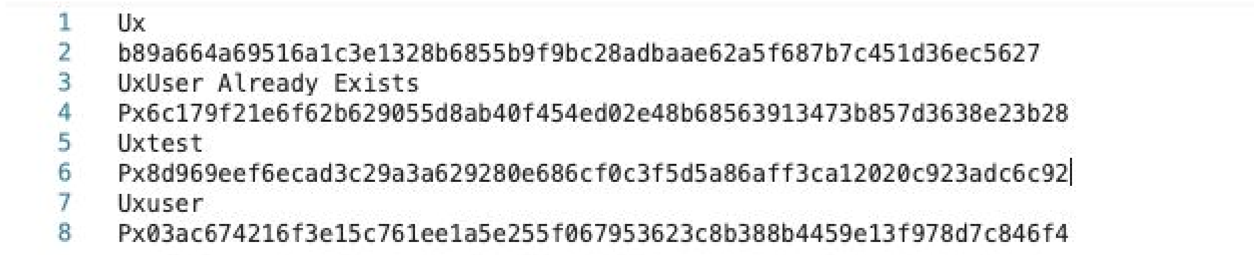
In this situation, the attacker requires a little bit more knowledge of the system to take over an account with deleting it first. Using inspection, they could determine that a SHA-256 hashing algorithm is used to protect user's passwords. Therefore, we can also try to use the same hash algorithm to generate a hash for a known password and swap it with the other user's password.

For instance the SHA-256 hash of '123456' is

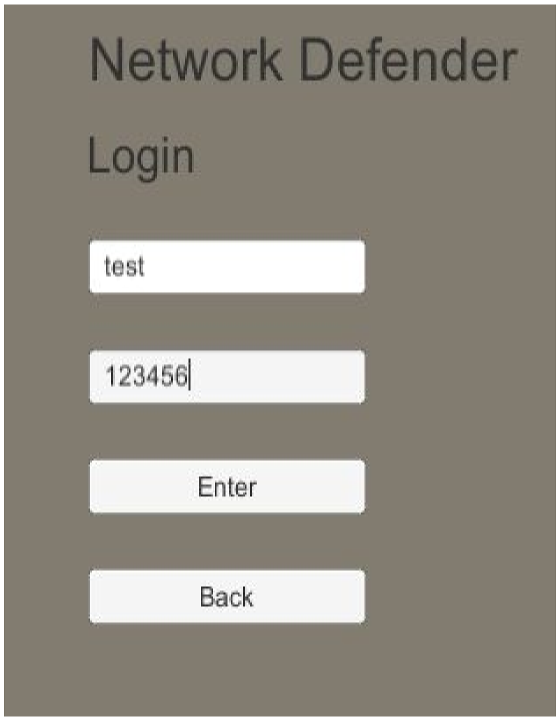
8d969eef6ecad3c29a3a629280e686cf0c3f5d5a86aff3ca12020c923adc6c92.

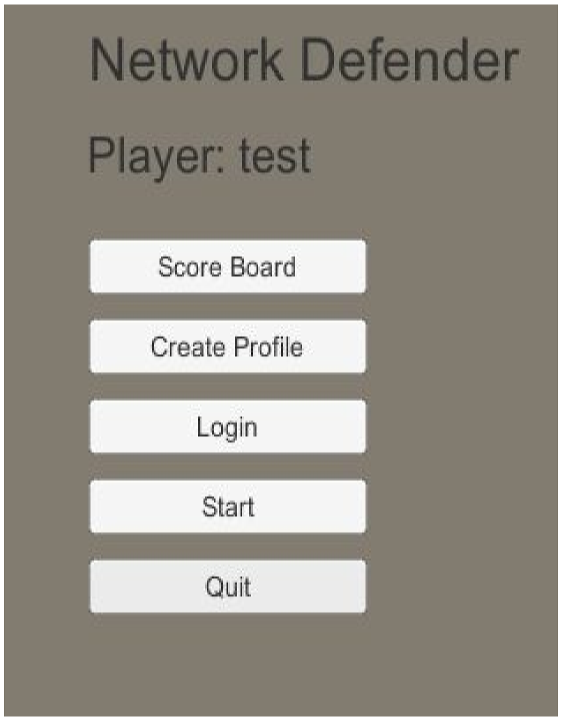
By replacing

'5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8' with my hashed password, I now have effectively changed the test user's password.



```
Ux
b89a664a69516a1c3e1328b6855b9f9bc28adbaae62a5f687b7c451d36ec5627
UxUser Already Exists
Px6c179f21e6f62b629055d8ab40f454ed02e48b68563913473b857d3638e23b28
Uxtest
Px8d969eef6ecad3c29a3a629280e686cf0c3f5d5a86aff3ca12020c923adc6c92
Uxuser
Px03ac674216f3e15c761ee1a5e255f067953623c8b388b4459e13f978d7c846f4
```





Discovering Another User's Password

This attack builds off of the last and is both the largest security threat and the hardest to fix. Once the attacker is able to pull the SHA-256 password hash from the Users.dat file, they can begin to brute force to find the password. Using a tool such as HashCat, they can specify the target hash and a wordlist to use when brute forcing the password. A weak password will be found quickly.

By including salt in front of the plaintext before the password is hashed, the difficulty of brute forcing the password increases exponentially, while also removing the case of a table lookup for common passwords (assuming a unique salt). This is part of the next steps in this implementation.

Unity Specific Attack Possibility

One possible attack vector recently thought of, but unexplored due to complexity is how Unity compiles and stores developer made C# or java scripts. The possibility to find how these scripts are compiled and stored in a finished product and if they can be reverse engineered to replace them with, or add on malicious code should be considered. Currently we will not pursue this due to time constraints.

Conclusion:

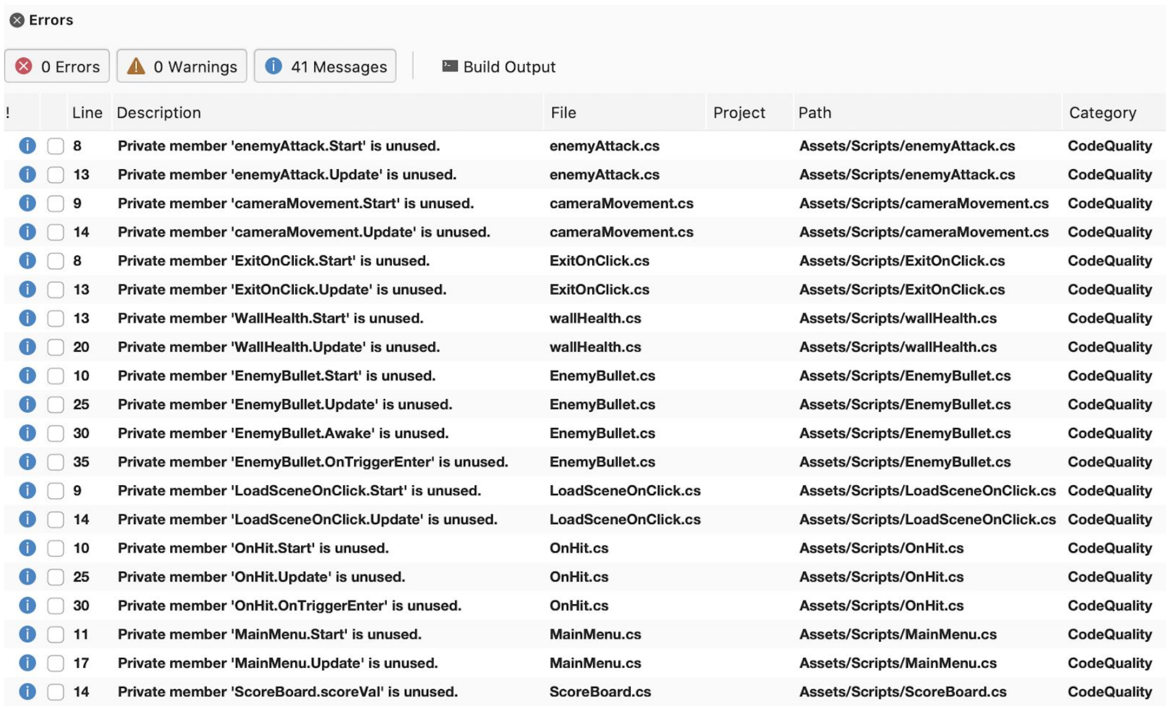
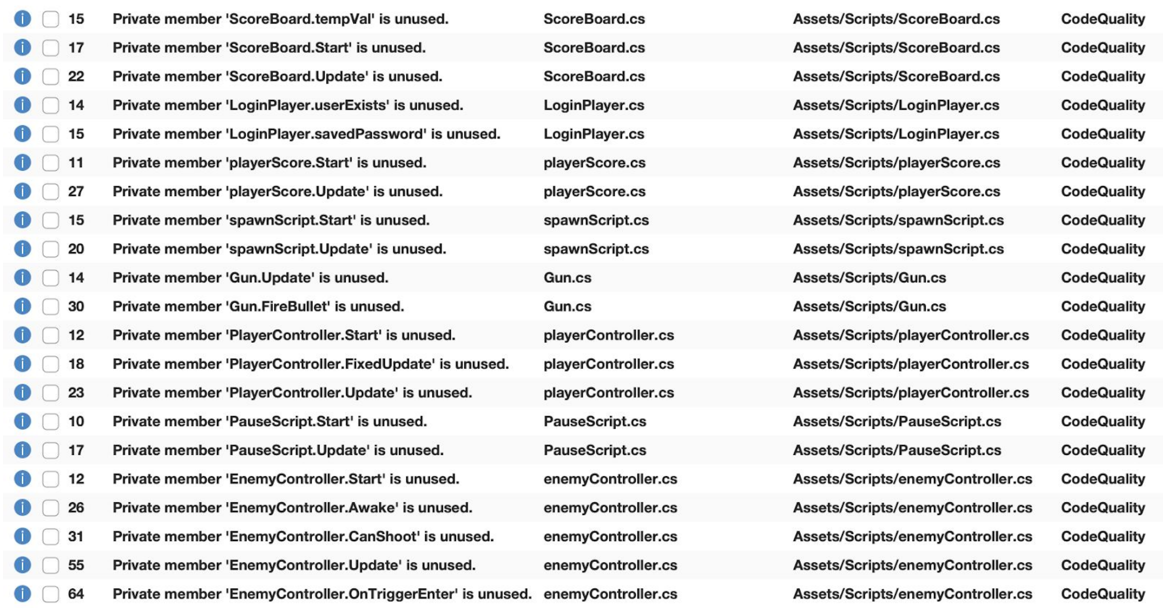
Since this game is run entirely on the client, there is very little that can be done to prevent these types of attacks from occurring. However, it is possible to make it more difficult. Removal of deprecated Ux/Px designators and hashing the usernames as well will increase security. Or, encrypting the Users.dat file would force them to retrieve and edit the information while the application is running. Even if we were able to stop an attacker from editing the data, there would still be a possibility of pulling the data from memory and finding the hashed passwords. Assuming the passwords are salted, it would be very difficult to find a password, but not impossible. Options to stop deleting data include adding in a file validation system that tracks current file vs last known version. This also means adding in a file backup system so as not to lose all other user data.

Other Notes:

Issue: A user is able to make an account that has 0 characters for a username/password.
Fix: Implement minimum username/password lengths.
Issue: A user is able to make an account that has unlimited characters for a username/password. This could lead to a crash/slow down by filling up all of the computer's memory.
Fix: Implement maximum username/password lengths.
Issue: Usernames and passwords have no restrictions for special characters.
Fix: Limit the characters allowed for readability of usernames.

Text fields are always treated as a string which prevents injection attacks. I thought the new line '\n' string may be able to ruin the structure of the file. Other special characters and character codes are all treated as text. In addition, there is no paste functionality, which makes injection more difficult.

B. Static Analysis Review

Errors

0 Errors | 0 Warnings | 41 Messages | Build Output

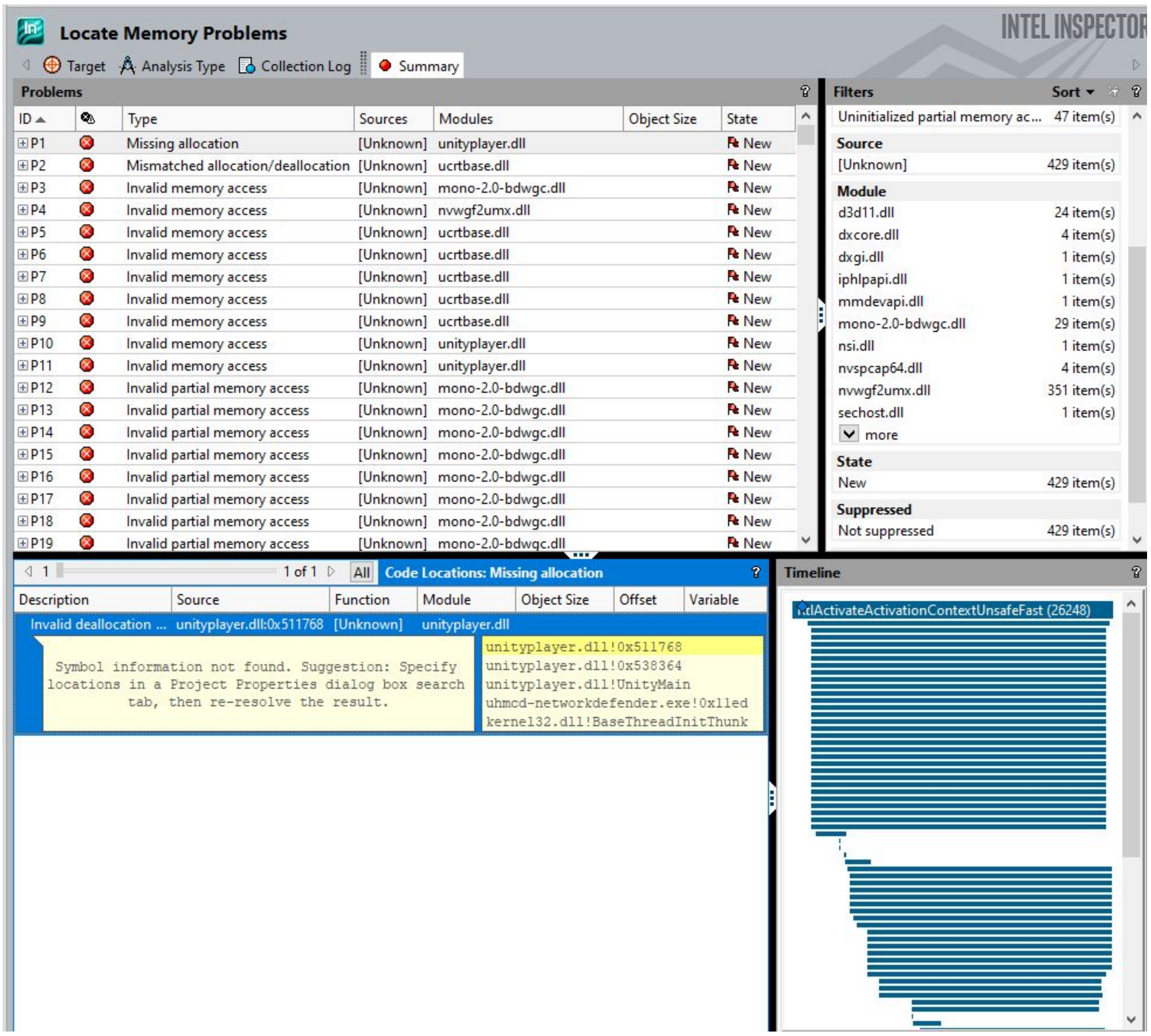
!		Line	Description	File	Project	Path	Category
ℹ	☐	8	Private member 'enemyAttack.Start' is unused.	enemyAttack.cs		Assets/Scripts/enemyAttack.cs	CodeQuality
ℹ	☐	13	Private member 'enemyAttack.Update' is unused.	enemyAttack.cs		Assets/Scripts/enemyAttack.cs	CodeQuality
ℹ	☐	9	Private member 'cameraMovement.Start' is unused.	cameraMovement.cs		Assets/Scripts/cameraMovement.cs	CodeQuality
ℹ	☐	14	Private member 'cameraMovement.Update' is unused.	cameraMovement.cs		Assets/Scripts/cameraMovement.cs	CodeQuality
ℹ	☐	8	Private member 'ExitOnClick.Start' is unused.	ExitOnClick.cs		Assets/Scripts/ExitOnClick.cs	CodeQuality
ℹ	☐	13	Private member 'ExitOnClick.Update' is unused.	ExitOnClick.cs		Assets/Scripts/ExitOnClick.cs	CodeQuality
ℹ	☐	13	Private member 'WallHealth.Start' is unused.	wallHealth.cs		Assets/Scripts/wallHealth.cs	CodeQuality
ℹ	☐	20	Private member 'WallHealth.Update' is unused.	wallHealth.cs		Assets/Scripts/wallHealth.cs	CodeQuality
ℹ	☐	10	Private member 'EnemyBullet.Start' is unused.	EnemyBullet.cs		Assets/Scripts/EnemyBullet.cs	CodeQuality
ℹ	☐	25	Private member 'EnemyBullet.Update' is unused.	EnemyBullet.cs		Assets/Scripts/EnemyBullet.cs	CodeQuality
ℹ	☐	30	Private member 'EnemyBullet.Awake' is unused.	EnemyBullet.cs		Assets/Scripts/EnemyBullet.cs	CodeQuality
ℹ	☐	35	Private member 'EnemyBullet.OnTriggerEnter' is unused.	EnemyBullet.cs		Assets/Scripts/EnemyBullet.cs	CodeQuality
ℹ	☐	9	Private member 'LoadSceneOnClick.Start' is unused.	LoadSceneOnClick.cs		Assets/Scripts/LoadSceneOnClick.cs	CodeQuality
ℹ	☐	14	Private member 'LoadSceneOnClick.Update' is unused.	LoadSceneOnClick.cs		Assets/Scripts/LoadSceneOnClick.cs	CodeQuality
ℹ	☐	10	Private member 'OnHit.Start' is unused.	OnHit.cs		Assets/Scripts/OnHit.cs	CodeQuality
ℹ	☐	25	Private member 'OnHit.Update' is unused.	OnHit.cs		Assets/Scripts/OnHit.cs	CodeQuality
ℹ	☐	30	Private member 'OnHit.OnTriggerEnter' is unused.	OnHit.cs		Assets/Scripts/OnHit.cs	CodeQuality
ℹ	☐	11	Private member 'MainMenu.Start' is unused.	MainMenu.cs		Assets/Scripts/MainMenu.cs	CodeQuality
ℹ	☐	17	Private member 'MainMenu.Update' is unused.	MainMenu.cs		Assets/Scripts/MainMenu.cs	CodeQuality
ℹ	☐	14	Private member 'ScoreBoard.scoreVal' is unused.	ScoreBoard.cs		Assets/Scripts/ScoreBoard.cs	CodeQuality
ℹ	☐	15	Private member 'ScoreBoard.tempVal' is unused.	ScoreBoard.cs		Assets/Scripts/ScoreBoard.cs	CodeQuality
ℹ	☐	17	Private member 'ScoreBoard.Start' is unused.	ScoreBoard.cs		Assets/Scripts/ScoreBoard.cs	CodeQuality
ℹ	☐	22	Private member 'ScoreBoard.Update' is unused.	ScoreBoard.cs		Assets/Scripts/ScoreBoard.cs	CodeQuality
ℹ	☐	14	Private member 'LoginPlayer.userExists' is unused.	LoginPlayer.cs		Assets/Scripts/LoginPlayer.cs	CodeQuality
ℹ	☐	15	Private member 'LoginPlayer.savedPassword' is unused.	LoginPlayer.cs		Assets/Scripts/LoginPlayer.cs	CodeQuality
ℹ	☐	11	Private member 'playerScore.Start' is unused.	playerScore.cs		Assets/Scripts/playerScore.cs	CodeQuality
ℹ	☐	27	Private member 'playerScore.Update' is unused.	playerScore.cs		Assets/Scripts/playerScore.cs	CodeQuality
ℹ	☐	15	Private member 'spawnScript.Start' is unused.	spawnScript.cs		Assets/Scripts/spawnScript.cs	CodeQuality
ℹ	☐	20	Private member 'spawnScript.Update' is unused.	spawnScript.cs		Assets/Scripts/spawnScript.cs	CodeQuality
ℹ	☐	14	Private member 'Gun.Update' is unused.	Gun.cs		Assets/Scripts/Gun.cs	CodeQuality
ℹ	☐	30	Private member 'Gun.FireBullet' is unused.	Gun.cs		Assets/Scripts/Gun.cs	CodeQuality
ℹ	☐	12	Private member 'PlayerController.Start' is unused.	playerController.cs		Assets/Scripts/playerController.cs	CodeQuality
ℹ	☐	18	Private member 'PlayerController.FixedUpdate' is unused.	playerController.cs		Assets/Scripts/playerController.cs	CodeQuality
ℹ	☐	23	Private member 'PlayerController.Update' is unused.	playerController.cs		Assets/Scripts/playerController.cs	CodeQuality
ℹ	☐	10	Private member 'PauseScript.Start' is unused.	PauseScript.cs		Assets/Scripts/PauseScript.cs	CodeQuality
ℹ	☐	17	Private member 'PauseScript.Update' is unused.	PauseScript.cs		Assets/Scripts/PauseScript.cs	CodeQuality
ℹ	☐	12	Private member 'EnemyController.Start' is unused.	enemyController.cs		Assets/Scripts/enemyController.cs	CodeQuality
ℹ	☐	26	Private member 'EnemyController.Awake' is unused.	enemyController.cs		Assets/Scripts/enemyController.cs	CodeQuality
ℹ	☐	31	Private member 'EnemyController.CanShoot' is unused.	enemyController.cs		Assets/Scripts/enemyController.cs	CodeQuality
ℹ	☐	55	Private member 'EnemyController.Update' is unused.	enemyController.cs		Assets/Scripts/enemyController.cs	CodeQuality
ℹ	☐	64	Private member 'EnemyController.OnTriggerEnter' is unused.	enemyController.cs		Assets/Scripts/enemyController.cs	CodeQuality

Summary: The static analysis results show 41 code quality errors where class methods were determined to be unused. No vulnerabilities or insecure coding practices were discovered. Overall, the status of our project (in regards to static analysis) remains unchanged since the last static analysis review.

Unity SDK Note: Some class methods that show up as unused are invoked using the built in Unity methods Invoke(string methodName) or InvokeRepeating(string methodName) and since are not called directly will show up as unused. Other unused methods are physics/trigger detection methods that wait for events and are unused until that event is detected by the game engine during runtime. Unused default methods (Start(), Update()) in scripts will be removed during code cleanup.

C. Dynamic Review

Locate Memory Problems



Locate Memory Problems

Problems

ID		Type	Sources	Modules	Object Size	State
P1	●	Missing allocation	[Unknown]	unityplayer.dll		New
P2	●	Mismatched allocation/deallocation	[Unknown]	ucrtbase.dll		New
P3	●	Invalid memory access	[Unknown]	mono-2.0-bdwgc.dll		New
P4	●	Invalid memory access	[Unknown]	nvwgf2umx.dll		New
P5	●	Invalid memory access	[Unknown]	ucrtbase.dll		New
P6	●	Invalid memory access	[Unknown]	ucrtbase.dll		New
P7	●	Invalid memory access	[Unknown]	ucrtbase.dll		New
P8	●	Invalid memory access	[Unknown]	ucrtbase.dll		New
P9	●	Invalid memory access	[Unknown]	ucrtbase.dll		New
P10	●	Invalid memory access	[Unknown]	unityplayer.dll		New
P11	●	Invalid memory access	[Unknown]	unityplayer.dll		New
P12	●	Invalid partial memory access	[Unknown]	mono-2.0-bdwgc.dll		New
P13	●	Invalid partial memory access	[Unknown]	mono-2.0-bdwgc.dll		New
P14	●	Invalid partial memory access	[Unknown]	mono-2.0-bdwgc.dll		New
P15	●	Invalid partial memory access	[Unknown]	mono-2.0-bdwgc.dll		New
P16	●	Invalid partial memory access	[Unknown]	mono-2.0-bdwgc.dll		New
P17	●	Invalid partial memory access	[Unknown]	mono-2.0-bdwgc.dll		New
P18	●	Invalid partial memory access	[Unknown]	mono-2.0-bdwgc.dll		New
P19	●	Invalid partial memory access	[Unknown]	mono-2.0-bdwgc.dll		New

Filters

Uninitialized partial memory ac...	47 item(s)
Source	
[Unknown]	429 item(s)
Module	
d3d11.dll	24 item(s)
dxcore.dll	4 item(s)
dxgi.dll	1 item(s)
iphlpapi.dll	1 item(s)
mmdevapi.dll	1 item(s)
mono-2.0-bdwgc.dll	29 item(s)
nsi.dll	1 item(s)
nvspcap64.dll	4 item(s)
nvwgf2umx.dll	351 item(s)
sechost.dll	1 item(s)
more	
State	
New	429 item(s)
Suppressed	
Not suppressed	429 item(s)

Code Locations: Missing allocation

Description	Source	Function	Module	Object Size	Offset	Variable
Invalid deallocation ...	unityplayer.dll:0x511768	[Unknown]	unityplayer.dll			

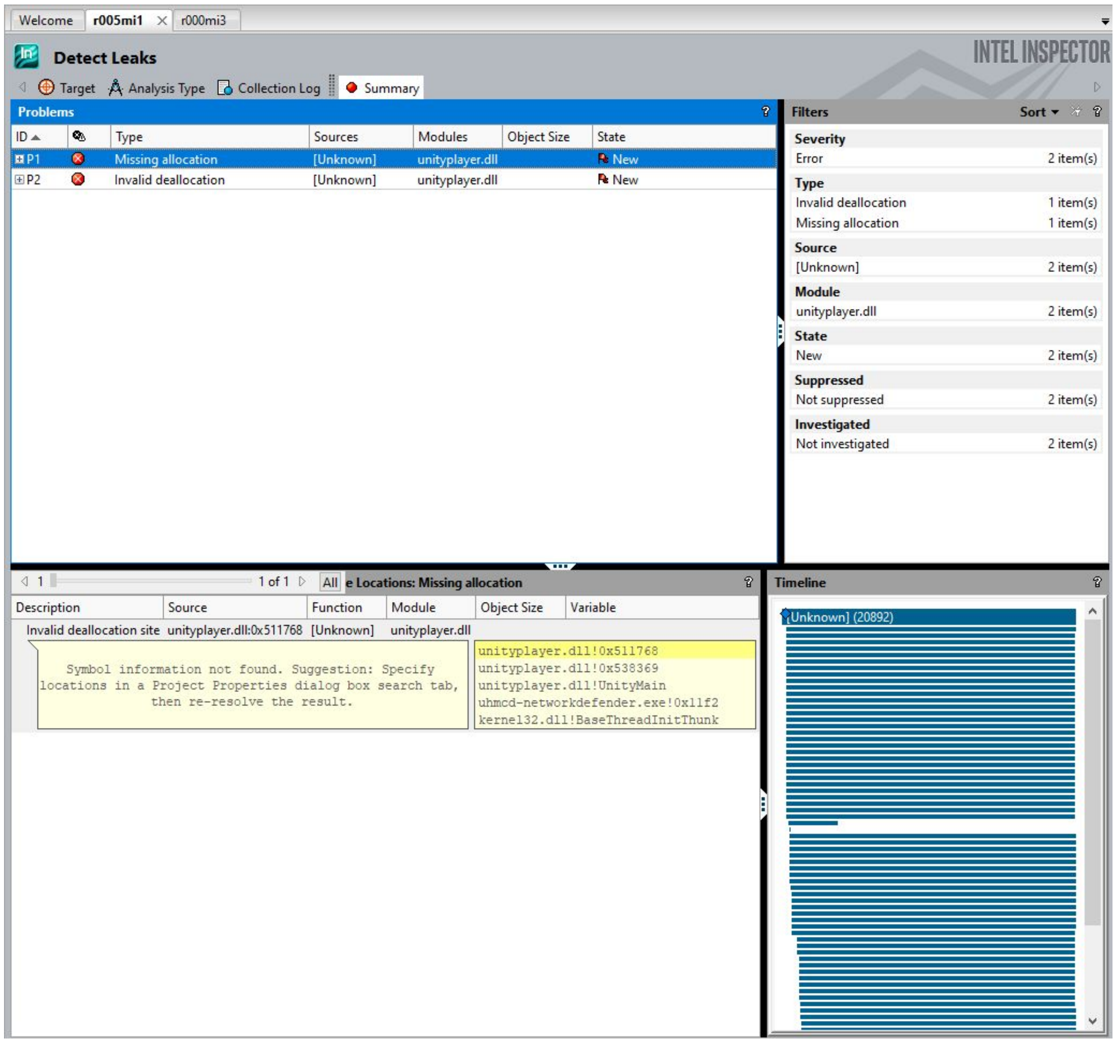
```
Symbol information not found. Suggestion: Specify
locations in a Project Properties dialog box search
tab, then re-resolve the result.
```

```
unityplayer.dll!0x511768
unityplayer.dll!0x538364
unityplayer.dll!UnityMain
uhmcd-networkdefender.exe!0x11ed
kernel32.dll!BaseThreadInitThunk
```

Timeline

Intel Inspector returned 429 items for Locating Memory Problems. Most references are built in Unity libraries or graphics libraries called by the system. Invalid access or missing/mismatched allocations and could be tied to empty methods or other errors in our C# scripts. Once code cleanup is done, another test will be done to see if any of the issues are cleared up.

Locate Memory Leaks



Problems

ID		Type	Sources	Modules	Object Size	State
P1	●	Missing allocation	[Unknown]	unityplayer.dll		New
P2	●	Invalid deallocation	[Unknown]	unityplayer.dll		New

Filters

Severity	
Error	2 item(s)
Type	
Invalid deallocation	1 item(s)
Missing allocation	1 item(s)
Source	
[Unknown]	2 item(s)
Module	
unityplayer.dll	2 item(s)
State	
New	2 item(s)
Suppressed	
Not suppressed	2 item(s)
Investigated	
Not investigated	2 item(s)

1 of 1 All e Locations: Missing allocation

Description	Source	Function	Module	Object Size	Variable
Invalid deallocation site	unityplayer.dll:0x511768	[Unknown]	unityplayer.dll		

```
Symbol information not found. Suggestion: Specify
locations in a Project Properties dialog box search tab,
then re-resolve the result.
```

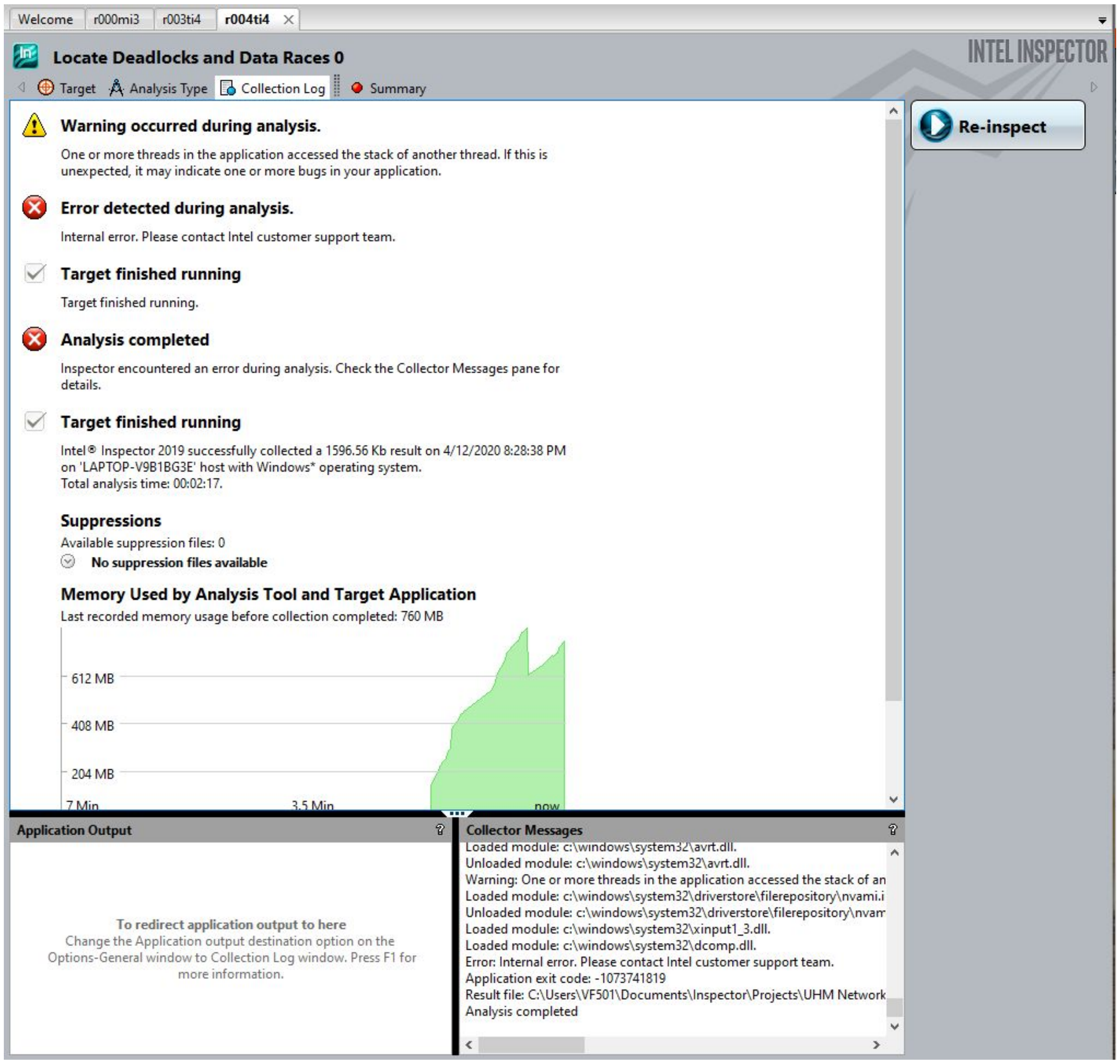
```
unityplayer.dll!0x511768
unityplayer.dll!0x538369
unityplayer.dll!UnityMain
uhmcd-networkdefender.exe!0x11f2
kernel32.dll!BaseThreadInitThunk
```

Timeline

[Unknown] (20892)

Narrowed scope of memory issue detection to just memory leaks. Shows only a missing allocation and a invalid deallocation from unityplayer.dll. Most likely an error within Unity itself as our C# scripts contain no instances of improper allocation or deallocation in accordance with the syntax rules of C#.

Locate Deadlocks and Races



Locate Deadlocks and Data Races 0

Warning occurred during analysis.
One or more threads in the application accessed the stack of another thread. If this is unexpected, it may indicate one or more bugs in your application.

Error detected during analysis.
Internal error. Please contact Intel customer support team.

Target finished running
Target finished running.

Analysis completed
Inspector encountered an error during analysis. Check the Collector Messages pane for details.

Target finished running
Intel® Inspector 2019 successfully collected a 1596.56 Kb result on 4/12/2020 8:28:38 PM on 'LAPTOP-V9B1BG3E' host with Windows* operating system.
Total analysis time: 00:02:17.

Suppressions
Available suppression files: 0
No suppression files available

Memory Used by Analysis Tool and Target Application
Last recorded memory usage before collection completed: 760 MB

Application Output

To redirect application output to here
Change the Application output destination option on the Options-General window to Collection Log window. Press F1 for more information.

Collector Messages

```
Loaded module: c:\windows\system32\avrt.dll.
Unloaded module: c:\windows\system32\avrt.dll.
Warning: One or more threads in the application accessed the stack of an
Loaded module: c:\windows\system32\driverstore\filerepository\nvami.i
Unloaded module: c:\windows\system32\driverstore\filerepository\nvam
Loaded module: c:\windows\system32\xinput1_3.dll.
Loaded module: c:\windows\system32\dcomp.dll.
Error: Internal error. Please contact Intel customer support team.
Application exit code: -1073741819
Result file: C:\Users\VF501\Documents\Inspector\Projects\UHM Network
Analysis completed
```

Intel Inspector repeatedly suffered an internal error and could not complete analysis. Issue seems to occur with loading the direct composition library dcomp.dll. Unable to determine why this error is occurring and cannot determine a fix. Testing Intel Inspector on another installed program did not have this issue, will try to determine what changes between the last testing segment and now could have caused this. Like last time the same warning cites that one or more threads accessed the stack of another thread. This warning can be ignored as

Unity is multithreaded and it occurs with inspection of other applications which also have multithreading.

Assignment 5: Release

Incident Response Plan

The purpose of the Incident Response Plan is to define a plan for handling privacy escalations with a systematic and timely manner. As the current state of the application is entirely run on the client's machine, a breach of privacy would only come from physical access to the device and manipulation of the file system. However, this plan was created proactively for when a future update that implements a server based login and scoreboard system. This is case, it would be critical to prevent unauthorized manipulation/deletion of scores and their associated users as well as secure the passwords of the users on our system.

<u>Privacy Escalation Team</u>

Escalation manager - Pauline Wu

The role of the escalation manager is to include the appropriate representation across the organization and be the driving force of the process. The process of privacy escalation response begins and ends with the escalation manager.

Legal representative - Kenneth Yamaguchi-Townsend

The role of the legal representative is to help resolve any legal concerns that may arise. This may include, but is not limited to, lawsuits or settlements from affected users, the legality of process as a whole, and any other law related complications.

Public relations - Joshua Ducey

The role of the legal representative is to effectively communicate the situation to the users of the application and the public. This may include, but is not limited too, public Relations outreach, online help articles, breach notification, documentation updates, and short-term and/or long-term product or service changes.

Security engineer - Michael Boyle

The role of the security engineer is to discover the vulnerability that lead to the privacy escalation incident and fix the issue within our application/network/organization. When the vulnerability is both detected and fixed, a report will be delivered to the escalation manager and then distributed to the appropriate parties.

Contact Us

In the case of an emergency, UHM Cyber Defense can be reached by email at:

UHMCyberDefense@gmail.com

Procedures for Incident Response

The procedure begins as soon as the problem is detected and the escalation manager is notified. The details of the incident are then reviewed by the escalation manager and a determination is made regarding whether more information is required and requires attention. If further steps are required, then the following steps will be taken:

- Determine the source(s) of the incident
- Identify the impact and scope of the incident
- Determine the validity of the incident
- Create a summary of the known facts
- Layout a timeline with expectations
- List employees that know about the affected system

With this information compiled, the escalation manager should contact the appropriate people and seek resolution of the incident. Depending on the scope and severity of the incident, the escalation manager can assign portions of the work to others. However, it is the escalation manager's responsibility to ensure the prompt resolution of the matter.

Once the appropriate resolutions are in place, an audit of the effectiveness of the actions should be taken and recorded. The documentation of the incident will be used to adjust this procedure and the steps within it.

Final Security Review

After running different tests and analysis on this program, as a result it passed FSR (with exceptions). This program is a game, so it would be hard for the user to access the code. In addition, we have solved many issues with the account creating and password setting which decreased the chance of getting attacked by password . As you can see, when conducting the tests and analysis, some issues arise for the allocation and memory store. These could not be solved, and some of the issues require deleting essential files, which is a risk since we are not sure what really causes the issues. These issues may be caused by Unity itself. We would know until a newer version of Unity comes out and when the program is modified to fit the newer version. Then we could do the tests and analysis again. And hopefully these issues will be solved/fixed.

Certified Release & Archive Report

Link to release version: https://github.com/UHM-Cyber-Defense/UHMCyberDefense/releases

Version number: 1.0

Summary of features

- Our application is a fixed-shooter game built on the Unity game engine. It allows for users to be able to create and log in to their own account. The game also features a scoreboard that displays the top 10 hi-scores. In regards to security, it currently encrypts the data file that contains usernames and passwords. User passwords are also stored and compared using an SHA-256 cryptographic hashing algorithm.

Future development plans

- We want to increase replayability by adding more levels. New levels would have different terrain and perhaps different enemy tank formations.
- We also want the user to be able to select a difficulty level. Different difficulty levels would either increase or decrease fire rates, movement speeds and health bars.
- Split scoreboards for different difficulties
- Moar enemy types pls.
- Power Ups
- Alternate Weapons
- Online Scoreboards
- Additional Testing of features to improve game balance and fun.

Installation

- Windows: Double Click on UHMCD_Setup.exe. Follow installer directions.
- Mac: Double click on Network.Defender.dmg and move Network Defender.app into your Applications folder.

Technical Notes

- Security
  - User generated data (user name, password, scores) are stored in AES encrypted files.
  - AES key is removed from system memory once the program no longer needs it.
  - Additional SHA256 password hashing.
  - Security features cause a 0.5-1.0 second hang on data access.

Minimum Specifications

- Windows
  - x86/x64 CPU with SSE2 Support
  - DX10/DX11/DX12 capable GPU
- Mac OS
  - x64 CPU with SSE2 Support
  - Metal Capable Intel and AMD GPUs

Repository: https://github.com/UHM-Cyber-Defense/UHMCyberDefense

Final Release:

- **Windows Version:**
  - https://github.com/UHM-Cyber-Defense/UHMCyberDefense/releases/tag/1.0
- **Mac OS Version:**
  - https://github.com/UHM-Cyber-Defense/UHMCyberDefense/releases/tag/v1.0