KARELIA University of Applied Sciences
Degree Programme In Business Information Technology

Elmeri Telimaa

Hacking Detection in Unreal Engine 4

Thesis
May 2021

Karelia
UNIVERSITY OF APPLIED SCIENCES

THESIS
February 2021
Business Information Technology

Tikkarinne 9
80200 JOENSUU
FINLAND
+ 358 13 260 600 (switchboard)

Author (s)
Elmeri Telimaa

Title
Hacking Detection in Unreal Engine 4

Commissioned by
-

Abstract

The goal of the thesis is to find an integrated tool within Unreal Engine 4 for detecting and combating cheating, that is quick to implement in a project. To achieve this goal, a prototype game was created and speedhacked. A counter for this hack was then implemented.

Cheating in online games results in worse experience for the other players playing against the cheater, who then take their business elsewhere. This has a negative impact on both the game developer's reputation and revenue.

The speedhack used manipulates time on a client, making the user move more than intended. The detection method included within Unreal Engine 4 compares the time sent by the client to the time of the server to determine if there is discrepancy in the values. Using these values, we can determine if the client is speedhacking.

The used detection method detected the hack and kicked the offending client out of the server. The method is easy to implement in a new project.

Language
English

Pages 32
Appendices 1
Pages of Appendices 1

Keywords
Unreal Engine, cheating, online games, speedhack, anti-cheat

Karelia AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Helmikuu 2021
Tietojenkäsittelyn koulutusohjelma

Tikkarinne 9
80200 JOENSUU
+358 13 260 600 (vaihde)

Tekijä(t)
Elmeri Telimaa

Nimeke
Hacking Detection in Unreal Engine 4

Toimeksiantaja
-

Tiivistelmä

Tämän opinnäytetön tavoitteena on löytää Unreal Engine 4-pelimoottoriin integroitu työkalu, jonka avulla voidaan havaita ja estää huijaamista. Menetelmän tulisi olla nopea ottaa käyttöön projektissa. Tähän tavoitteeseen pääsemiseksi luotiin peliprototyyppi, ja tämä prototyyppi speedhackattiin. Tämän jälkeen luotiin esto kyseistä speedhackia vastaan.

Verkkopeleissä huijaaminen heikentää niiden pelaajien kokemusta, jotka pelaavat peliä niin kuin sitä on tarkoitettu pelata. Rehelliset pelaajat käyttävät tämän johdosta peliin vähemmän rahaa. Tällä on vaikutusta sekä pelintekijän maineeseen että liikevaihtoon.

Speedhack manipuloi aikaa asiakasprosessissa, mahdollistaen pelaajan liikkumisen nopeampaa kuin on tarkoitettu. Unreal Engine 4-pelimoottorin huijauksen havaitseminen tapahtuu asiakkaan ja palvelimen aikaleimoja vertailemalla. Näissä arvoissa ilmenevien eroavaisuuksien avulla voidaan havaita huijaava asiakas.

Käytetty havaitsemismenetelmä havaitsi huijauksen ja poisti huijaavan asiakkaan palvelimelta. Menetelmä on helppo ottaa käyttöön uudessa projektissa.

Kieli
englanti

Sivuja 32
Liitteet 1
Liitesivumäärä 1

Asiasanat
Unreal Engine, huijaaminen, verkkopelit, huijauksenesto

Contents

Appendices

1 Introduction

I chose cheat detection as a subject as I found it to be an aspect of game development that is not generally well covered in Unreal Engine 4 materials for new game developers, even though it is a vital aspect of multiplayer games. What I would like to achieve, is an easy to follow guideline on how to start development of a game with cheat protection in mind from the start. The goal is not to create a comprehensive guide, but a solid foundation on which it is easy to expand upon, providing game developers the tools and foundation to think about cheat protection early in development.

This thesis observes cheating in games, establishing what different kinds of cheats there are and how they are used in games? Cheating in games can be any illegitimate method a player uses to gain an unfair advantage. What is considered cheating differs on a case to case basis, as a method of gaining advantage may be allowed by one game and disallowed by another. How large of an impact cheating has on the revenue and reputation of online game industry? What choices exist for the game developer to combat cheating and how cheating in games is handled by legislation? Some examples to illustrate cheating incidents in professional gaming are also included.

The thesis aims to find if Unreal Engine 4 has an integrated tool for detecting and combating cheating, that does not require extensive effort to implement? Cheating in online games is a significant issue due to the effect it can have on both the reputation of the game and the reduction in enjoyment for paying customers. The aim is to research some ways of combating speedhacking with the integrated tools within Unreal Engine 4 (UE4 from now on). The type of hack chosen for our research is a speedhack, which allows the cheater to increase the mobility of their character, because it is a technique that can be detected and countered completely on server side within the game engine while causing perceptible issues if nothing is done about it. For example, a wallhack can be much more challenging to detect as they are usually on the client side and the

cheater can attempt to hide the usage. In this thesis, the term “hack” is used to describe a way of altering a game process itself.

The hacking of games can be achieved by manipulating parameters the game stores in memory, such as increasing the health value of the player character. Network packets, transporting information about character movement, status updates or location of players, can be intercepted and read without accessing the memory. Code injections insert code into the process and routes the execution through custom code which can be used, for example, to block a player from ever taking damage or being seen by the AI. Cheats can also enable the player to move faster than intended, which is the technique we look into in this work.

1.1 Previous research on the subject

While research on cheating in general is readily available, there were no sources to be found that handle practically tackling cheating within the UE4 game engine in a form of a tutorial. As a result, setting up a simple solution for hacking detection can demand much more research than a new developer might expect.

The engine documentation does provide a description of the variables and classes that govern these systems. This documentation does lack simple instructions on how the systems can be enabled.

2 Cheating in games

Cheating in games be anything from using external programs that improve a cheater’s aim or allow them to see through walls to using bugs in the game to gain an unfair advantage. Cheating can also occur outside the game itself; for example, in a professional tournament, the participants may use illegitimate means to gain information on the other team’s actions, for example, look at a screen within their line of sight to see where the opponents are on the map.

Cheating in online games is a serious issue for the gaming industry. In a survey, 37% of gamers said they had cheated at least at some time. 88% of gamers have had negative gaming experiences due to cheaters. As gamers grow tired of having to compete against players that have unfair advantages, they will move on to games that offer a more even playfield. (Irdeto 2018a.)

Cheating in online games deteriorates the experience for the other players playing against the cheater, and legitimate players lose faith in the developers, taking their business elsewhere. Continuous customer loyalty has become crucial as gaming revenues have shifted from the sale of the base game to the sale of in-game-purchases. In a survey by Irdeto, 78% of gamers reported to be less likely to spend money on a game with a cheater problem. (Irdeto 2018b.)

Statista.com listed the online game market revenue forecast for 2020 to be around 21,113 million USD (Statista 2021). If the 78% figure above is considered, the cheating could impact 16,468 million USD worth of gaming revenue. As the amounts in question are significant, it is necessary for game developers to find affordable ways to combat cheating in their games.

Cheating in single-player games, when not playing in any competitive capacity, can be seen less problematic as opposed to cheating in online games. In these types of games, cheating can also be a positive aspect, as it may allow a player to pass a point in a game that is simply too difficult, or allow the player to modify the game, making it more fun for them. This kind of cheating will not degrade the experience for any other player.

2.1 Types of cheats

Cheating has many forms, and it is neither sensible nor possible to list them all here. The ones listed here are the ones most fitting for this work as they augment the cheater’s abilities or allow them access to information that should not be available to them. New forms of cheats are developed continuously.

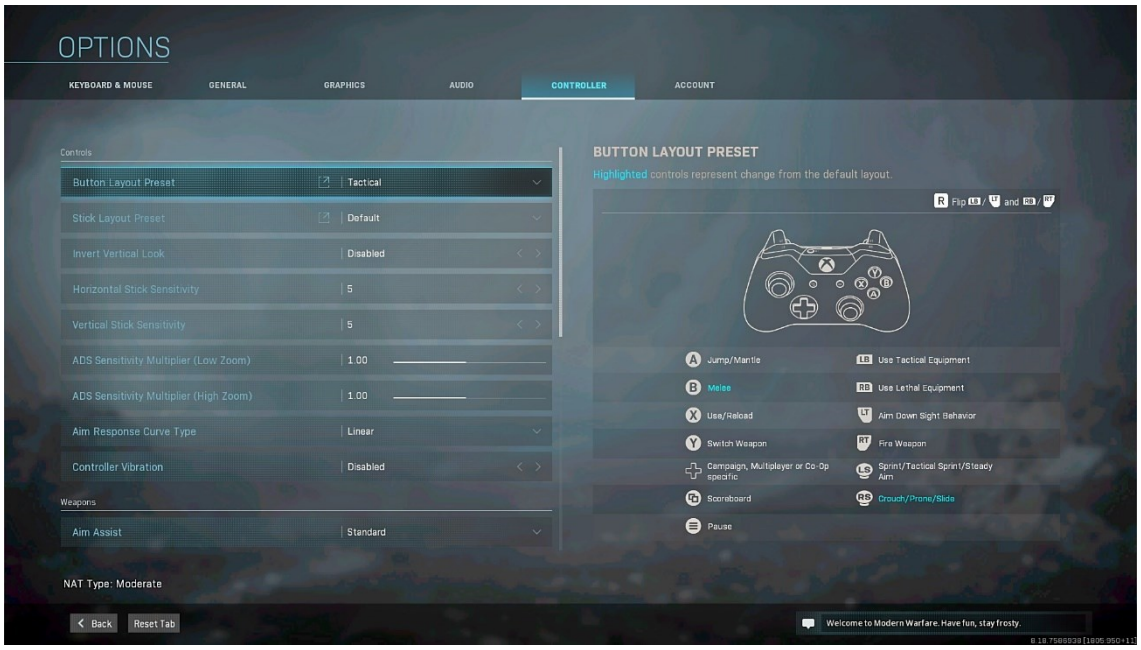
Aimbot:

Aimbots are cheats that move a player's aim on top of the target, often to the part that causes the highest damage when hit. Some aimbots allow them to be set to different levels of effect, instead of instantly snapping on the head of the target, in order to avoid detection. This can be combined with another type of cheat, a triggerbot, which fires the weapon when the aim is on target. (Singh Rawat 2019.)

A player can legitimately experience similar behavior to an aimbot. Most console shooters and some PC shooters allow the player to enable aim-assist, which essentially works as an aimbot but is designed to be in the game. When implemented by design, this is used to assist players that use controllers, since a controller is generally not as accurate as a mouse (Davison 2020).

Introduction of crossplay, which enables PC players and console players to play together on the same servers, has also sparked debate on the fairness of aim-assist. While generally agreed that controller players are at a disadvantage compared to players using mouse and keyboard, it is also argued that while attempting to level the playing field, aim-assist gives controller players unfair advantages especially in close range fights. (Davison 2020.)

Call of Duty: Warzone is an example of a game that allows crossplay and provides controller users the option to turn on aim-assist (Picture 1). The aim-assist available is most effective in fights that happen at close range. Additionally, aim-assist does not care if the target is visible to the player, locking on to targets hiding in the foliage where a player normally cannot see. (Perry 2020.)



Picture 1. Call of Duty: Warzone controller settings page (Picture: Perry 2020)

Mobility hack:

Cheats that improve a player's ability to move are called mobility hacks. These can be higher movement speed or ignoring collisions (Singh Rawat 2019). If the user of the hack does not care about getting caught, they can become unreachable for other players. In a racing game, these could be used to increase the car's acceleration, maximum speed and grip, providing a distinct advantage.

Wallhack:

Wallhacks and extra sensory perception (Picture 2) allow players to see through walls and, in the case of extra sensory perception, also additional information about the target's status (Singh Rawat 2019). If the user of the hack is experienced, they can often hide the usage of this type of cheat quite well.

Picture 2. ESP in Escape From Tarkov (Picture: Pinterest 2021)

Exploits:

Exploits include abusing software bugs, technical limitations or design flaws to gain an advantage. IL-2 Sturmovik: forgotten battles, a flight simulator that was released in 2003, was well protected from the most obvious cheats, for example, mobility hacks or aimbots, for a long time. The only cheat widely used was pressing the print screen button on the keyboard. This took a screenshot each time it was pressed, resulting in the client not sending updates to the server for that moment. By continually tapping the key, a player could stop sending updates, causing their plane to continue moving in the same direction and speed for the duration, looking like lag. Once they stopped tapping the key, their craft would quickly warp according to the new updates finally making their way to the server. As a result, the player that was hunting them had followed a false representation of their target, often resulting in them finding themselves in a poor position compared to the targets real position.

Audience:

Using audience clues is somewhat of a controversial topic. There have been clear cut instances, as audience waving banners giving information to their favorite team (Singh Rawat 2019). There is more room for interpretation when we talk about more normal audience reactions. Peter Rasmussen, a professional CS:GO

player, commented on the subject in 2019 stating that while players reacting to the audience getting excited is a perfectly understandable reaction, the audience shouting directions is completely wrong (Rasmussen 2019).

2.2 Types of cheaters

A cheat developer interviewed for PCGamer.com put cheaters into two categories: rage cheaters and closet cheaters. Rage cheaters are the ones that get banned almost immediately, blatantly using all the tools at their disposal, making it obvious they are cheating. The author of the article “Hacks! An investigation into the million-dollar business of video game cheating” tried this sort of cheating in Counter Strike: Global Offensive and was banned within hours. (Maiberg 2014.)

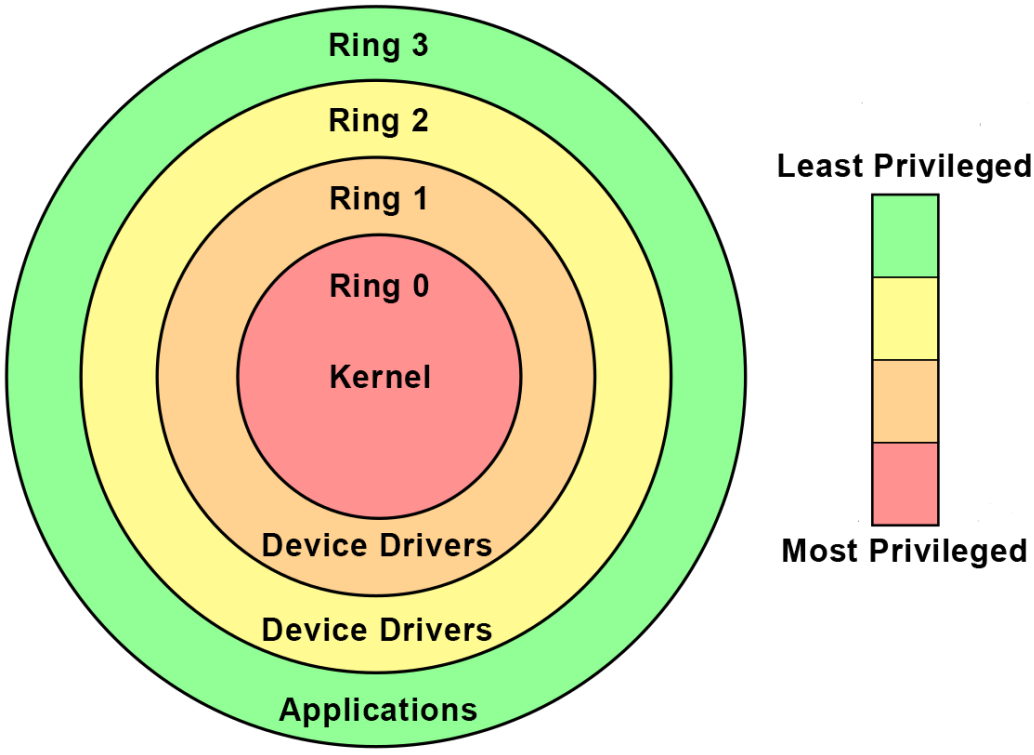
Closet cheaters are the ones that use cheats and spare no effort in making sure they will not be detected. They avoid using aimbots that assist one’s aim and other cheats that make detection obvious, favoring information-based hacks such as wallhacks. They also intentionally lose some games to make themselves seem legitimate. When using wallhacks that allow one to see through walls, they try to act as if they did not have it. They also try to make their game account used for cheating look legitimate, logging in idle hours of game time and buying some in-game items. (Maiberg 2014.)

2.3 Anti-Cheat software

The function of anti-cheat software is to detect cheating and ban the offending accounts. Anti-cheat software detects running cheat programs. Software companies developing anti-cheat software are secretive about their methods, which leads to very limited information on their functionality being available. (Lehtonen 2020, 6-8.)

Some anti-cheat software function on the kernel level. “The kernel is a program that constitutes the central core of a computer operating system. It has complete control over everything that occurs in the system.” (The Linux Information Project 2005). With this level of access, a program can access operating system API functions and system memory (Picture 3). Forcing this level of access to a user’s computer has caused alarm among some players, as they are afraid the developers will use this access to gather information about their users. A League of Legends developer Phil Koskinas states that they would have far easier methods for gathering data from their user’s computers and that this level of access is required to combat cheats that also function on kernel level. (Koskinas 2020.)

Riot Games’ anti-cheat software, Vanguard, has been criticized for running constantly in the background, even when the game is not being played, potentially making the computer vulnerable to keyloggers, bitcoin miners or other malicious activity (Alford 2020).



Picture 3. Application access levels (Picture: Koskinas 2020)

Statistical anti-cheat, such as FairFight, creates a profile for all players and tracks their gameplay data. Their actions are compared with statistics to detect cheating without requiring invasive software solutions. It is also easy for developers to implement in their games. (Lehtonen 2020, 28.)

2.4 Report system

Many multiplayer games offer a reporting system, allowing their players to report suspected cheaters and players that otherwise act against the game's rules. Reports can flag a user for further investigation, and the reporting player also receives a score based on the accuracy of their reports (Faceit 2021).

Official CS:GO matchmaking gives each player a trust factor. The trust factor is determined by a wide range of activities, including the number of reports they receive from other players. Matchmaking then tries to match players with similar trust factors together. The details of the information calculated are not disclosed by Valve, the developer of CS:GO. (counter-strike.net 2021.)

2.5 Cheating in legislation

South Korea has passed a law that makes "distributing programs to modify or gain an advantage during play" illegal and punishable by fines or jail time (Humphries 2016). In absence of such laws, some companies have tried combating cheaters with lawsuits based on more traditional legislation, such as the Copyright Act of 1976, with varying results (Rightberg 2016).

Cheating and creating cheats are forbidden in the terms of use of most games, a contract each player must agree to, before being able to play the game. Acting in a manner that is against the Terms of Use constitutes a breach of contract. Most of the times, this often means the offending player will be banned from playing the game in the future. (Symes 2017.)

2.6 Professional tournaments

Should cheating in esports occur in the form of aim-assists, wallhacks or other methods directly affecting the players' ability to function in the game, it can be compared to using performance-enhancing drugs in traditional sports. These methods, just like drugs, enhance an athlete's abilities in certain aspects.

In LAN (Local Area Network) tournaments, where each contestant is present in the same arena, cheating is quite challenging as computer systems can be better controlled and there can be judges physically present monitoring the players.

Due to the outbreak of Covid-19, esports tournaments have moved from LAN setting to remote tournaments. Some players play from home while some teams have their team together in the same space. Webcam feeds shine light on what is happening, but as a physical presence for judges is much harder to arrange, there is more emphasis on anti-cheat software to assure the games are legitimate.

2.7 Cheating cases in professional tournaments

There have been some instances where cheating has occurred in a professional tournament, with money on the line. Multiple League of Legends players were found to be looking at the main screen (Picture 4) on stage to see what their opponents were doing (Plunkett 2012). A semi-professional CS:GO player was caught cheating, which ultimately lead to other pro players being alleged of cheating (Grayson 2014).

Picture 4. League of Legends stage in 2012. (Picture: Lien 2012)

In 2018 a CS:GO player Nikhil “forsaken” was given a ban by ESIC (Esports Integrity Commission) for 5-years (ESIC 2018). This cheat was detected by a combination of anti-cheat software and a manual inspection of his computer. Upon inspection, they found a suspicious program that he tried to shut down and delete (HLTV 2018).

In 2020 multiple professional CS:GO coaches were suspended after they were caught exploiting a bug in the game, which allowed them to see more of the game map than they were supposed to. Thirty-seven coaches were sanctioned as of September 2020, and investigations are ongoing. (Matthews 2020.)

In Blast premier global finals 2020, a high level CS:GO tournament played January 2021, team Vitality staff had the live broadcast of the event shown in their lounge area. It was seen that any information from the stream was not given to the players of the team and the incident was a matter of negligence instead of an attempt to gain an unfair advantage. As a result, the team was fined for $10,000 but were allowed to retain their results. (ESIC 2021a.)

2.8 Punishment for cheating

As a punishment for cheating, players can receive a ban on their account, which prevents the account from playing the game. These bans can be for a duration or permanent. In the case of VAC (Valve Anti-Cheat System), the bans are permanent (Steam 2021).

In professional tournaments, disciplinary action is governed by the tournament rules. ESIC partners with some tournament arrangers, such as ESL (Electronic Sports League, Dreamhack and Blast Pro Series, to provide oversight on fair play (ESIC 2021b).

ESIC rates offences in levels from 1 to 4, where level 4 includes the most serious offences. Penalties from offences include warnings, fines, loss of prize money and suspensions. Sanctions escalate for repeat offences. (ESIC 2021c.)

2.9 Overview

It has been established that the game industry is losing revenue due to dishonest players that affect the experience of legitimate players. Players trying to enjoy the game in a manner it was intended by the developer can become frustrated when they cannot compete with cheaters that have enhanced aim, vision or mobility.

With the rise of professional gaming and esports, maintaining the credibility of the business becomes a priority. For esports to be acknowledged as a serious sport, there need to be systems in place to detect, prevent and punish any attempt at foul play.

While the current anti-cheat software offers results against known cheats and cheats that are obvious, cheating has remained a prevalent problem within the gaming community. This is helped by more and more titles being free to play, meaning a cheater who gets banned may create a new account without risk.

Some anti-cheat software also functions with a level of access to the user's computer that can raise questions about their own security. Functioning with these privileges comes with responsibilities. The anti-cheat software itself needs to be secure and not become a security loophole that other malicious software could exploit.

Game developers themselves are not without responsibilities either. They are the ones that decide whether or not to implement anti-cheat software in tandem with their game and which software they choose to use. Should there be any problems with the software they have licensed, it will reflect poorly on them as well since the problems were caused by software bundled with their game. For some games, a solution that utilizes methods for combating cheats, built in the game engine, would be preferrable.

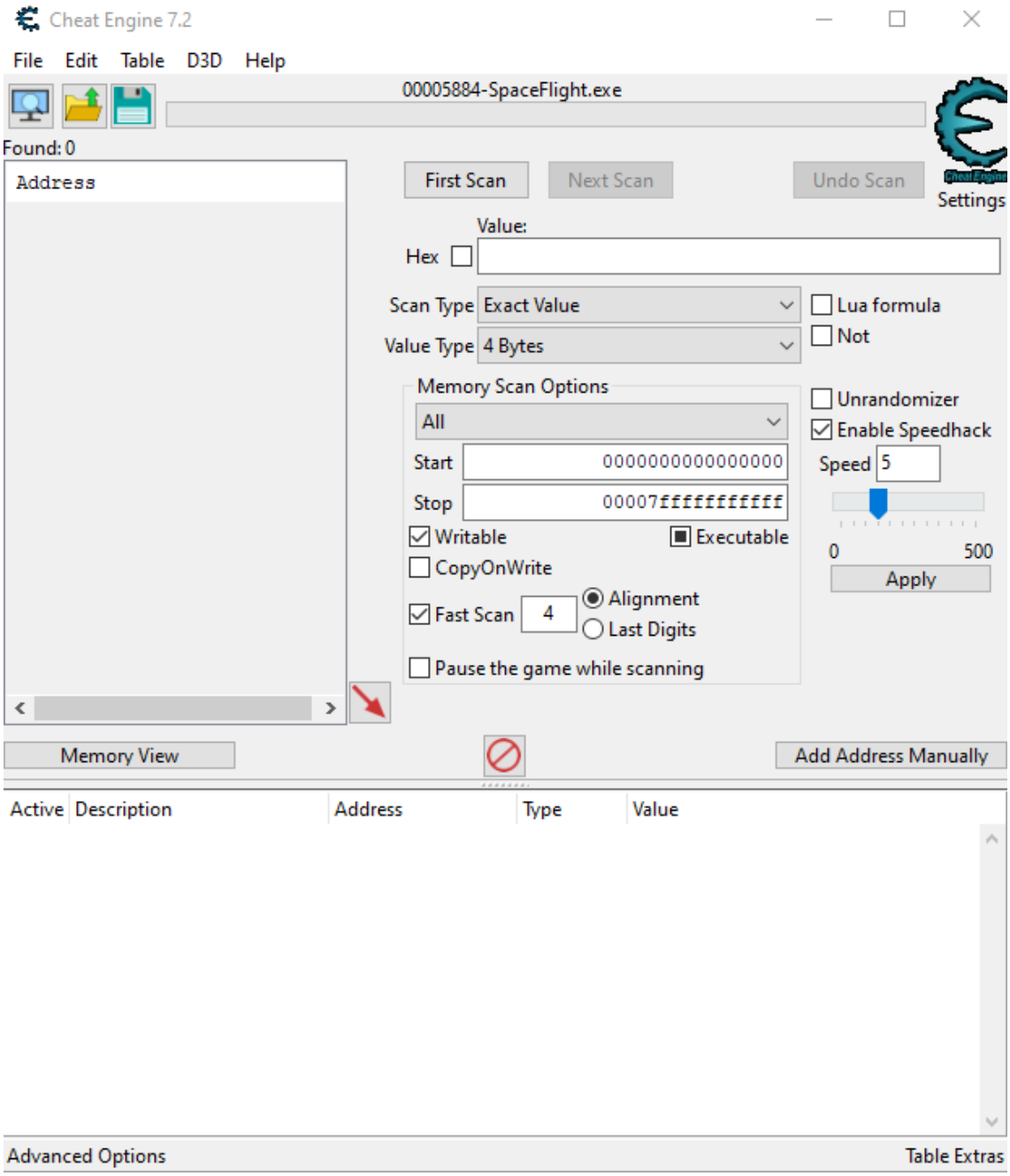
3 Tools

Choosing the correct tools is an important part of a successful project. Since the subject of hack detection is already time consuming, it was important to get the prototype game ready without much delay. For hacking the game a ready to use tool was needed, as the goal of this thesis is not to instruct the reader on how to create a tool for hacking games.

3.1 Cheat Engine

Cheat Engine is an open source tool, that can be used to alter a game process in various ways. Its intended use is to provide the means to modify a single player game in an accessible way. These features include a memory scanner, debugger and various other tools beneficial for both hobbyists and professionals alike. It can also be used for DLL injection, enabling the user to run their own commands. (Cheat Engine 2021a.)

In this work we will use cheat engine to speedhack a game prototype (Picture 7). While Cheat Engine is designed for use with single player games, it provides easy access to the game process on the client, which is all we need to speedhack the prototype.



Picture 7. Cheat Engine GUI with speedhack enabled. (Picture: Elmeri Telimaa)

The speedhack provided by cheat engine hooks into the game process and alters the timers within the game to speed up or slow down time (Cheat Engine 2021b).

This only affects the game process that runs on the client and will not affect the server.

With the altered timers the client process will be manipulated to think there was more time spent within frames than in reality. This time is stored in the DeltaTime variable. As the movement component of a character determines the amount of distance the character needs to move based on DeltaTime, the manipulated value causes the character to move further than normally

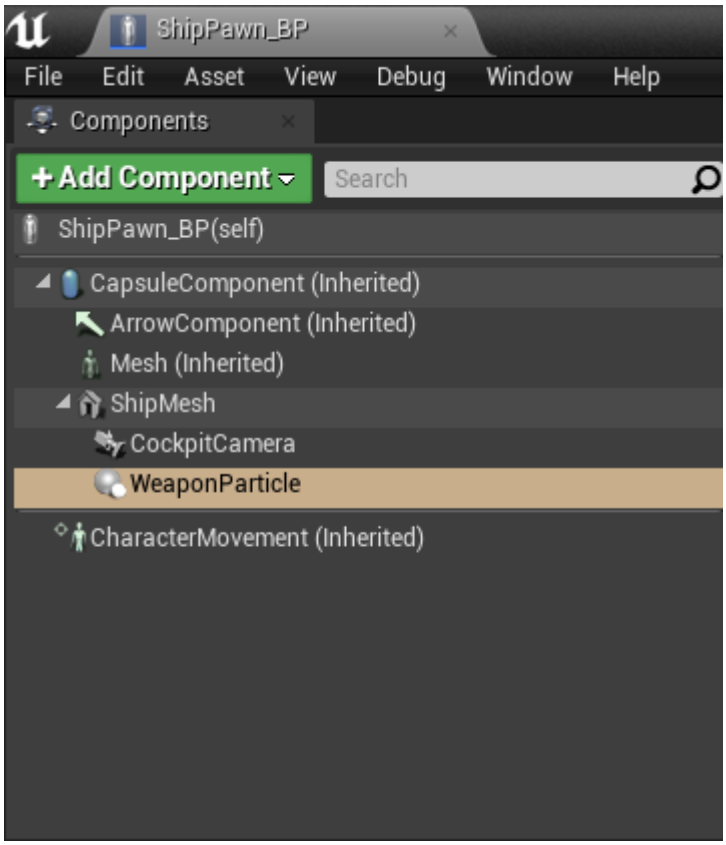
3.2 Unreal Engine 4

Unreal Engine 4 is a game engine capable of AAA-quality games. The programming language used in UE4 is C++, but a scripting tool called Blueprints can also be used for programming. UE4 allows the users free access to the complete engine source code, enabling developers to customize and extend the engine. The engine also provides a multiplayer framework for the development of networking solutions. (Epic Games 2021a).

UE4 relies heavily on class inheritance. The class that inherits another is called a child class, while the original is called a parent class. Inherited classes retain the parent classes functionality but can build more functionality on top of it, without affecting the parent class. Some methods of the parent class can also be overridden, replacing the method completely. All objects in UE4 are based on the UObject class (Epic Games 2021b).

Actors are the base for most objects visible in a UE4 level. The AActor class provides the base on which to build gameplay functionality on. It also includes settings for network features regarding the actor.

Actor components are classes that can be attached as a child to an actor. These components cannot be placed in the world by themselves. The particle effect of firing a weapon is a component attached to the ship (Picture 5).



Picture 5. Actor components in editor. (Picture: Elmeri Telimaa)

A server runs the game and has authority over all the game's events. The server is the only one that has a real-time representation of the game's state, as the state relayed to the players is always slightly behind in time due to latency.

A client is a game instance that connects to a game session hosted by the server. The client sends the actions of the player to the server using remote procedure calls, a way of requesting a method to be executed on another machine, and the server validates them before executing them.

AGamemodebase inherits from AActor. This is a class that only exists on the server and is designed to be the authority over the rules of the game. It has overridable functions that get called as a new player enters and leaves the game.

Actors can be set as replicating. Replicated actors use a system called replication to automatically synchronize variables of the actor from the server to the relevant clients. All replicated actors have a unique network ID, which is used to find the

correct object on all game sessions. In order to be able to send RPCs, an actor needs to be replicated.

4 UE4 Multiplayer Framework

The networking architecture used by UE4 is server authoritative (Diagram 1), where the information sent by the clients is validated by the server. This method generally requires the server to perform checks to validate actions, for example when a client notifies the server that the player hit an opponent, the server should look back at that moment and see if it is indeed correct. Only if the event is valid, the server will perform the action and update the results to the clients using either RPCs or replication. Two different versions of the server are built for this thesis, one that validates client movement and one that does not.

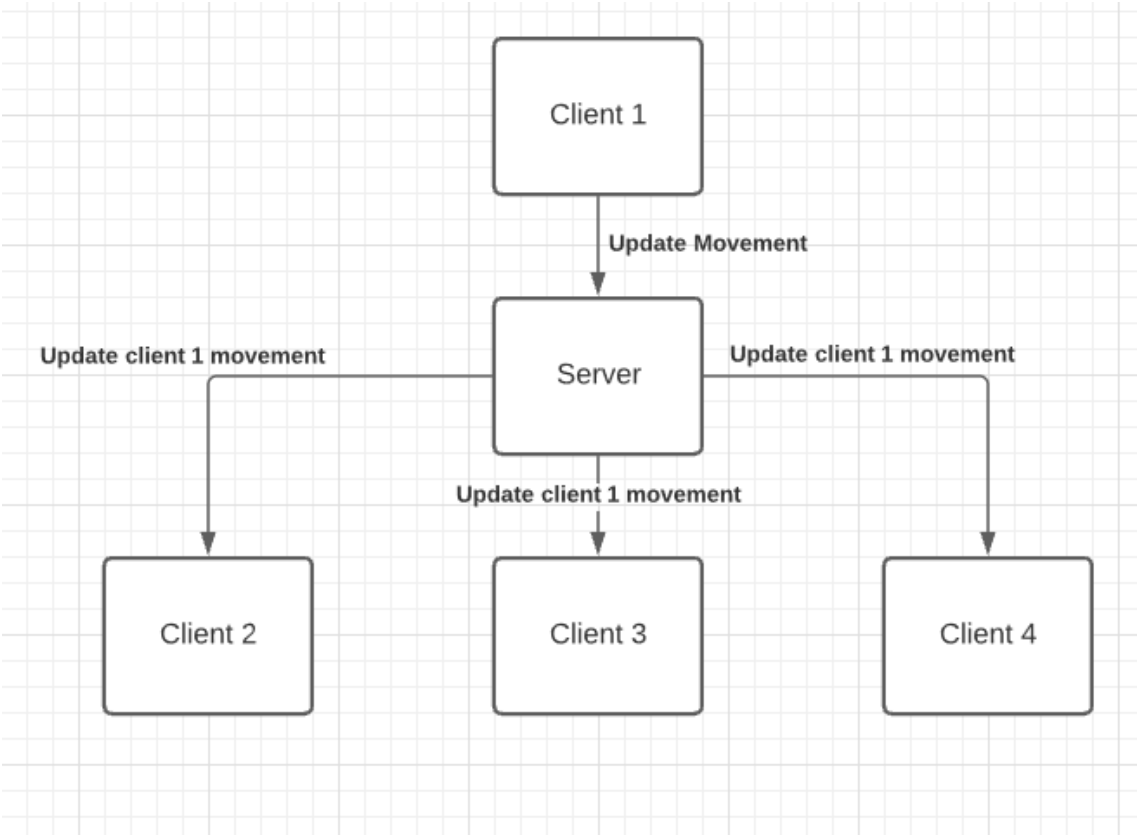
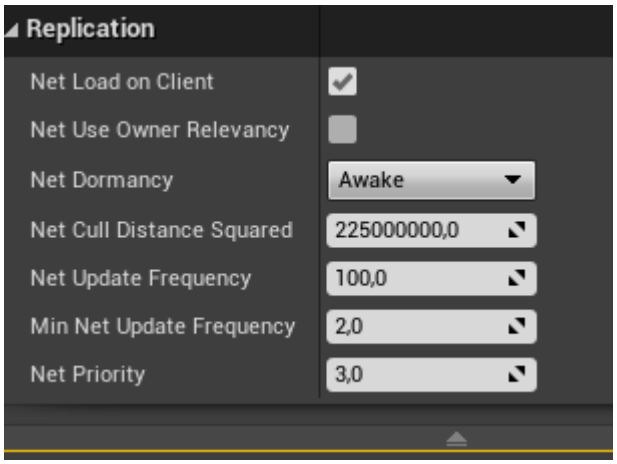


Diagram 1. SpaceFlight network architecture (Diagram: Elmeri Telimaa)

4.1 Property Replication

For actors that are set to be replicating, the server updates to all relevant clients the values of variables that are set to be replicating. These values are only updated from the server to the client. If the value of such variable is changed on a client, it will continue to be different from the server until the server overwrites it with its own update. (Epic Games 2021c.)

The frequency that the server updates these variables can be set individually for each type of actor. To optimize network usage, it is a good idea to set the frequency of updates as low as is possible without causing gameplay problems. This is set in the “NetUpdateFrequency” variable in AActor (Picture 6). Replication can also be set to update only the client that have ownership of the actor with the “bOnlyRelevantToOwner” variable.



Picture 6. NetUpdateFrequency setting in editor. (Picture: Elmeri Telimaa)

4.2 RPC – Remote Procedure Call

Remote Procedure Calls (RPC) are functions that allow function calls to be called on a remote process. Using these calls, the server can tell a client to run a specific function and vice versa. (Epic Games 2021d).

RPC functions can be validated. This is enabled with the “WithValidation” keyword. Validation is a custom check, where the developer can define their own logic of validating the action. The validation function returns true or false. If it returns false, the offending client is kicked out of the game as a cheater.

Client and NetMulticast RPC calls are meant to be called on the server (Table 1). A client-RPC will be executed on the client with ownership of the actor and will not be executed on any other client. NetMulticast will be called on all the clients and the server, regardless of ownership.

Actor ownership	NetMulticast	Server	Client
Client-owned actor	Runs on server and all clients	Runs on server	Runs on actor's owning client
Server-owned actor	Runs on server and all clients	Runs on server	Runs on server
Unowned actor	Runs on server and all clients	Runs on server	Runs on server

Table 1. RPC called from the server (Table: Epic Games 2021d)

Clients only have one functional RPC-type (Table 2). These RPCs must be called on an actor that the client has ownership of. The method will then be executed on the server. The server should always validate the request before accepting it.

Actor ownership	NetMulticast	Server	Client
Owned by invoking client	Runs on invoking client	Runs on server	Runs on invoking client
Owned by a different client	Runs on invoking client	Dropped	Runs on invoking client
Server-owned actor	Runs on invoking client	Dropped	Runs on invoking client
Unowned actor	Runs on invoking client	Dropped	Runs on invoking client

Table 2. RPC called from a client (Table: Epic Games 2021d)

RPCs are excellent for running customized functionality where altering a variable is not enough. For example, it would be possible to replicate a Boolean value “bIsHit” on the server when we hit another player. The server would constantly check if the value of the variable returns true and act accordingly. RPCs provide a cleaner and more effective way of dealing with this type of situation, allowing the developer more control over when to execute the required logic. As the client hits the target, it can send an RPC to the server. This RPC can have references to the player that was hit, the player dealing damage, weapon used etc. This way the server receives all the information it needs to check if the event was valid, how much damage was made and who to award points to without having to constantly check for a value.

RPCs are declared reliable or unreliable. If no declaration is given, an RPC function is unreliable (Epic Games 2021d). Reliable calls should be used when they are absolutely necessary to arrive to their intended recipient. This includes gameplay critical functions, such as hit events or movement. Unreliable calls are usually good for synchronizing visual effects, where missing a function call on a client is not necessarily even noticed by the player.

It is possible to pass pointers to other actors as pointers in an RPC. The actor needs to be replicated in order for this to work as it needs to have a valid network ID.

```
UFUNCTION(Server, Reliable, BlueprintCallable)
void ServerNotifyHit(AShipPawn* TargetHit);
void ServerNotifyHit_Implementation(AShipPawn* TargetHit);
```

Listing 1. Server RPC from SpaceFlight. Called from a client.

4.3 Actor Ownership

Each actor is owned by either a client connection or the server. A client connection has ownership of the pawn it is possessing and what the pawn owns, including the pawns components. The server uses ownership to determine which clients receive replication updates and RPCs. (Epic Games 2021e.)

In SpaceFlight, this system is used when ships receive damage. The RPC notifying server of a target hit (Listing 1) includes a pointer to the ship that was hit. Since this is a replicated actor, the server can find the target ship. At this point the server should validate the event, but in SpaceFlight it will simply believe it to be true. The server will deduct the damage dealt from the target ships Health-variable. Since the health variable is replicated (Listing 2) and set to replicate regardless of ownership, this value will be synchronized from the server to all clients.

```
UCLASS()
class AShipPawn : public ACharacter
{
	GENERATED_BODY()

public:
	// Sets default values for this pawn's properties
	AShipPawn();

	UPROPERTY(Replicated, EditAnywhere, BlueprintReadWrite, category =
	"Properties")
	int32 Health = 100.f;
```

Listing 2. Health variable declaration.

5 Case prototype: SpaceFlight

SpaceFlight is developed solely for the purposes of this thesis. It is an arcade multiplayer space flight game. Player controls the pitch, yaw and roll of his craft, is able to fire a beam weapon and has 4 engine settings (Idle, Slow, Fast, Full). The game project is available for download from GitLab.com: https://gitlab.com/Jeke_/spaceflight.

The game is designed to be simple and easy to understand. There is no physics simulation. The controlled ship moves towards the current forward direction each tick. The amount of movement is determined from the current active engine setting. Pitch, yaw and roll have independent maximum speeds and receive axis input from controller. All of these controls utilize DeltaTime to determine the amount of movement necessary for each tick.

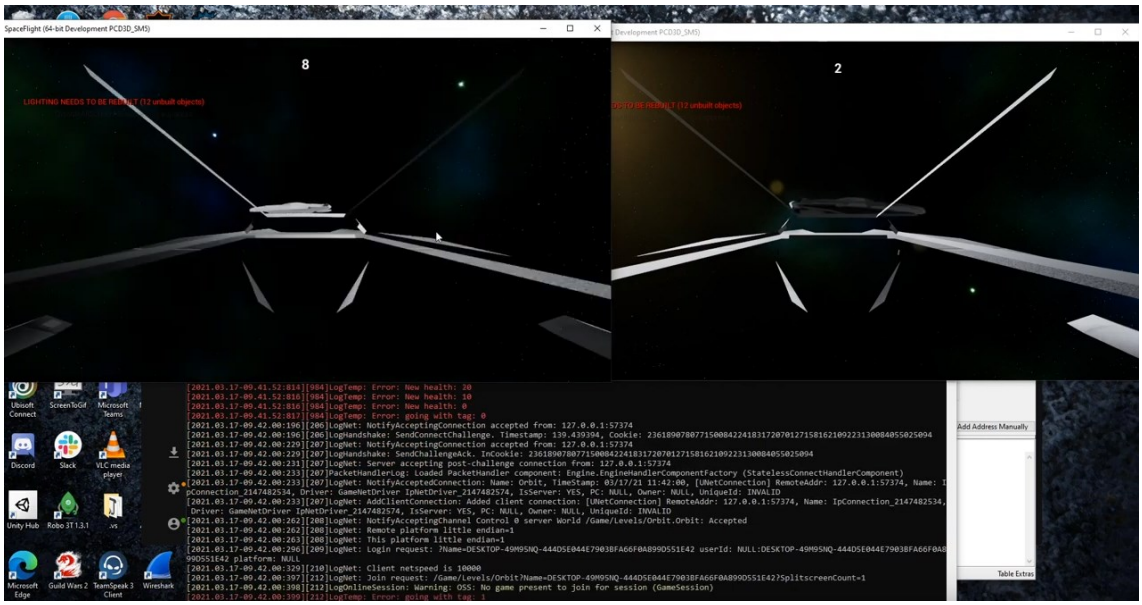
Firing mechanism is a line-trace system, where the game calculates a line forward of the player's craft to the distance set in weapon attributes. If this line hits another craft, it will affect the target. In case of a hit, the client will send a message to the server notifying of the hit. To save time, there is no validation for this on the server, and the client will always be trusted.

5.1 Prototype: Network architecture

SpaceFlight uses UE4's replication system as a networking solution. Replication system is integrated within UE4 and is easy to build a fast prototype with. Replicated movement is used to handle networked movement and logic of firing weapons and taking damage is handled through RPCs (Remote Procedure Calls)..

5.2 SpaceFlight speedhack

SpaceFlight was speedhacked using Cheat Engine. To make sure the speedhack can be quantified, a new gamemode and scene were created. This gamemode uses the same ship character as ADMGamemode. In this scenario, the player always starts with the engine at full power. A larger ship is spawned ahead of the player. Around the large ship is a triggerbox. A triggerbox is an invisible box that can detect objects that enter it. After spawning, the player has 10 seconds to reach the triggerbox or be destroyed. If the target is reached, the player will no longer be destroyed once the time has run out. In this case the player can now roam the level freely. (Picture 8).



Picture 8. Two clients moving towards target ship. (Picture: Elmeri Telimaa)

This system is rigged against a player that does not cheat, as a legitimate player can never reach the target before running out of time. A legitimate player was compared with a player with a speedhack that increases their movement speed to five times the intended speed in a video demonstration. In this demonstration the legitimate client is seen locked in an unwinnable loop, while the cheating client coasts trough the scenario and can continue flying. (Telimaa, 2021).

5.3 UE4 integrated anti-hacking tools

Speedhack detection within UE4 relies on AGameNetworkManager and UCharacterMovementComponent. AGameNetworkManager class is inherited from AActor and is not placeable in the world, meaning the initial attributes can only be edited in the DefaultGame.ini-file. These values can also be altered when the program is running. UCharacterMovementComponent handles movement input and transmitting the information between a client and the server.

AGameNetworkManager class handles many network-related tasks. We will focus on attributes related to cheat detection (Listing 3). These attributes include whether the cheat detection is enabled and threshold values for the different variables utilized. Full settings used in SpaceFlight are listed in Appendix 1.

```
bMovementTimeDiscrepancyDetection=true
bMovementTimeDiscrepancyResolution=true
MovementTimeDiscrepancyMaxTimeMargin=0.20f
MovementTimeDiscrepancyMinTimeMargin=-0.05f
MovementTimeDiscrepancyResolutionRate=1.0f
MovementTimeDiscrepancyDriftAllowance=0.0f
```

Listing 3. AGameNetworkManager settings related to speedhack detection.

bMovementTimeDiscrepancyDetection is the setting that determines if speedhack detection code will be executed at all. MovementTimeDiscrepancyMaxTimeMargin sets the amount of time in seconds a client can be ahead of the server before it is flagged for time discrepancy. This setting should always have some margin for errors, in case of possible network issues. MovementTimeDiscrepancyMinTimeMargin acts similarly with MaxTimeMargin but checks if a client is actually running slower than the server. MovementTimeDiscrepancyResolutionRate governs how many times per second the check is performed. MovementTimeDiscrepancyDriftAllowance is the amount of time, percent of a second, a client's timestamp is allowed to be different from the servers.

5.4 Speedhack detection turned on using default settings

Speedhack detection is turned off by default in a new UE4 project. A simple version, where the engine will detect speedhacking and reset the offending players character into the correct position, can be simply turned on with a single Boolean value.

In this case, a speedhacking client will constantly snap back to a corrected position, making a speedhack nearly impossible to gain any advantage with. This is already a valid counter to speedhacking. It will only react when the offending character is executing a move. The detection code will detect time discrepancy even while the character is not moving, but there is no functionality bound to this detection. This enables the hacker to still achieve higher turn rates then

stationary. While this is a scenario that can rarely be used to gain advantage, it is still possible in some cases.

5.5 SpaceFlight speedhack detection

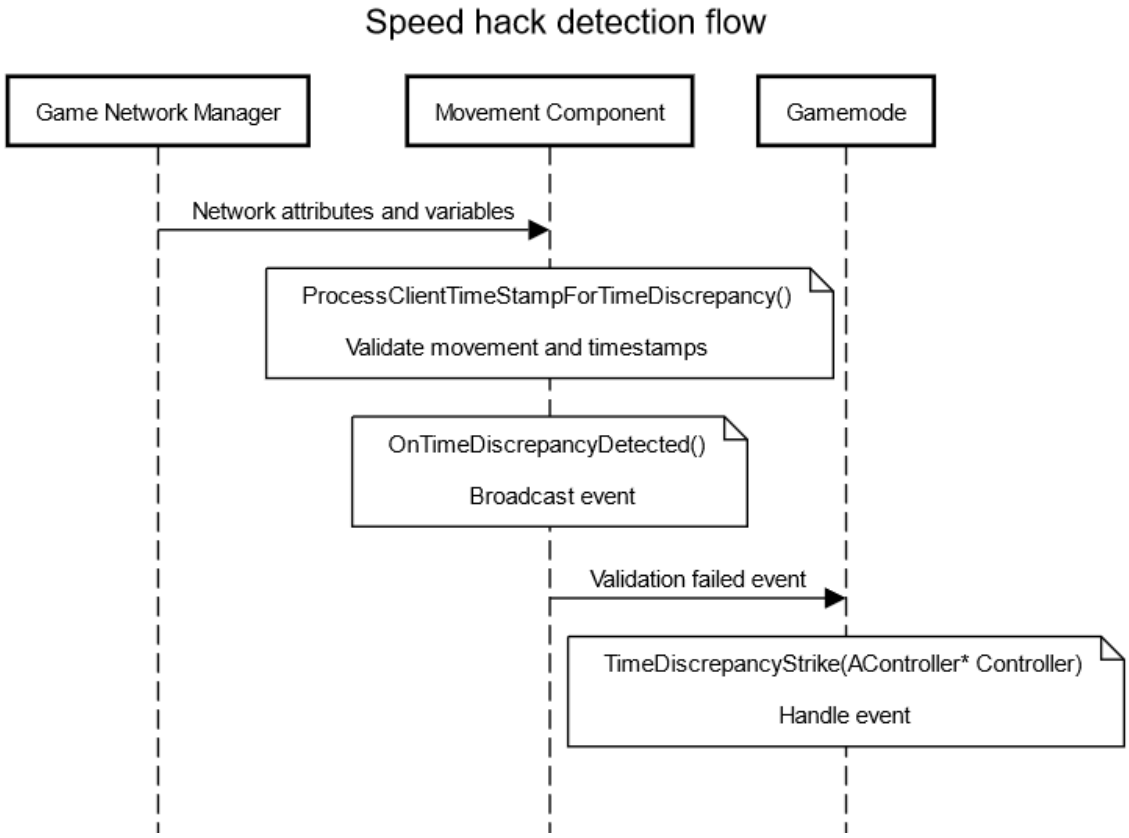


Diagram 2. SpaceFlight speedhack detection flow. (Diagram: Elmeri Telimaa)

The AGameNetworkManager stores variables that are used in speedhack detection by the movement component of each remotely controlled character. When the server receives movement information from the client controlling the character, the move is then validated on the server using the attributes stored within AGameNetworkManager (Listing 4).

```
if (NewTimeDiscrepancy > GameNetworkManager-
>MovementTimeDiscrepancyMaxTimeMargin)
{
	if (GameNetworkManager->bMovementTimeDiscrepancyResolution)
	{
		ServerData.bResolvingTimeDiscrepancy = true;

		ServerData.TimeDiscrepancy = (NewTimeDiscrepancy -
		EffectiveClientError);
	}
	else
	{
		ServerData.TimeDiscrepancy = 0.f;
	}
	OnTimeDiscrepancyDetected(NewTimeDiscrepancy,
	ServerData.LifetimeRawTimeDiscrepancy, WorldTimeSeconds -
	ServerData.WorldCreationTime, ClientError);
}
else
{
	ServerData.TimeDiscrepancy = NewTimeDiscrepancy;
}
```

Listing 4. Time Discrepancy detection in UE4 source code with comment lines omitted.

The code in listing 4 compares the time discrepancy between a client and the server. NewTimeDiscrepancy is the time discrepancy on the newest move added to the discrepancy cumulated from previous moves. If this grows higher than the threshold, the method OnTimeDiscrepancyDetected will be called. Any action desired in the case of time discrepancy can be executed there (Listing 5). In a case where the time discrepancy is over the threshold the time discrepancy is reset and if it is still within allowed margins, the cumulative new time discrepancy is stored for use with the next check.

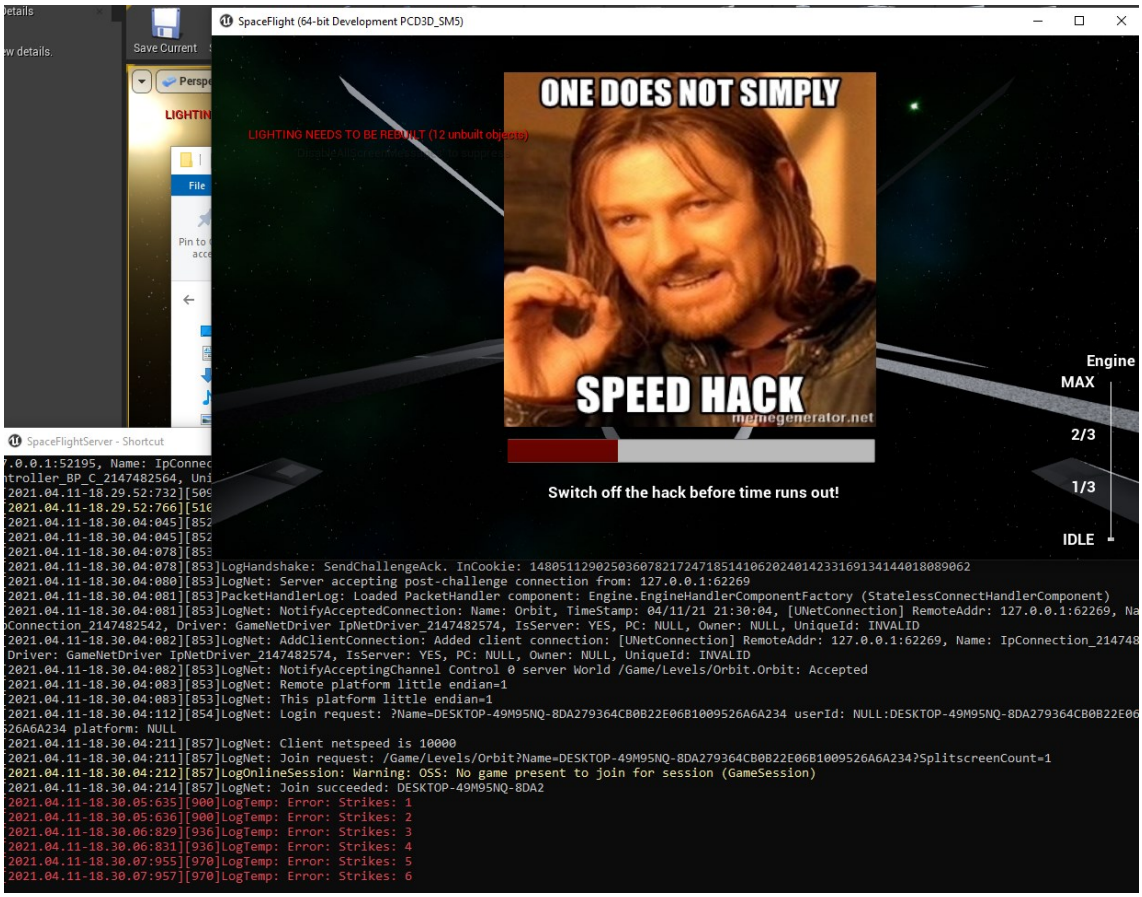
```
void UCharacterMovementComponent::OnTimeDiscrepancyDetected(float
CurrentTimeDiscrepancy, float LifetimeRawTimeDiscrepancy, float Lifetime, float
CurrentMoveError)
{
	ACharacter* OwnerCharacter = Cast<ACharacter>(GetOwner());
	if(OwnerCharacter != nullptr)
	{
		OnTimeDiscrepancyStrikeDetected.Broadcast(OwnerCharacter-
		>GetController());
	}

	UE_LOG(LogNetPlayerMovement, Verbose, TEXT("Movement Time Discrepancy
detected between client-reported time and server on character %s.
CurrentTimeDiscrepancy: %f, LifetimeRawTimeDiscrepancy: %f, Lifetime: %f,
CurrentMoveError %f"),
CharacterOwner ? *CharacterOwner->GetHumanReadableName() : TEXT("<UNKNOWN>"),
		CurrentTimeDiscrepancy,
		LifetimeRawTimeDiscrepancy,
		Lifetime,
		CurrentMoveError);
}
```

Listing 5. OnTimeDiscrepancyDetected.

In SpaceFlight, we use a custom gamemode class, ADMGameMode, that keeps track of all player controllers currently on the server. We have added a custom event into UCharacterMovementComponent class. This is achieved by altering the UE4 source code of UCharacterMovementComponent header and cpp files. This event is called every time the function OnTimeDiscrepancyDetected is called as shown in listing 5.

The gamemode handles these events and adds a strike for the controller of the MovementComponent. A strike represents a single event of exceeding the allowed margins for time discrepancy. When a controller receives their first strike, an RPC call is sent from the server to the client owning the connection. This RPC includes parameters to either hide or show a warning screen and a float value representing the percentage of strikes received from the maximum number of strikes allowed (Picture 9).



Picture 9. Warning prompt on the client. (Picture: Elmeri Telimaa)

While the number of strikes is below the threshold of kicking the offending client, they have the possibility of turning their hack offline. This will allow them to continue playing, but the number of strikes is not reset and prevails for the duration of the game.

5.6 Caveats of SpaceFlight speedhack detection method

The method used to detect and handle in SpaceFlight requires modification to the engine source code. This presents some scenarios that a developer should be aware of when making the decision whether to implement the feature.

Version control:

Version control usually contains only as much data on the project as is required by necessity. This means the version control repository usually only has files belonging to a specific project. Due to customized source code, the engine code

would also need to be added into version control and distributed among all developers.

Moving engine versions:
There may be a time during game development where a move to a newer version of the engine is warranted. In this case customized engine code is one more aspect to take into consideration.

6 Conclusion

The purpose of this thesis was to find a way to implement speedhack detection using the tools provided within the Unreal Engine 4 game engine. The goal was not to make a ready to ship feature, but to provide a base system, which would be fast and simple to implement and expand upon.

To achieve this, a game prototype was built. Cheat Engine, an external program, was used to alter the timers within the game process. This provided us with the speedhack to detect. A built in feature of UE4 was used, which compares server timestamps to the timestamp received from the client, to detect time manipulation.

6.1 Results

Here is reflection of how the goal set for the thesis was fulfilled. An overall look at cheating in general and its effect on revenue is conducted in section 2, where it was established that cheating has a significant impact on the industry. Some of the different types of cheats available are looked at in paragraph 2.1. Sections 2.3 and 2.4 contain research of the options available to game developers for combating cheating, which include third party anti-cheat software and reporting systems. Legislation regarding cheating is looked at in section 2.5. Legislation mostly relies on copyright laws in regard of cheating.

I learned a lot about how networking works in UE4. Researching the methods for detecting speedhacks within the engine also gave me greater confidence in my ability to understand and apply the information given in UE4 C++ API, a resource that can be daunting to use if one is not used to it.

Find if Unreal Engine 4 has an integrated tool for detecting and combating cheating, that does not require extensive effort to implement?

Functionality built within the UE4 engine was used to detect the speedhack. This functionality is built in the engine but is not active by default. The system can be turned on with a single boolean variable, and functions to some extent out of the box.

There are no built in actions further than detecting the issue and, if resolution is set to be on, automatic correction of the position of the character controlled by the offending connection.

As it was desired to be able to kick the offending player, a custom event was added into the UCharacterMovementComponent in the engine source code. In this event, a pointer to the offending controller was passed to our Gamemode, which handled the event. While the method chosen is not overly complicated to implement, altering the game engine source code is suboptimal.

The goal of finding and implementing speedhack detection was a success. The solution found is also easy and quick to implement. While the system by itself does not provide an out of the box solution when it comes to handling these offenses, it is simple for game developers to expand on the solution to make it function in a way that benefits their project. Since finding the correct settings and enabling the system required a considerable amount of research, I hope that this work makes finding and implementing these solutions more accessible.

6.2 Further research

Further functionality could be developed to the handling of users caught cheating in SpaceFlight. This could include IP-bans, Hardware based bans and database for storing the required information outside the game instance.

The handling of cheating events can also be expanded upon. Currently, such events never expire, even if a significant amount of time has passed since the infraction. The notification screen is also never removed from the client.

The solution is only tested within local area network. This eliminates issues caused by packet loss and latency. The solution can be tested in conditions that simulate different quality connections and find sensible default settings for speedhack detection.

References

1. Alford A. Riot's Vanguard Anti-Cheat Remains a Security Risk. 2020 [Referred 27.1.2021] Available at: https://www.hotspawn.com/valorant/news/riots-vanguard-anti-cheat-remains-a-security-risk
2. Cheat Engine. About Cheat Engine. 2021a [Referred 26.3.2021] Available at: https://www.cheatengine.org/aboutce.php
3. Cheat Engine. Cheat Engine:Internals. 2021b [Referred 26.3.2021] Available at: https://wiki.cheatengine.org/index.php?title=Cheat_Engine:Internals#Speedhack
4. Counter-strike.net. The Trust Factor. 2021 [Referred 2.2.2021] Available at: https://blog.counter-strike.net/index.php/the-trust-factor/
5. Davison E. Aim assist in the crosshairs. 2020 [Referred 1.2.2021] Available at: https://www.washingtonpost.com/video-games/esports/2020/10/16/aim-assist-debate/?arc404=true
6. Epic Games. Next-Gen Hub. 2021a [Referred 1.5.2021] Available at: https://www.unrealengine.com/en-US/nextgen
7. Epic Games. UObject. 2021b [Referred 8.4.2021] Available at: https://docs.unrealengine.com/en-US/API/Runtime/CoreUObject/UObject/UObject/index.html
8. Epic Games. Property Replication. 2021c [Referred 5.4.2021] Available at: https://docs.unrealengine.com/en-US/InteractiveExperiences/Networking/Actors/Properties/index.html
9. Epic Games. RPCs. 2021d [Referred 5.4.2021] Available at: https://docs.unrealengine.com/en-US/InteractiveExperiences/Networking/Actors/RPCs/index.html
10. Epic Games. Actors and their Owning Connections. 2021e [Referred 5.4.2021] Available at: https://docs.unrealengine.com/en-US/InteractiveExperiences/Networking/Actors/OwningConnections/index.html
11. ESIC. Esports Integrity Coalition (ESIC) Announces 5 Year Ban for Player Nikhil "Forsaken" Kumawat for Cheating. 2018 [Referred 28.1.2021] Available at: https://esic.gg/esports-integrity-coalition-esic-announces-5-year-ban-for-player-nikhil-forsaken-kumawat-for-cheating/
12. ESIC. Esports Integrity Commission (ESIC) Statement Regarding Alleged Stream Sniping/Ghosting by Vitality in BLAST Premier. 2021a [Referred 4.2.2021] Available at: https://esic.gg/esports-integrity-commission-esic-statement-regarding-alleged-stream-sniping-ghosting-by-vitality-in-blast-premier/
13. ESIC. Members & Supporters. 2021b [Referred 4.2.2021] Available at: https://esic.gg/members/
14. ESIC. Code of Conduct. 2021c [Referred 1.2.2021] Available at: https://esic.gg/codes/code-of-conduct/
15. Faceit. How to Report a Cheater and more. 2021 [Referred 2.2.2021] Available at: https://support.faceit.com/hc/en-us/articles/115001314150-How-to-Report-a-Cheater-and-more

16. Grayson N. Top Counter-Strike Players Caught In Big Cheating Scandal. 2014 [Referred 19.1.2021] Available at: https://kotaku.com/top-counter-strike-players-caught-in-big-cheating-scand-1662810816
17. HLTV. FORSAKEN CAUGHT CHEATING AT EXTREMESLAND; OPTIC INDIA DISQUALIFIED. 2018 [Referred 28.1.2021] Available at: https://www.hltv.org/news/25118/forsaken-caught-cheating-at-extremesland-optic-india-disqualified
18. Humphries M. South Korea Makes Game Hacking Illegal. 2016 [Referred 19.1.2021] Available at: https://uk.pcmag.com/games/86425/south-korea-makes-game-hacking-illegal
19. Irdeto. Global gaming survey. [E-Book]. 2018a [Referred 18.1.2021] Available at: https://resources.irdeto.com/irdeto-global-gaming-survey/irdeto-global-gaming-survey-report-2
20. Irdeto. Grand theft gaming. [E-Book]. 2018b [Referred 26.1.2021] Available at: https://go.irdeto.com/ebook-grand-theft-online-gaming-cheating-problem/
21. Koskinas, P. /DEV/NULL: ANTI-CHEAT KERNEL DRIVER. [Blog] 2020 [Referred 27.1.2021] Available at: https://na.leagueoflegends.com/en-us/news/dev/dev-null-anti-cheat-kernel-driver/
22. Lehtonen S. 2020. Comparative Study of Anti-cheat Methods in Video Games. University of Helsinki, Department of Computer Science. Master's thesis.
23. Lien T. 'League of Legends' World Championship faces cheating allegations. 2012 [19.1.2021] Available at: https://www.polygon.com/2012/10/7/3471178/league-of-legends-world-championship-faces-cheating-allegations
24. Maiberg, E. Hacks! An investigation into the million-dollar business of video game cheating. 2014 [Referred 19.1.2021] Available at: https://www.pcgamer.com/hacks-an-investigation-into-aimbot-dealers-wallhack-users-and-the-million-dollar-business-of-video-game-cheating/
25. Matthews E. 37 CS:GO coaches have been banned for abusing the Spectator bug. 2020 [referred 19.1.2021] Available at: https://www.pcgamer.com/37-csgo-coaches-have-been-banned-for-abusing-the-spectator-bug/
26. Perry K. Call of Duty: Warzone – Are controllers viable on PC? 2020 [Referred 4.2.2021] Available at: https://www.pcinvasion.com/call-of-duty-warzone-are-controllers-viable/
27. Pinterest. 2021 [Referred [Referred 4.2.2021] Available at: https://i.pinimg.com/originals/74/55/da/7455da7fc72bf342b542e1f31e6eb711.png
28. Plunkett L. Accused League of Legends Cheaters Fined $30,000. 2012 [Referred 19.1.2021] Available at: https://kotaku.com/accused-league-of-legends-cheaters-fined-30-000-5950746
29. Rasmussen P. Tweet. 2019 [Referred 27.1.2021] Available at: https://twitter.com/dupreeh/status/1203494270851862528?ref_src=twsrc%5Etfw%7Ctwcamp%5Etweetembed%7Ctwterm%5E1203494270851862528%7Ctwgr%5E&ref_url=https%3A%2F%2Fwww.dexerto.com%2Fcsgo%2Fdupreeh-speaks-out-after-csgo-crowd-helps-astralis-in-epl-playoffs-1297096%2F
30. Rightberg J. Cheating In Gaming: Will Copyright Laws Level Up?. 2016 [Referred 19.1.2021] Available at: https://www.forbes.com/sites/legalentertainment/2016/09/01/cheating-in-gaming-will-copyright-laws-level-up/

31. Singh Rawat A. 8 types of hacks and cheats in CS:GO. 2019 [Referred 26.1.2021] Available at: https://afkgaming.com/articles/csgo/News/2619-8-types-of-hacks-and-cheats-in-CSGO
32. Statista. Online games – worldwide. 2021 [Referred 26.1.2021] Available at: https://www.statista.com/outlook/212/100/online-games/worldwide
33. Steam. Valve Anti-Cheat System (VAC). 2021 [Referred 1.2.2021] Available at: https://support.steampowered.com/kb/7849-RADZ-6869/
34. Symes A. Cheating in games: legality and market trends. 2021 [Referred 19.1.2021] Available at: https://www.lexology.com/library/detail.aspx?g=e8271099-771c-4854-946e-41f7bf37671b
35. Telimaa E. 2021. Proof of speedhack. https://youtu.be/UV8fe_MvWGw. 15.4.2021.
36. The Linux Information Project. Kernel Definition. 2005 [Referred 7.5.2021] Available at: http://www.linfo.org/kernel.html

SpaceFlight DefaultGame.ini:

```
[/Script/EngineSettings.GeneralProjectSettings]
ProjectID=048DF7C1475F088AB59B08AB696DE3FE

[/Script/Engine.GameNetworkManager]
TotalNetBandwidth=32000
; Min/Max bandwidth dynamically set per connection
MaxDynamicBandwidth=7000
MinDynamicBandwidth=4000
; ----------------------------
; Player replication
MoveRepSize=42.0f
MAXPOSITIONERRORSQUARED=3.0f
MAXNEARZEROVELOCITYSQUARED=9.0f
CLIENTADJUSTUPDATECOST=180.0f
MAXCLIENTUPDATEINTERVAL=0.25f
MaxMoveDeltaTime=0.02f
ClientNetSendMoveDeltaTime=0.0166
ClientNetSendMoveDeltaTimeThrottled=0.0222
ClientNetSendMoveThrottleAtNetSpeed=10000
ClientNetSendMoveThrottleOverPlayerCount=10
ClientAuthorativePosition=false
ClientErrorUpdateRateLimit=0.0f
; ----------------------------
; Movement Time Discrepancy settings for Characters (speedhack detection and
prevention)
bMovementTimeDiscrepancyDetection=true
bMovementTimeDiscrepancyResolution=true
MovementTimeDiscrepancyMaxTimeMargin=0.20f
MovementTimeDiscrepancyMinTimeMargin=-0.05f
MovementTimeDiscrepancyResolutionRate=1.0f
MovementTimeDiscrepancyDriftAllowance=0.0f
bMovementTimeDiscrepancyForceCorrectionsDuringResolution=false
bUseDistanceBasedRelevancy=true
```