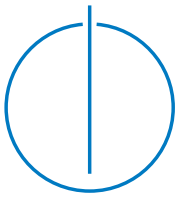
Department of Informatics
Technical University of Munich

Bachelor's Thesis in Informatics: Games Engineering

Anti-Cheating Measures in Video Games

Caroline Andrea Rendenbach



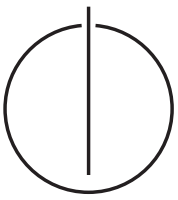
Department of Informatics
Technical University of Munich

Bachelor's Thesis in Informatics: Games Engineering

Anti-Cheating Measures in Video Games

Anticheatingmaßnahmen in Videospielen

Author: Caroline Andrea Rendenbach
Submission Date: Feburary 11, 2022
Supervisor: Prof. Gudrun Klinker, Ph.D.
Advisor: Daniel Dyrda, M.Sc.



I confirm that this bachelor's thesis is my own work and I have documented all sources and material used.

Garching, Feburary 11, 2022 Caroline Andrea Rendenbach

Acknowledgments

“This was a triumph
I’m making a note here; ”Huge success“
It’s hard to overstate
My satisfaction

[..]

But there’s no sense crying
Over every mistake
You just keep on trying
Till you run out of cake
And the science gets done
And you make a neat gun
For the people who are
Still alive”
-Still Alive by Jonathan Coulton from the Video Game Portal

I want to thank my advisor Daniel Dyrda and my supervisor Professor Gudrun Klinker for their support.

Abstract

Cheating in video games has seen an uptick since players do not have to buy magazines or books for cheat codes anymore but rather have hundreds of ready-made cheats at their disposal through one google search. This also now impacts more than just the cheater themselves as many video games have moved their services online and multiplayer has become an integral part of gaming. Video game developers need to provide anti-cheating measures with their game to ensure a fair playing experience. Developing an anti-cheat engine is a huge undertaking and knowing what kind of anti-cheat engine the game demands is hard to evaluate. This process is made easier by defining the constraints of the system before development. The thesis proposes questions from different areas to establish the basic framework of what the anti-cheat engine demands. We then present different existing anti-cheating measures which are put into the context of the constraints and assessed through them. Thus, this thesis supports developers in the process of designing an anti-cheat engine while evaluating existing measures. The work also proposes different repercussions players can face for cheating.

Contents

1 Introduction

The video game industry is continuously growing, reaching $178.73 billion in revenue worldwide in 2021 which is an increase of 14.4% compared to 2020 [105]. This is an unexpected growth and it is speculated to be a result of to the Covid-19 pandemic. With more people forced to stay at home and they had time to play video games. There was not just an uptick in sales, but also in forums where users come together to discuss ways to get around anti-cheating measures and even offer ready-made cheats [53]. Video games and the technology they use have evolved substantially in the last twenty years. At the same time cheating has become more ubiquitous, especially since video games have shifted a lot of functionalities online and games often have a multiplayer aspect to them. Additionally, often as soon as an exploit is fixed that cheaters used, they find a new way to exploit the system.

A survey by Irdeto shows that about 37% of players said they cheated at least once in online games, but also had the option not to say whether they cheated or not [46]. Thus, there is a possibility that the percentage is even higher. Anti-cheating measures are integral to the user experience because if the measures are inadequately developed then cheaters flock to the game. Even though working on and improving anti-cheating measures might seem challenging, it is crucial to do so in order to ensure that consumers play the game long term and spend money on microtransactions such as in-game items or currency. As shown in the survey, across multiple countries about 77% of users stated that they would stop playing a game if they believed other players were cheaters and 48% stated that due to cheaters they are less willing to spend money in-game [46].

This thesis aims to establish a clear picture of what cheating is and what tools are available to do so. With that in mind, we lay the groundwork that helps developers define the framework of their anti-cheat engine before starting the project. Based on the constraints we give the developers the tools needed to design their anti-cheat engine according to their expectations. Thus, this thesis supports developers in the process of designing an anti-cheat engine while evaluating existing measures.

At first, this thesis takes a look at what cheating is and what kind of cheats are commonly used in video games. By providing a definition and assessing what a developer is working against, the development of an anti-cheat engine will have a clear goal. As mentioned above the developed engine has an influence on the user experience, thus this work does not only focus on what is technically possible but also what fits in with good game design. To achieve this a definition of good game design has to be established.

Based on this definition, this thesis introduces essential questions that developers should ask themselves before developing their engine. The answers create a basic framework that the anti-cheat engine should fulfill. The constraints are split into different categories such

as general constraints which are questions about how the game is set up, ethical constraints which give the developers the option to think critically about what cheating is to them, user experience constraints establish how developers want users to experience the game and how anti-cheat influences it, budget constraints which are questions about how much resources can be spent and privacy constraints which establish how invasive the engine should be.

With the basic framework defined, this work introduces different anti-cheating measures which are evaluated based upon how they fit into the constraints. The measures are divided into technical options as well as options beyond the game and the *trustworthiness factor*. They can have the effect of either preventing or detecting cheaters.
Furthermore, this thesis presents repercussions that can be faced by cheaters after being detected by the anti-cheat engine.

2 Related Work

Research in this field has increased as the video game industry has become more relevant over the last years and the importance of anti-cheating engines in the development of games has been taken a step further. Most papers in this field concentrate on certain aspects: engines for a specific game, engines for a specific genre of video games, engines that try to solve certain cheats, etc. There are also papers that give an overview or survey of anti-cheating measures.

In the following exemplary papers are presented:
Ki et al. first give a classification of exploits known to be used by video game hackers and then explain the appropriate anti-cheating measures for each exploit. The attacks are classified by two factors: are they an attack with a certain objective such as acquiring hidden information and are they an attack done with a certain method such as software modification. This paper introduces a great number of solutions but consequentially every measure is explained less thoroughly. [56]
Webb et al. focus on cheating and measures against it for games that use peer-to-peer architecture and present the RACS (see Section 5.2.1). At first the differences between client-server and peer-to-peer architectures are explained. Then different cheat classification are introduced and evaluated, focusing on a classification based on what level the exploit attacks and giving the fitting solutions to the attacks. [104]
Lehtonen categorizes and evaluates different anti-cheating measures with distinct criteria by implementing them. The criteria are resistance to tampering, ease of implementation, lack of overhead, non-invasiveness and suitability for a wide variety of games. [64]
Woo et al. identify the main exploits used in video games, offer solutions to each one and then categorize them into where the solutions are applied (client-, network-transportation- and server-side). The exploits are evaluated in their severity of the effect on the video game company and on the user. The paper only considers solutions for client-server architectures as possible structures for video games. [109]

Most papers in this field focus on the technological advancements of anti-cheating measures but this thesis expands on that as these measures can have a huge impact on the experience of the user. Thus, ascertaining good game design while and before developing an anti-cheating engine can be crucial to the process. Moreover, the aspect of questioning what is relevant and what is demanded from the anti-cheat engine plays an important part in this thesis.
Relevant technical research is referenced when discussing individual anti-cheating measures in Section 5.2.

3 Basic Notions and Definitions

The video game industry has been an ever-changing space in the last decade. Thus, to establish common ground, we provide clear definitions of good game design and cheating. They are needed to establish what principles of game design apply to the design of an anti-cheat engine. This chapter additionally proposes a definition for cheating in the context of video games as it is different from cheating for example in an academic context. In doing so we also give an overview of different types of common cheats and classify them by how readily they are accepted by the average user.

3.1 Good Game Design

Because this thesis aims to help developers use anti-cheating measures that not only are technically feasible but that demonstrate good game design and lead to a great user experience, we first have to define what good game design means. Shell states that in good game design the developer wants to create an experience for the users, which engages them, has rules, can be won, surprises them, makes them curious and lets them solve problems, but does so playfully and for the sake of the game not because they are forced [85]. Problem-solving and its encouragement is also deemed one of the reasons why people might cheat. The players see a problem: the game got boring after a while, they can't reach a certain skill, etc. Thus, they try to creatively solve that problem through developing cheats. Thus, a game should be designed with the thought in mind that players need to be challenged often but also given the chance to improve without it feeling like a chore.

Competitive games should be suspenseful until the end and give players the sense that they improve through repetition [21]. These are crucial aspects that are destroyed by cheating. Playing with cheaters unknowingly and thus losing often can give the user the feeling that their skill has no impact on the outcome of the game and frustrate them. Knowing that other players are cheating destroys the tension of who will come out on top and bores players. If the surprise in the game is gone so are the players. This is why anti-cheating measures even in their most basic form are essential to good game design.

But too strict and harsh anti-cheating measures can also be counterproductive. Disturbing the game and user experience by punishing the suspected cheater, especially if it's a false positive, frustrates players. So there has to be a diligent assessment on how restrictive the engine should be. Another reason why this is important is to uphold the Flow. Flow is the concept that there needs to be a good balance between challenges and skills. Too much challenge (y-axis) and not having the skills (x-axis) to solve them leads to anxiety as seen in A3. Not having enough challenge and more skills than needed to solve them leads to boredom, as seen in A2 in Figure 3.1 [19]. The anxiety side is the relevant factor here as to while some challenges might be perfect for most they might be not for all. One of the reported reasons people are cheating is not progressing in the game. So giving the people

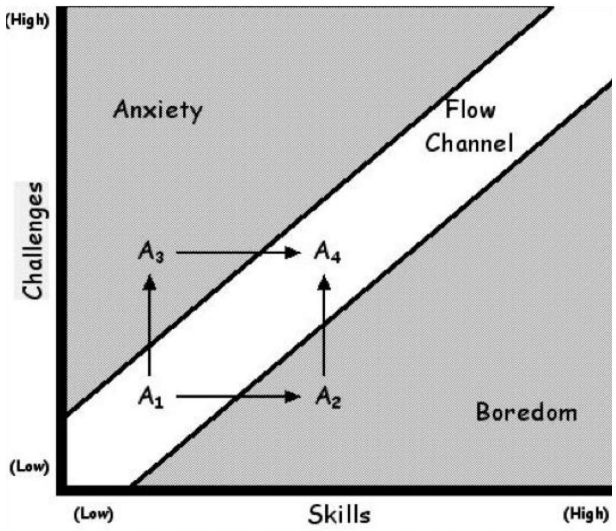


Figure 3.1: *Flow by Mihaly Csikszentmihalyi [19]*

the option to cheat in some way, meaning getting help from a friend, prevents feeling the anxiety from not advancing. The anti-cheating measures should preferably not interrupt the gameplay because as a developer we do not want to interrupt the user experience.

Aspects where the anti-cheating measures could be noticeable to the user is before taking more invasive measures meaning asking for consent. This is to not invoke the impression that they are under a general suspicion for cheating. The feeling of anxiety should not be invoked every time the player opens the game just because someone could watch every move they are doing. As such asking for consent can be crucial, options to ask are when purchasing the game and then always if more drastic measures such as scanning the memory are being taken as this is pretty intrusive. Not consenting can then lead to being seen as less trustworthy, but the decision is in the player's hands. The decision should be made reversible, so the user can opt-in and opt-out at any time.

3.2 Cheating

What is or is not considered cheating in the real world or video games is often a point of contention. The Cambridge Dictionary definition of the word cheating gives a great starting place: "to behave in a dishonest way in order to get what you want" [12]. This might seem clear, but there are a lot of situations up to interpretation. For example, when an adult plays a card game with a child, it would not be seen as cheating if the child got help from another child with choosing their next step but more likely as leveling the playing field. If an adult would do the same, it might be seen as cheating. Often there are no clear rules that can be applied to any and all situations. It could be said that by participating in social situations everyone signs an unwritten contract that states they will adhere to social norms and be fair. By breaking this contract people open themselves up

to consequences such as being excluded. But this contract too is up for interpretation as social norms depend on culture and upbringing.

The same applies to playing video games. What does or does not constitute cheating is even harder to decide and this often lies with the developer. If players gain the impression that actions which they consider cheating are allowed and it ruins the game for them. On the other hand actions they do not judge as cheating are reprimanded, might make developers lose out on customers. The socially accepted definition needs to be refined depending on the impact on the user experience. Looking at Consalvo's three perspectives on cheating [18] we can get a clearer picture of what can be considered acceptable:

The Purist	**Code is Law**	**You can only cheat another player**
Any outside help is cheating, which means anything other than the solo effort is an ethical compromise for the purist	Walkthroughs and anything that does not modify the game is allowed, the user should experience the game as the developer intended	Here is the intent or purpose relevant, because cheating only exists in relation to other players eg. stealing

But when examining these definitions problems appear: looking at the purist's version, disabled people who often use keyboards while playing on consoles because of (physical) limitations which might not be the intended way, if developers did not account for it. A purist would already consider this cheating but looking at it from a moral point of view, it is hard judge this cheating. Analyzing the "code is law" perspective, games often have quick-time events where you have to press a button in a short amount of time, but people with a bad reaction time will always get stuck at that point in the game. Slowing the game down would not be as the game developers intended, but some might feel like that everyone should get the opportunity to play through a game. The last perspective hinges on what is deceiving another player: would it be deceiving if a player exploits a bug that is part of the game to get a rare item?

Another definition can be given but focusing on what is not cheating and it is a bit more general: "the ethic of pursuing one's happiness as long it does not directly hamper another's possibility to pursue their happiness" [58, p. 33]. Kimppa et al. give a definition that mostly applies to multiplayer games because if a player is cheating in a single-player game, they are only deceiving themselves.

3.2.1 Types of Cheats

Before we can give a clear definition of what cheating is, we first need to take a look at what commonly is advertised as a cheat. The following list tries to give an overview over the most used cheats out there to help gain an understanding of what cheaters can achieve technically, thus painting a picture of what an anti-cheat engine might want to restrict.

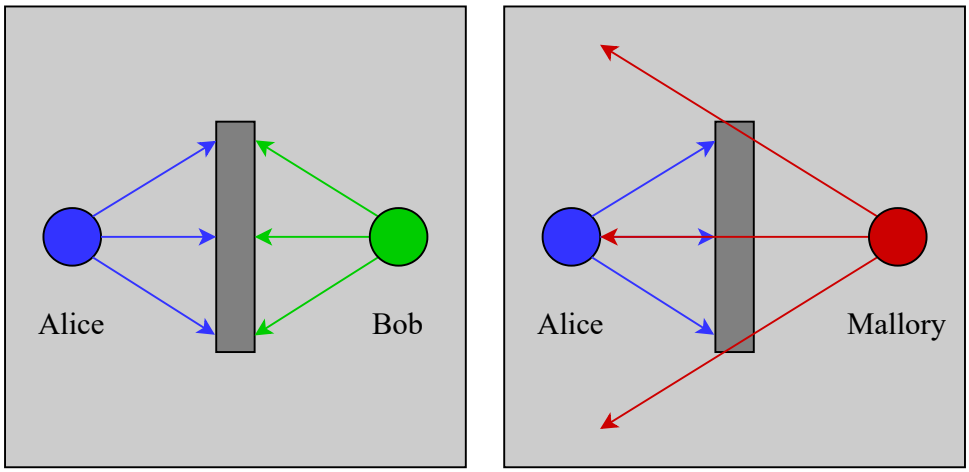


Figure 3.2: *Playing fair and playing with a wallhack*

Wallhacks To compensate for network latency the server often shares certain information with the client even if the client should not be able to gain access to the information yet in the game. For example, the enemy's position will be told to the client even if they are behind a wall to make sure that if the player turns the corner and will be able to see the enemy, the enemy will then be rendered. Wallhacks make use of this information and will render other players or game objects even if the client should not be able to see them. In Figure 3.2 on the left Alice and Bob play the game as intended and the client does not reveal that another player is behind the wall. This is visualized through the arrows which depict the viewing angle. On the left the arrows stop at the wall because Alice and Bob are not able to see through the wall. In contrary to Mallory who is able to see the wall using the wallhack. On the right, Mallory plays with a hacked client using a wallhack thus knowing Alice is behind the wall and gaining an advantage.

Aimbots Multiplayer online games often have as the main task hitting the enemy with a weapon in some type of form. For example, in First-Person Shooter (FPS) games the player has to shoot the enemy with a gun. Hitting a moving target can be challenging thus some players might feel the need to get help from aimbots. These cheats cut out the inaccuracy of the player by knowing exactly where the enemy is,. This is done through accessing the games' memory and shifting the player's aim to the position of the enemy. The most rudimentary aimbots do this by harshly correcting the aim to the exact position, more advanced programs try to imitate more natural player movements while correcting to the right position thus are harder to detect.

Not allowed movements Every game has ways that players can move around in the game and ways that they cannot. Most games allow players to move on the ground and jump up to a certain height. Many hacks read out the player's position in the game and write different positions in the memory or send a different position to the server. If these

positions are not cross-checked against the game logic and the last known position, players can jump higher, fly or teleport through the game.

Gold farming Increasingly more people offer their services to players who do not want to do the repetitive work or do not have the time to play every day to progress in the game and gain in-game currency. Instead, these people pay money to third-party companies for them to play. The gold refers to the items or Often games like World of Warcraft, where players need to log on often and play for long hours to advance, are a target for these services. For most games, sellers can be found on the internet selling in-game items, currency, or accounts with higher levels. Until recently, when eBay decided to ban these types of sales, reasoning that this type of account sharing is not allowed by gaming companies, the grey market for these kinds of services was huge on the website [22]. One great example of this is Manfred, a guy who lived 20 years solely from the income of hacking in Massively Multiplayer Online Role-Playing Games (MMORPGs) as well as selling the currency and items gained through that exploiting the game. At first, exploiting unintended behavior in code in Ultima Online, then being able to delete other players' houses or logging in and out of the game Dark Age of Camelot without the game servers correctly handling the events and thus duplicating his character and all his items [67].

Replay attacks Replay attacks are not specific to video games, but a known exploit in network security. It is the act of eavesdropping[1] on packets sent when certain tasks are performed, then replaying the packets which means sending the same packets again to perform the same task without the authorization of the legitimate user [96]. This could be used when making an in-game trade with a seller, replaying the packet where the player receives the item, manipulating the game to think the player received items multiple times without them having to spend the needed currency for the items.

Time cheats As mentioned in the Wallhack paragraph, in multiplayer online games latency can be a significant problem. Players from all over the world want to play together but everyone has a differing internet speed. Thus, often players lag[2] because their internet is not as good. Time cheats use this to their advantage and fake the latency, giving the cheater an advantage. One such time cheat is the lookahead cheat. Using the lookahead cheat the player waits for the action of another player and then responds accordingly but sends that response in a packet with a timestamp before the action was taken by the other player [34].

DoS (Denial of Service) and DDoS (Distributed Denial of Service) attacks DoS and DDoS attacks happen frequently in most technology but also in video games, a report from Imperva states that DDoS attacks in the gaming industry and others have risen by 100% in 2021 and these attacks do not cost much. "A 100$ is enough to cripple

[1] Eavesdropping on packets here means to listen in on the packets that are being sent without being one of the legitimate communication partners

[2] Lag is when there is a delay in data packets being sent and players take seemingly a longer time to react than regularly

a network." [45, p.=4] A DoS attack makes an application unavailable for a certain user on purpose and a DDoS attack does the same for a large sum of users or brings even the whole network down. Denying other players the option to play during certain events or mid-game, can give the attacker an advantage such as outright winning the game through disconnecting other users.

Phishing Phishing again is not a problem only specific to video games. Generally, phishing tries to get sensitive information from a person by imitating a trusted third party [47]. Phishing in video games mostly looks like this: users get an email that seems to be sent by the video game company and the email states that there is for example a problem with their account, they need to click on this link and log in with their credentials. But unbeknownst to them, the email is from a malicious actor and they now have their login information to the account. Accounts gained through phishing could be sold off to other players if it has a high rank and items or in-game currency could be sold off as seen in Goldfarming. Attackers could also try to gain access to any payment details that are connected to the compromised account.

3.2.2 Summary

Figure 3.3 consolidates what kind of cheats are mostly accepted to which are not. Moving into the red zone cheats might be illegal and thus average players will be deterred to use these kinds of methods due to the fear of the repercussions. Wallhacks and aimbots are a legal grey zone: buying/downloading these cheats is not illegal per se, but will most likely get the cheater banned in most games and makes them vulnerable to getting hacked themselves (e.g. Trojans). On the other hand, engineering those hacks and making them available requires reverse-engineering the game which is not allowed through the End User License Agreement (EULA). Phishing and hacking game servers are illegal in most countries.

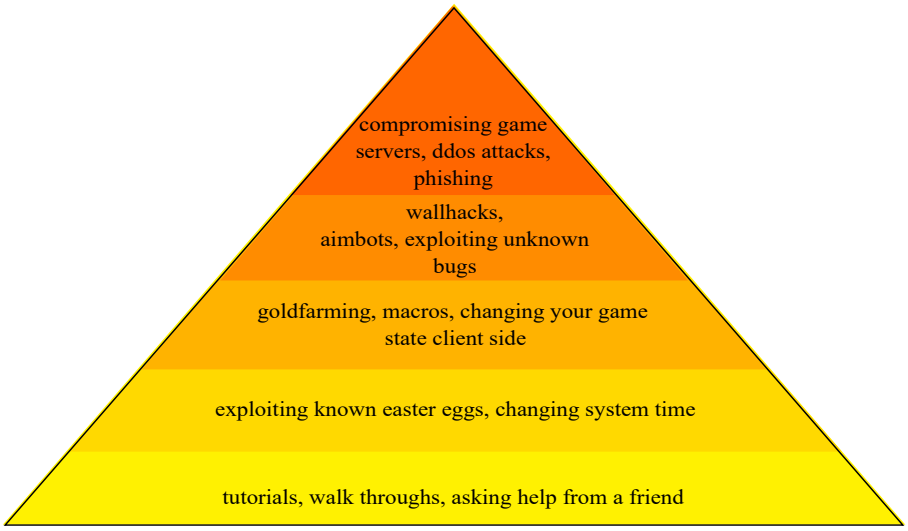


Figure 3.3: *Scale goes from yellow (commonly accepted) to red (commonly not accepted)*

All developers have to assess for themselves what is appropriate to them but also what is possible to restrict. A Purist perspective is just not realistic to implement as it is just impossible to detect if someone watched a tutorial. Moreover, there needs to be a consciousness that restricting some kinds of help does hinder disabled people from accessing the game. This applies to macros or using a mouse and keyboard in a console game. Reflecting on the cheats, for this thesis we find tutorials, walkthroughs, etc. and exploiting known easter eggs[3] (such as "motherload" in Sims which gives the user great amounts of in-game currency), changing system time to be acceptable and also not realistic to track. Everything else that is in the pyramid and everything that is like it, is considered cheating and the proposed framework tries to restrict that behavior.

[3]Easter eggs are cheat codes, hidden messages or hidden areas in video games left by developers

4 Defining the Basic Framework

There are many kinds of different anti-cheat engines already out in the industry and big gaming companies often develop an in-house solution that can be licensed for example by indie game studios [25]. Not every studio wants to use solutions that they have not developed themselves, but developing an anti-cheat engine might seem like a challenging task and it is hard to find a place to start. This chapter aims to give developers the tools to define basic principles of their anti-cheat engine. By asking relevant questions in different fields we provide the project with a clear structure before coding starts. Through this, studios can be more successful in developing an engine that fits their standards.

4.1 General

First, developers need to define a clear structure of the game.

- Online or offline?
- Single or multi-player?
- Player versus player, player versus machine, team versus team?
- What platform? Can it be played cross-platform?

Cheaters in offline and single-player games might be less of a concern for developers as to when they cheat they only ruin the game for themselves. Developing an online and multiplayer game means the possibility, that the game could be ruined for more users than just the cheater, is higher and more infrastructure than just the game files are open to exploitation. Depending on what platform the game is being developed in, meaning which engine is used and what platforms it is being developed for, there are different ways the game could be exploited. This should be taken into account during development. For example on most consoles the user does not have admin rights and is limited in what they can do, thus it might be harder to exploit the game as more knowledge in hardware hacking is needed. This does not mean people have not managed to crack consoles: poobs4 is a kernel-level exploit for the PS4[1] that works on firmware version 9.0.0. Two weeks after the exploit was made public Sony already rolled out an update that fixed the loophole that the exploit used [106]. Even with knowledge in hardware hacking console hacking has become a tremendously hard task with consoles such as the Xbox One because it is highly locked down. For example the plaintext of a game on the Xbox One only exists on a tiny chip which was specifically designed to ward off hardware hackers. Everywhere else on the console the game is encrypted and cannot be decrypted by the user [17].

For PC gaming it is a little easier as there are more tools for privilege escalation[2] and more ready-made cheats available. Sites like UnKnOwNcHeAtS, GuidedHacking or Aimware

[1] The PS4 is the fourth iteration of the Play Station, a game console developed by Sony

[2] Privilege escalation is the act of exploiting a flaw in software or an operating system to gain higher privileges (e.g. admin access)

have cheats or tutorials ready for players to use for free or in exchange for money. In mobile gaming, there are also a lot of tools available to hack apps on Android systems sometimes even without rooting[3] the phone for example Freedom Apk, Game Guardian, or CreeHacks [48]. Whereas it can be harder for iOS as it is often needed to first jailbreak[4] the mobile device. This can void the warranty which might deter people. It often also makes a difference if players from different platforms can play together. PC users might have the advantage of less latency as a PC can be upgraded individually and console players might have faster reaction times with a controller.

- What inputs are allowed?
- What structure if it is online?
- Do users have to make an account to play?

Depending on what platform the game can be played on, it is also important to know what inputs can be used. If the game is available on PC can it also be played with a controller? Can a console game be played with a keyboard and mouse? For accessibility reasons, developers might want to allow keyboard and mouse or special tools to play console games, as people with limitations might not be able to hold and use controllers. Most games that are developed for gaming consoles do not support keyboard and mouse input, but there are adapters like the GameSir Aimbox VX that translate keyboard input into game controller input [37]. This can give players an advantage over others but is very hard to detect with anti-cheating measures. Similarly, there is the Xbox Adaptive Controller made for players with mobility issues with big programmable buttons and it can be individualized with different joysticks and switches [72]. Controllers like the one developed by Xbox can both solve the issue of accessibility and fairness between players. Knowing what inputs can be expected helps with detecting anti-cheat because unexpected inputs could be a sign of cheating.

If the game is multiplayer and online there are different kinds of structures the game can use to communicate information about the game from and to all game clients. There are two kinds of popular structures: the client-server model (see Figure 4.1) or peer-to-peer architecture (see Figure 4.2). In the client-server model clients request information from one main server, that answers all requests from clients [11]. In peer-to-peer networks most clients have the same rights and information is distributed on all clients which communicate with each other [20].

- Does the game have a payment structure?
- Does the game have in-game transactions?
- Is there an in-game economy with in-game currency?
- Can players trade in-game items?

Payment structure here means: the game could be free-to-play, a one-time fee or a monthly fee. Sometimes the basic game is free-to-play but for certain services or for a premium

[3] To root a phone here means to escalate privileges through exploitation on a smart phone

[4] Jailbreak here in the context means using a privilege escalation exploit to root a phone thus removing software restrictions

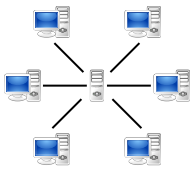


Figure 4.1: *Client-Server Model by Mauro Bieg [7]*

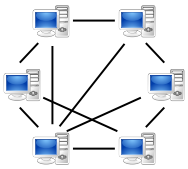


Figure 4.2: *Peer-to-Peer Model by Mauro Bieg [8]*

account, there is a fee. This is relevant because cheaters might be deterred from cheating if they already spent money and effort on a game and would lose all that if they got banned. Free-to-play games are more prone to have cheaters because after a ban they can just make a new account. If the accounts of users are linked to their payment details, there is also more pressure on the developers to ensure that this data will not be breached by hackers. Having in-game transactions such as the player with the game or players with other players opens up the game to exploits because users want items without having to pay the respective price or cheaters will try to farm (see Goldfarming in Section 3.2.1) items for monetary gains. In-game currency comes with the same problems especially if it can be bought or exchanged for money and not just won by playing the game well.
To protect players developers can enable different kinds of authentication. There are three types of authentication: knowledge such as passwords, property such as a smartphone or a physical card and biometrics such as a fingerprint. Single Factor Authentication uses one of these authentication methods. A very common version is Two Factor Authentication which is the combination of two authentications for example a password and a smartphone. Multi Factor Authentication combines more than two authentications. [77] Cheaters will also try to undermine the in-game economy by selling their currency, which they got by exploiting the game, at a lesser price than it can be bought in-game through sites like eBay.

- Can the game be played competitively?
- Is there a leaderboard?
- Does the game have contests?
- Is the game suitable for e-sport?

When a game can be played competitively, meaning at the end a game can be won and users level up, where the users level, rank or position in the leaderboard can be compared,

cheaters want to achieve that notoriety without having to put in the respective effort to keep up appearances or gain status [23]. The same can apply to contests especially if something of value can be won at the end. If the game should be suitable for e-sports and is being developed with that in mind, the results of games need to be trustworthy similarly to other sports tournaments as there are often big prize money pools and sponsors involved [31]. For example in 2020 two professional Valorant players Phox and W3ak were banned after the anti-cheat engine Vanguard suspected them of cheating just before the e-sport tournament PAX where the prize money was set at $25,000, thus ending their career as Valorant players indefinitely [76]. Banning the players before the tournament was also crucial to uphold the integrity of the tournament.

4.2 Ethics

- What is considered cheating?
- Is using the game as it is not intended by the code allowed?
- What should players do that do not progress in the game?

Although there is a definition of cheating established in Section 3.2 that does not mean it is applicable to all game developers' ethics. Some developers might have a harsher outlook on what is or is not considered cheating. Others might not even want to go against cheaters that strictly. They might view the act of cheating just as creative problem solving and because everyone has access to the internet and could also cheat still as fair. But developers must be aware this has an impact on user experience and users that do not subscribe to the same ethics might not play the game.

If most cheating is not allowed, developers need to face how they expect players to react if they do not progress in the game. Because it is highly likely that players might get stuck and cheaters indicate that being stuck is one of the reasons why they cheat [23].

4.3 User Experience

- What do players expect from the game?
- How are users affected by cheating?
- How much cheating can players tolerate?
- Are the players willing to be part of the anti-cheat system?

Relying on players for reporting cheaters or participating in something like "the Overwatch" (see Section 5.3.4) too heavily might feel like the burden is all on them. If there seems to be no changes after reporting the potential cheater, it might frustrate players even more than just playing with cheaters. But providing the opportunity for players to act proactively against cheating is seen already as a must as the report functionality is widely established in most games. Most players can live with some cheating especially if is not noticeable meaning a player only cheats to be a little better than the average player and all other

players still have or feel like they have a chance at winning. But some players who take the game more seriously might feel like all forms of cheating should not be possible.

- How much should the anti-cheating interrupt the experience?
- Should cheating be allowed on certain servers?
- False positives vs. false negatives

Some games allow cheating on special servers where everyone knows beforehand that cheating takes place. This can be a great way to let players try a cheat out without feeling guilty and not hurting others. A cheat server can also come with downsides as people might be encouraged to find cheats even if they wouldn't have before. Thus the demand for cheats might be higher and more people would try reverse-engineering the game and even try it on servers where cheating is not allowed.

Both false positives and false negatives while detecting cheaters can negatively impact the game: false positives where a non-cheating player is punished for cheating will lead to players leaving or being forced to leave the game who are innocent. False negatives might lead to players that are frustrated by the cheaters and thus might leave the game. Thus, developers need to make a weigh the consequences of what risk they are more willing to take.

- Are cheaters punished?
- Can players do something to appeal a punishment?
- How is feedback about the system handled?

Developers also need to decide if cheaters face punishment and if yes what kind of punishment. Chapter 6 has some suggestions on what consequences cheaters could face. Giving the players the opportunity to appeal the decision is the more ethical way and will give players the impression of a fairer process but also might exhaust resources that could be spent on maintaining and improving the game. Feedback can be crucial to improving a system but developers need to differentiate between constructive criticism and cheaters that are disgruntled about being banned. Waiting too long to respond to feedback if it is a bigger problem, might result in unwanted negative attention on social media.

4.4 Budget

- How many resources can be dedicated to anti-cheating?
- Losing players because of rampant cheating vs having fewer resources for the rest of the development of the game
- Are there resources to develop anti-cheating continually even after the game is released?
- Are there human resources that can verify cheating?

What kind of budget is available highly affects the quality and extensiveness of the anti-cheat engine. Sometimes indie game developers make the mistake of treating anti-cheat

as an afterthought, either only implementing the bare minimum or even abandoning it completely because the budget is tight. Indie hit Fall Guys made that mistake and players reported that the game had high numbers of cheaters and made it unplayable for non-cheating players [80]. As not to lose all players suddenly the developers needed to implement better anti-cheating in way less time than they might have had if they had done so in the development stage. Implementing in such a hurried way might also consume more resources because if the cost is calculated as the product of the quality of the development, time and scope of the job, then the time greatly factors into the cost [1].

Similarly to anti-virus scanners, anti-cheating is a continuous effort, because cheat developers always look for new ways to exploit the game. Thus, in the budgeting phase it should be acknowledged that after the release there should be resources allocated to continually work on refining the measures. Not doing so might lead to frustrated players in the months after the release of the game. It should not be assumed that the anti-cheat system is perfect, hardly any system is. Thus, before a player is permanently excluded from playing the game, the suspected cheating might need to be verified by a human to lessen the risk of false positives. But these are reoccurring costs that need to be allocated in the budget.

4.5 Privacy

- What data will be collected about users?
- Should that data be saved long term?
- How is the data intended to be used?
- Are users informed about what and when information is collected?

Data that is collected about users can often be used to reveal private information about them. Thus, developers should be aware of the responsibility they have. It needs to be assessed what information is actually needed for the anti-cheat. Moreover, countries have laws about what can be collected and how long it can be stored. The United States has the Federal Trade Commission (FTC) Act, Section Unfair and Deceptive Trade Practices that states consumers have to be informed what data is collected and how it is used [103]. Failing to comply and thus misleading customers by not maintaining the security of sensitive information or causing serious injury for the consumer, is persecuted by the FTC and law enforcement [33]. There are other laws in the US that could apply to the game: the COPPA (Children's Online Privacy Protection Act) applies if there are players known to be under 13 parental consent has to be acquired before data is collected, used and parents are allowed to see what data is collected [73] or the California Consumer Privacy Act. The European Union has the General Data Protection Regulation (GDPR) and this applies if a company supplies goods or services to anyone in the EU. EU citizens have the right to see what data is collected about them and can request that this data is to be deleted. It also calls for an easy-to-read and understandable privacy notice [38].

Having a server-side only anti-cheat would mean that game logs and their metadata would be collected which is only game related data, client-side only anti-cheat can highly differ in

what data is collected. Here it really depends on how the engine operates because depending on how invasive it is, it has more access to more information. There is the opportunity to not only collect game related data but also general user data. This information could be useful to detect cheaters but could also be used to analyze users' behavior even outside the game. For example, just analyzing the writing of the user can reveal what gender the user probably has, with that information one could target advertisements better [98]. Free-to-play games like World of Tanks already use the data they collect while a user is logged in, mentions of the game in social media and on gaming forums to retain customers, convert them into paying customers, sell them other games and more [43]. Thus, if the privacy of users is valued by developers, data collection should be limited to what is definitely needed. It should be evaluated what is needed as not to expose information about users. Information can be given to users at the same time as the Terms and Conditions of the game, inside the game and on the game's or company's website. The more transparent and visible the privacy policy is, the better users might be able to make an informed choice about downloading the game.

- 100% security vs 100% privacy
- How are users protected from data breaches?

As a developer it is important to gauge where the company stands on the spectrum between privacy and security. One end of the range is maximum security, i.e. ensuring that there are definitively no cheaters but in doing so giving the users no privacy. On the other end, the focus is on ensuring that the user's privacy is never mishandled through never collecting data on players but thus running into the risk of having cheaters. Between the extremes, there is also a middle ground. Wherever the company or the game lands on the spectrum in the end should be communicated to the consumer.

Data breaches here mean the act of finding some way into a system and steal data. In the case of video games this could mean that a singular user is breached. For example their account password got phished, their computer was taken over by hackers or the company's network is breached. Hackers could then steal data about users such as e-mails and passwords, source code or other important information about the company's inner workings. In 2021 the video game company Electronic Arts (EA) was compromised and the source code to FIFA21[5] was stolen. According to the company user's privacy was not threatened [29]. Also in 2021 CD Project, another video game company, was breached and the company suspects that information about video games that currently were being developed and personal data of employees were stolen [14]. User information like e-mails, passwords and other sensitive information of League of Legends[6] players was stolen in 2012 according to Riot Games [42]. These are just some examples of data breaches in the video game industry. This shows that it is important that if a company collects data about users they need to have security measures in place.

[5]FIFA is a popular soccer video game developed by EA
[6]League of Legends is a popular Multiplayer Online Battle Arena (MOBA) game developed by Riot Games

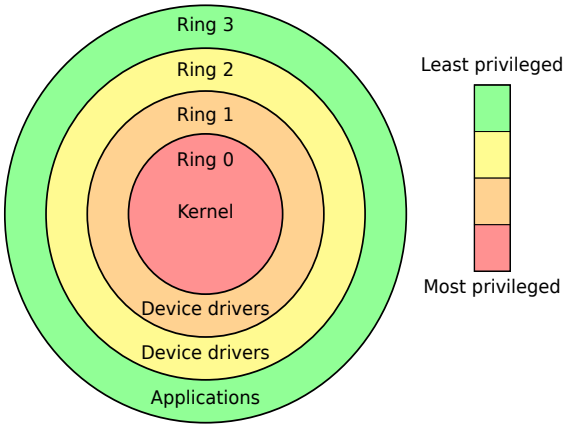


Figure 4.3: *The privilege rings by Hertzsprung [41]*

- Ring 0 vs. ring 3?
- Should consent from users be asked before more invasive measures are being taken?

Ring 0 or kernel level anti-cheats are measures that work at the level of the operating system in a computer. Figure 4.3 shows what this can look like for a x86 CPU, a game for example works solely on Ring 3 privileges as do most other applications. How the privilege rings are set up is enforced by the CPU. Commonly kernel-level or ring 0 means the highest privilege level. Having Ring 0 privileges' means that a driver can access and do anything [74]. Operating anything on kernel-level comes with risks and only highly trusted drivers should be given that privilege because the kernel and hypervisor reside there. The potential to compromise the user's system is high because with ring 0 privileges the driver has access to important parts of the operating system and mistakes in code or malicious code left by bad actors in the company could lead to crashing the users' PC. These kinds of anti-cheat engines can also send all data collected on the user's computer to the company as such users need to have huge trust in the company to not collect or even sell their private information. Even collecting this kind of data could land a company in trouble if and when they have a data breach.

But there are not only downsides, there are also reasons why developers use such a measure. First and foremost client-side anti-cheat measures that operate in the user space are limited in their ability to detect and protect from cheating. Cheat developers already work at ring 0 privileges as to be undetected from the anti-cheating at ring 3. At ring 0 the driver has full access to memory, hardware, all CPU instructions and critical operating system infrastructure. Thus, a kernel-based anti-cheat driver can ban other drivers that are part of cheats from working or stop processes that are commonly used by cheating programs and collect information about suspected drivers to send it to a server to further analyze. This makes the anti-cheat more powerful, but if the privacy and security of the user is highly valued one should refrain from using this.

Another option to both have the more invasive measures but also respect the user's privacy, is to first ask for consent before using these. One way to implement this could be to tell the user they are suspected of cheating, that their next matches are more closely being watched and if they consent to other measures like scanning the user's memory. The user could now disable their cheating program but there is a high chance of a noticeable difference in gameplay. It also would be important to monitor more than just one game because one game might not reflect the average gameplay of a user.

To summarize, in this chapter we established basic principles through different questions that can be used to ease the process of designing an anti-cheat engine because developers know what is demanded of the system before starting to code. There are four categories of questions. At first, there are general constraints to asses how the game is structured, then we address the ethics of the studio to establish to what extent cheating is tolerated. With the user experience constraints, developers assess what the expectations of the users are for the experience of the game and what experience the developers want the game to support. What kind of budget is available to the developers for anti-cheating directly impacts the quality of the engine, thus we have questions that establish how much budget is available. Lastly, this chapter questions the developers' stance on privacy and how developers will protect their users from data breaches.

5 Anti-Cheat Design

Knowing the requirements of the anti-cheat engine is only the first step towards its development. This next chapter focuses on what is technically feasible and what options are suitable for the constraints that were evaluated in Chapter 4. At first, we take a look at the technical options available. Some measures need certain architectures such as client-server or peer-to-peer while others try to solve specific problems such as the lookahead cheat or packet tampering. Not only can technology be utilized for anti-cheating but there are many more options beyond the game that for example involve support from humans. After both subsections, we give a brief overview of the proposed measures and their evaluation based on the constraints. Finally, we take look at the *trustworthiness factor* as a tool for ranking players in terms of suspicion that they are cheaters.

5.1 Overview of Existing Anti-Cheat Engines

Before looking at the measures that could be taken it should be evaluated what is already out there to a comparison. Though companies closely guard what information is released to the outside, we can get an idea of how these software programs operate. This is not a complete list but gives a general overview for comparison.

5.1.1 Valve Anti Cheat (VAC)

VAC is an engine developed by Valve and is a component of Steamworks and Steam. Games such as Counter-Strike: Global Offensive, Left 4 Dead and Call of Duty use VAC [93]. Valve states that it works like an antivirus program, scanning the computer of a player for known cheating software. Once a cheater is detected, they are marked for a delayed ban [94]. The ban is delayed in hopes of not revealing to the cheater what tipped the system off. VAC also utilizes deep learning with their new software VACnet. When this software detects cheating, players are submitted to be reviewed through Overwatch, which is explained more thoroughly in Section 5.3.4.

5.1.2 Vanguard

Vanguard is a relatively new anti-cheat engine developed by Riot first and foremost for the game Valorant, which was developed with e-sports tournaments in mind. Vanguard is different from most anti-cheat systems because Vanguard runs as soon as Windows is booted if Valorant or other games that use Vanguard are installed or if Vanguard is not uninstalled. Meaning this software is intrusive, which makes it controversial. At first, Vanguard even blocked other drivers that were not cheat applications but for example just monitored temperature [108]. Now Riot has scaled back to allow most such software [110] and gives users the option to disable Vanguard while not running the game. This is also relevant because uninstalling Valorant does not mean Vanguard is uninstalled too [95]. Vanguard works as a kernel-mode driver. This means it can ban anything it wants and

see all programs that are running. Riot explains that this is needed because cheats are being developed at the kernel-level and will not be detected if the anti-cheat engine is not kernel-level too [108].
It must be noted that these ring 0 anti-cheat engines have a bad reputation, due to previous scandals. For example, ESEA anti-cheat released malware into their kernel-level program and forced users to mine bitcoin. According to co-founder Craig Levine, they were testing bitcoin integration and an employee was using the test code for their own gain [84].

5.1.3 Easy Anti-Cheat

Easy Anti-Cheat is an engine used by games such as Fortnite, Dead by Daylight and Rust and is an Epic Games company. Easy states that they believe mass penalization and invading privacy will not fix the cheating problem. They deem Easy Anti-Cheat to be a noninvasive software. It states that it is a hybrid with kernel-driver code and machine learning [25]. This means it partly operates client-side and partly in the cloud. Moreover, it searches for corrupted or unknown game files as well as untrusted system files. In order to prevent reverse engineering, it notices if the game is launched with a debugger or kernel debugging is enabled [26]. This means it is also a kernel-based anti-cheat driver, thus can be as intrusive as Vanguard.

5.1.4 Fair Fight

FairFight is a server-side anti-cheating software developed by i3D.net, a Ubisoft company. It states to be a noninvasive engine that analyzes player statistics and compares them against average players from the game. It is completely server-side. Cheaters get an automated penalty if a certain threshold is overtaken. Penalties are escalated in a graduated penalty system: warning, restriction and suspension. Cheaters are warned that a continuation will lead to harsher penalties. FairFight is used in games such as Battlefield V, Titanfall 2 and Rainbow Six Siege. The software allows developers to customize the rule sets that are used to determine whether cheating took place or not [99].

5.2 Technical Options

5.2.1 General

Writing good code This might seem obvious but is often forgotten in the context of anti-cheating. Bugs that go unnoticed in the development phase and even later are the first point of entry for hackers. Writing good, bug-free code is the first step to protecting a game.

Goodliffe states that there is code that works, given expected inputs the code does what is needed. Correct code will not crash when receiving unexpected inputs but still is not good code. To write good code we have to assume the worst about the code: the code will someday at some point throw errors or go wrong. Unwritten assumptions about how the code should work will cause faults. Using this concept, code can be written in a

defensive programming style, meaning code is written with the prevention of bugs and invalid assumptions in mind. This kind of code might need more resources but will save time while debugging and it has to be always assumed that code will be used wrongly, if intentionally or not. First and foremost code should be written readable and have a good structure. Keeping it simple, code that is elegant but hard to understand will introduce problems. Input should not be trusted unless it is validated: even libraries can have faults. Code and variables that will only be used internally should not be able to be accessed from the outside. Always check if data will fit into buffers as not to write past the end of the buffer, this is often exploited. Release memory as soon as it is not needed anymore and do so cleanly. Memory is a scarce resource and should be handled accordingly. Initialize variables when declaring them to prevent forgetting about them or in the case of C or C++ using them with random values. What versions of language and software are used should be clearly defined and known to everyone. Unwritten assumptions should be turned into writing through code that checks if each condition is met. [39]

Having done all this, a huge part of validating that the code is bug-free is to test it. Every possible action that could be done to the code has to be tested: playing the game as a genuine user, playing as a malicious actor and more.

Not trusting the client Most multiplayer online games have a client-server structure. Therefore, it is of the utmost importance to verify the game data that is sent from the client. Not checking if the data sent is correct can lead to many problems such as players teleporting through a map, getting money without selling money, killing an opponent without having hit them. This means more resources for servers and networks are needed, obviously, thus it cannot be done for every little instance but for most to secure the integrity of the game. But due to cloud computing costs can be reduced, which makes it more possible and gives fewer reasons as not to have such an integral part of anti-cheating.

Having a client-server infrastructure calls for appropriate packet handling. Meaning the packets and the protocol need to meet three of six goals of cybersecurity: confidentiality, integrity, and authentication[1] [63]. Confidentiality means that information must be concealed, so any user cannot read and understand information. Integrity is the accountability that changes to data will be noticeable. Authentication is knowing that the other communication party is who they say they are. These goals are also important in video games. We need to make sure that a malicious actor cannot read what is written in the packets sent from and to the client because that could give them information where they need to manipulate the packets, which is covered by confidentiality. (Data) integrity is certainly important, here specifically to ensure that if a player manipulates a packet it will be noticed by the server. Authentication here is needed to stop Man in the Middle attacks, which are attacks where a bad actor sits in-between a valid communication partner and an application to either sniff packets[2] or to even imitate the valid partner. This is relevant here for example to

[1] The other three goals are availability, accountability and controlled access
[2] Sniffing packets means to listen in on communication without being noticed

prevent hackers from intervening with innocent players' gameplay or even stealing their payment information [66].

These goals can be enforced with the protocol Transport Layer Security (TLS) which is a widely used protocol for client-server communication because it prevents eavesdropping, manipulation of packets and forgery. The common versions are TLS 1.2 and TLS 1.3 because TLS 1.0 has security flaws and is outdated. It addresses the three security goals: TLS establishes a secure channel between client and server, authentication can be done through asymmetric cryptography (e.g. RSA[3], ECDSA[4] or EdDSA[5]). Asymmetric cryptography is the general term used for public-key cryptography, which entails that all communication partners have two keys: a public key, which is shared with others, and a private key, which is never shared with anyone. Meaning if two entities want to communicate through an insecure channel, one encrypts their message with the receiver's public key and the receiver can only decrypt it with their private key. Symmetric cryptography on the other hand is that all entities use the same private key that is exchanged over a secure channel and never shared with anyone else [55]. Confidentiality is a given after a secure connection is established with TLS. Changes made to the packets after establishment will be detected thus enforcing integrity. A secure connection is ensured through the TLS handshake. Figure 5.1 shows the message flow of a TLS handshake. In the key exchange shared keys are exchanged and it is determined which cryptographic algorithm is used. All communication after this is encrypted. The server parameters determine other relevant information, the authentication of the server and optionally the client is authenticated [81]. For this application, the client should definitely be authenticated.

No easter eggs Easter eggs can be cheat codes, hidden messages or hidden parts of the game written into the code by game developers themselves and intended to be used by players. They are a huge part of gaming culture. But having easter eggs give hackers a foot in the door, giving them the ability to expose parts of the code. As soon as parts of the code known to users, hackers do not have to reverse engineer the game anymore, and they can exploit the game. Further easter eggs can even introduce bugs on their own. Sometimes these are even placed in the code maliciously, called 'logic bombs', meant to cause harm (to the software) when triggered. These are left by developers inside the team possibly seeking vengeance [4]. To ensure the integrity of the game, easter eggs are an easy thing that can be left out or they need to be tested as extensively as the rest of the code but these are not hidden messages from programmers rather fun bits meant to be part of the game.

[3] RSA stands for Rivest–Shamir–Adleman and is a public-key cryptosystem used for encryption and digital signing [82]

[4] ECDSA stands for Elliptic Curve Digital Signature Algorithm and is a digital signing algorithm using elliptic curve encryption, which allows for smaller keys [49]

[5] EdDSA stands for Edwards-Curve Digital Signature Algorithm and uses Schnorr's signature with (possible twisted) Edwards curves [51]

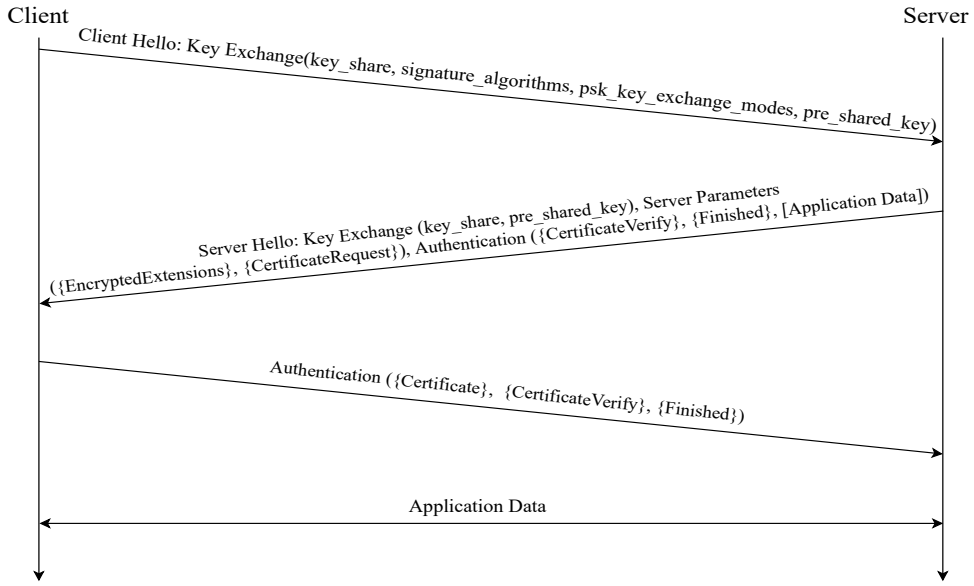


Figure 5.1: *TLS handshake: {} Indicates messages protected using keys derived from a [sender]_handshake_traffic_secret and [ ] Indicates messages protected using keys derived from [sender]_application_traffic_secret_N [81]*

Awareness about existing cheats Knowing how, where, and what cheats get created helps while developing the anti-cheating measures. This does not stop after releasing the game, on the contrary, these efforts then increase. Hackers now have a chance to specifically find the flaws in the game. Meaning, as a developer, it is important to listen to the feedback of users to notice bugs early and to read forums where cheaters and cheat developers exchange information or ready-made cheats. UnKnOwNcHeAtS is a popular example for such a forum. Knowing the common exploits and looking at cheats released for the game can help to fix the loopholes the cheats exploit and to make a blacklist of programs to ban or warn for when scanning a computer like an anti-virus program. Suarez et al. have automated the search through these forums to analyze what types of cheats are being developed and what games they target. This can be used to identify what exploits are used to develop cheats for a game [53]. Cheats are often only written by a small handful of people. Thus, just regularly keeping up with the works of certain cheat developers for the game can be helpful to improve the anti-cheating measures.

Dependency Management As a developer, especially in smaller teams, it is impossible to implement everything yourself, so using already existing software makes sense. But this can introduce vulnerabilities to the game that could be exploited without realizing they are there. Outdated libraries for languages or deprecated functions should not be used. Developers need to make sure that all software they are using is updated to the newest version. If this does not happen and a bad actor knows which version is used, known exploits for this version can easily be found on the internet. Another reason to vet what

software is incorporated into the game is that the creator of that software might put a backdoor into it themselves.

Malicious game objects in Unity[6] are exemplary. Game objects are entities that are part of every object in the game: a ball, grass, the camera, or effects. They do not do anything on their own, different components like code or 2D/3D elements have to be attached to the game object. Through the game objects malicious code can be introduced to the game. One way is that developers can download whole assets in the Unity Asset Store in which the creator of the asset builds a backdoor into the code. Another way is the Unity AssetBundles that allow for game objects to be incorporated at run time, which are non-script but can use the visual scripting systems. AssetBundles can be downloaded at runtime over the internet without transport encryption such as HTTP, which allows for Man In The Middle attacks, where a hacker can manipulate the bundle or even replace it with a malicious one. Even when using HTTPS, Unity does not always check if the certificate is valid: Unity states that some platforms will validate against a root certificate authority store but others will not [100]. This can allow for Man In The Middle attacks [57].

Obfuscation Obfuscation is a common technique in the IT industry to try to hide and protect code from malicious actors. Developers want to hide their source code because through reverse-engineering the game cheaters know how to develop cheats, for example, which variable is the enemy's position and other valuable information. Obfuscation tries to prevent users from understanding the source code by changing its structure without impacting its functionality [6].

This can involve changing variable names to something less identifiable, inserting random code that does nothing, functions that are never called, conditions that are never met or restructuring of code. Obfuscation is independent of hardware and works without additional information such as a cipher key like with code encryption [6].

Code Encryption This is a technique often applied in software engineering to prevent people from reverse-engineering their software. Code encryption tries to establish tamper-resistant code and thus does not give cheat developers a chance to read, understand and change the code. This needs more resources than just obfuscation but also gives more protection. There are many ways to encrypt code. One option is to have two techniques: integrity checking and encryption. Integrity checking is done to ensure no one changed the code and can be done by either hashing or checksum. [13]

A program can either be decrypted at once at launch or functions can be decrypted on demand when they are needed and re-encrypted after use. Different encryption modes include code packing, on-demand decryption to hinder dynamic analysis, bulk decryption, etc. On-demand decryption creates more overhead but if only code that pertains to game

[6]Unity is a popular game engine for developing video games on different platforms

logic is encrypted the overhead is manageable. Developing good encryption takes major effort and time that might not be available to developers, but choosing already existing code encryption software might make it easier for malicious actors to break it. [64]

Game File Verification Verifying game files is a way to see if the user has tampered with the game: for example, the game server sends a challenge to the client that it has to run a hash over the game files and send the hash to the server. The server then compares the result to the hash of the valid version of the game and if it matches, no tampering has occurred. A one-time non-match is not definite proof of tampering because in transit a bit of the hash could flip and thus the hashes would not match. Easy Anti-Cheat for example provides in their anti-cheat SDK the `anticheat_integritytool` which gives the developer the option to check if the hashed client's files match the hashed valid game files [30].

Lockstep Lockstep (LS) is a protocol that tries to eliminate the Lookahead cheat (see Section 3.2.1) in a peer-to-peer architecture. The protocol splits the game into rounds and every player has to submit a hashed decision in that round. Only after submission, the player will be sent the moves of all other players in plain text and they can verify its legitimacy by hashing the plain text. Though it eliminates the possibility that players can react to the actions of other players before the action is taken, it performs badly as every player has to wait for all other players which can take longer due to network latency. [5] Pipelined Lockstep (PLS) is a proposed protocol optimization of Lockstep. With Lockstep, there might be pending plain texts of a player while they want to take another action. PLS solves this by letting the player send the hashed action earlier unless there is a conflict between the two actions [62].

Asynchronous Synchronization Asynchronous Synchronization (AS) is an improvement to the Lockstep protocol where all clients asynchronously from other clients track the gameplay unless interaction is required between clients then AS goes into Lockstep protocol to ensure fair play [5]. Though it is performance-wise an improvement, AS is still too slow for most modern games [104].

Referee Anti-Cheat Scheme (RACS) RACS is a scheme that is a peer-to-peer and client-server architecture hybrid. It tries to solve the problem of cheaters getting information over the network about other players' moves that they are not supposed to know and cheaters that want to run prohibited commands. It consists of three parts: the authentication server, the clients, and the referee. The referee runs on a trusted host and validates the game state. To uphold the scalability clients still communicate with each other. Trusted players are allowed to send updates directly to other clients while cheaters are forced to go through the referee. [20] In this context cheaters are defined as all players where the packets containing their move are delayed to eliminate the Lookahead cheat but this also penalizes players with a bad internet connection. RACS is an improvement to AS and LS but the referee can still pose as a bottleneck [104].

5.2.2 Ring 0/Kernel-Level Anti-Cheat

As explained in Section 4.5 working at kernel-level gives the anti-cheat a lot of privileges. Thus, it has a lot of drawbacks and benefits.

A client-side kernel-based anti-cheat works by looking for known drivers that can be exploited. For example forcing privilege escalation for a program that should not have this access, known drivers that are cheating programs, and system calls that are known to be used in cheat software. Such system calls are for example `WriteProcessMemory`, a Windows function that like the name suggests writes data in a specified area of memory of a process [54]. The anti-cheat then stops the call and examines the arguments it is called with to determine if it is an attempt to cheat or not [64]. It works similarly with `ReadProcessMemory`. Both calls can be used for example for aimbots (see Section 3.2.1) where the cheating program needs to know the position of the enemy to adjust the mouse position to where the enemy is to shoot them perfectly.

Client-side kernel-level anti-cheat drivers additionally often aim to detect whether a game is being played in a virtual environment as it can make it easier to exploit games that way. On the other hand, Linux users often use a virtual machine with Windows as the operating system to play games as Linux is not supported by many games. This because traditionally Linux gives its users full access to the system, while Windows is designed to limit the users' capabilities. Though this seems to be changing and more games now support Linux based operating systems, there is a trade-off between securing the game and supporting the user's experience.

Another functionality that a kernel-level anti-cheat has is access control of the game process. If another process tries to get access to the game process, the anti-cheat notices this while handle-monitoring and if the other process does not have a valid handle, meaning it is not a process approved by the game, it will be stopped from accessing the game process [75]. Cheaters also might want to inject a DLL[7] file into csrss.exe which is the Client/Server Runtime Subsystem that manages starting, ending processes and threads and manages the console [70] to be recognized as a valid process by the anti-cheat while handle-monitoring [17].

Besides privacy related concerns, kernel-level anti-cheat is also more complicated to set up. Developers need to find uncommon ways to detect cheaters because kernel-level cheats try their best to hide in the system. One such technique is utilized by VAC, the software crosschecks with information from the DNS cache if the user connected to cheat servers, thus confirming that the player uses commercial cheat software [2]. Not being able to use common techniques for detecting in kernel-level anti-cheat means that designing such measures is more expansive than other measures.

[7] DLL stands for Dynamic Link Library and these files contain for example data, resources or executable code

5.2.3 Game State Analysis

Knowing what kind of risks are involved with ring 0 anti-cheat developers might rely more upon the analysis of gameplay and other metadata that are created by playing the game to detect cheaters as it is way less invasive and does not make users more vulnerable to attacks. While analyzing game logs the engine tries to look for anomalies that are not totally out of the ordinary like players teleporting or flying when they should not because those should already be detected server-side by cross-checking with the game logic, see Section 5.2.1. Here the detection focuses on the finer anomalies, on behavior that an average player would not exhibit.

One high publicity case of cheating where this analysis was done by the gaming community and not the developers and lead to uncovering a 10-year streak of cheating was the cheating scandal of Trackmania players such as Riolu [91]. Track-records are prestigious in the racing game Trackmania. In these records, it is noted at what timestamp which input was processed. There are four inputs: step on the gas, brake, left and right. Upon completion of a race on a track, the game gives the player a replay file which they can either submit this file or a live recording to Trackmania Exchange to appear in the leaderboard. Two Trackmania players Wirtual and donadigo wrote a script to analyze the inputs that are needed for these specific plays through the replay files. They did find unusual spikes of buttons being pressed in certain inputs: left and right buttons were sometimes pressed so fast after each other that it wasn't humanly achievable. Cheaters were playing Trackmania games at a slower speed to be able to react better to the changes of the track, thus achieving better scores and being on top of the leaderboard [97]. One accused player confessed to playing the game at 40-80% speed. The developer Nandeo stated that they analyzed records of the last fifteen years and most of the ones that were marked for cheating were owned by a small group of people [91].

Different kinds of data that is collected in the game logs can be analyzed. If the game requires client to server or client to client communication of game data, the network traffic could be analyzed. For example one could analyze when packets of clients arrive, Chan et al. noticed that when a bot is playing the game they respond immediately or within a certain timeframe to new information that is sent by the server with a new command, leading to unusual peaks that human players do not have. Additionally, humans tend to react to network conditions subconsciously which bots do not do. [16]
Another way is to analyze the behavior while playing relying on the thought that there is a difference between human and machine behavior. For example the difference between being active, meaning moving around and taking actions, and being idle, meaning standing around and not moving the avatar. This depends on the game, but humans often take an action and then while thinking about their next step are idle. On the other hand, a bot would have less idle time as they do not think but just act on what the constraints tell them to [15].
If a game supports and encourages social play, meaning players play in a team of two or more players, the difference between human and bot teams could be analyzed: human

teams often go for more challenging game aspects whereas bots on the other hand go for easier tasks where they can get more items [52]. Other markers that could be analyzed are the movements in general or before and after a kill, how fast buttons are pressed after another, how good disadvantages, like recoil after firing a gun, are compensated, etc.

5.2.4 Game State Analysis with Machine Learning

Machine learning has become relevant in most parts of technology and can also be applied here: trying to analyze the game state with game logs is extensive computation work and knowing which factors are important to determine what is or is not a cheat is difficult. Machine learning can be utilized here because it is already used to analyze human behaviors and to detect anomalies. It also allows the opportunity for anti-cheating measures that are non-invasive and as such have fewer security risks.

The markers already mentioned in Section 5.2.3 might be easier evaluated with machine learning. For example, Galli et al. analyze the time it takes the player to shoot their weapon as soon as the enemy is visible, the average aiming score while firing at the enemy, and the difference in viewing angles using data from Unreal Tournament[8]. This data is used to determine if a player is using an aimbot even with features like slow aiming that does not immediately focus on the enemy to improve the player's skills. Five different supervised learning methods were used: decision trees[9], Naive Bayes[10], random forests[11], neural networks[12] and support vector machines[13]. Out of the five Naive Bayes and support vector machines were the most accurate, moreover Naive Bayes did not misclassify honest players. [36]

Laurens et al. take a similar approach by analyzing for 'illegal traces'. Illegal traces here mean players that focus on other players, players with strange behavior such as seemingly staring blankly at a wall because they can see through it. Traces that do not last long are seen as random occurrences because a cheating player is more likely to use the cheat continually. They used behavior-based monitoring to do automated cheating detection finding that their system can detect a cheater fast and with less likely false positives. [59]

Jonnalagadda et al. propose a vision-based cheat detection: using Deep Neural Networks (DNN) to detect visual hacks. Moreover, also doing a confidence analysis, i.e. the probability that cheating takes place in that frame and the confidence of that probability to prevent reporting of legitimate players. Additionally, it checks if the DNN need retraining because cheating software changed. Thus, leading to fewer false positives. The system can also account for adversarial attacks. [50]

[8] Unreal Tournament is a first-person shooter game
[9] Decision trees are a predictive modeling approach using tree to model
[10] Naive Bayes does probabilistic classification with the Bayes theorem
[11] Random forests is a classifier that uses uncorrelated decision trees
[12] Neural networks are algorithms modeled after the brain to recognize patterns
[13] Support vector machines are also algorithms for classification using supervised learning models

There are many different approaches to use machine learning for cheat detection and some companies like Valve in VACnet already use deep learning. But this is a measure that is more costly in resources, takes some time to fine-tune and often needs retraining as soon as cheat software changes or new ones are introduced. But it is also one of the least intrusive measures and can be highly accurate as seen above.

5.2.5 Summary

Figure 5.2 gives an overview of all proposed anti-cheating measures presented in this paper and how they fit in with the constraints from Chapter 4. It also indicates whether if the measure is a method to prevent cheaters or detect cheaters. Each measure can get a rating of either ++, +, 0, - or –. A measure with ++ in user experience means it improves it, ++ in budget means that does not need a huge amount of resources, ++ in privacy means that the measure does not disturb the privacy of the user. Anti-cheating measures are here also classified by how effective they are. The effectiveness of some is in a range as it depends on the implementation of the measure. This overview aims to ease directly comparing the measures with each other in order to evaluate which of them fits in best with the demands of the anti-cheat engine.

5.3 Options Beyond the Game

5.3.1 Reports

One of the first approaches to use when getting support from players, is letting users report suspected cheating. Users have the most experience knowing what is achievable through normal means and what is not and thus can detect cheating pretty well. A huge problem this does pose is that some players are inclined to abuse the report button when they feel wronged by other players or if they want to anger players, thus leading to a high number of false positives. This means humans have to check themselves whether the report is valid or not.

Giving the option of submitting video proof of gameplay records can further help the employees who are evaluating whether someone is cheating. Moreover, players that are already suspected of cheating if they are reported are taken more by face value, and those who are not are more probable to be false positives. Into the evaluation also goes the explanation why someone is reported as a cheater.

Reporting is a tool that can be very helpful but also might frustrate players. If a player recognizes a cheater or sees someone harassing other players, reporting them can give the player a sense of justice. But when a player is doing their due diligence reporting cheaters and they feel like in the grand scheme of things it did not make a difference because the mass of cheaters does not seem to deplete, this can frustrate players. An option to go against that is to give players feedback when their report was successful in improving the game. League of Legends has implemented this with their instant feedback: when a player

	Prevention	Detection	General Constraints	User Experience	Budget	Privacy	Effectiveness
Writing good code	X		applicable to everyone	++	++	++	average
Not trusting the client: cross-checking with game logic	X	X	applicable to everyone	+	-	++	average
Not trusting the client: packet protection	X	X	online, multi-player	+	0	++	above average
No easter eggs	X		applicable to everyone	0	++	++	low
Awareness about cheats	X	X	applicable to everyone	++	0	++	average to above average
Dependency Management	X		applicable to everyone	++	+	++	average
Obfuscation	X		applicable to everyone	++	0	++	average to above average
Code Encryption	X		applicable to everyone	+	-	++	above average
Challenges		X	needs client-server architecture	+	0	+	low
Lockstep	X		P2P architecture	- -	0	++	average
Pipelined Lockstep	X		P2P architecture	-	+	++	average
Asynchronous Synchronization	X		P2P architecture	-	+	++	average
RACS	X	X	P2P/client-server hybrid	0	+	++	low to average
Ring 0/Kernel Level	X	X	not applicable to browser and mobile games	0	- -	- -	high
Game State Analysis		X	needs client-server architecture	++	-	++	above average
Game State Analysis with Machine Learning		X	needs client-server architecture	++	- -	++	high

Figure 5.2: *Comparison of technical anti-cheating measures proposed*

gets banned based upon a report of another player, they get the feedback the next time they open the game that the ban took place [61].

5.3.2 Account Verification

Verifying accounts means that people give out information about themselves that is a unique identifier e.g. phone numbers etc. or letting users link their accounts to other services they use such as social media accounts, gaming accounts (e.g. Steam, Epic Games Store), payment details, etc. This makes it easier to identify them if users want to create a new account after a ban and gives more confidence that the user is not a bot.

Another tool that is often used to identify whether a user is a bot or not is CAPTCHA. CAPTCHA is short for Completely Automated Public Turing Test to tell Computers and Humans Apart. This is done by giving the user a challenge that presumably only a human could respond to, thus authenticating the user as a human. There are different types of CAPTCHAs such as text, image, video, audio or puzzle-based. Text-based CAPTCHAs are easy to implement and require the user to identify the characters that are obscured. Image-based CAPTCHAs are the ones often utilized by Google: the user is given a certain amount of images and needs to select all the ones with a certain item in them. [90]
In terms of accessibility, the CAPTCHA methods are controversial for example the visually impaired have problems with the text, image, video and puzzle-based CAPTCHAs. Audio-based ones are problematic for people with hearing loss and text-based ones can be hard to understand for people with dyslexia [89].

5.3.3 Responsible Disclosure without Retribution

Responsible disclosure means allowing hackers to responsibly report bugs and exploits they found to the company and thus giving the company a chance to fix the problem to better protect itself and the people it provides a service to. Allowing white hat, ethical hackers to do this without them having to fear lawsuits or being banned directly positively influences the security of a game, as exploits that might have gone unnoticed and exploited by bad actors now can be fixed. Video game companies like Riot Games, RockstarGames, and Valve all support some form of responsible disclosure with most having a bug bounty program [10]. Bug bounty is when a company monetarily compensates a hacker after disclosing an exploit to them based upon how critical the exploit was to the system. Compensation might also deter people from selling these exploits on the dark web because they still get the money for the time they spent on finding the exploit but do not have to do it illegally.

5.3.4 The Overwatch

The Overwatch is a technique currently known to be used by Counter-Strike: Global Offensive. It lets users who are "selected based on their CS:GO activity (competitive wins, account age, hours played, Skill Group, low report count, etc.) and, if applicable, prior Overwatch participation level and score (a function of their accuracy as an investigator)"

[101, Section: How do investigators get selected?] be investigators. Their job is to watch a replay of eight randomly selected games from the suspected player and review if they committed the reported acts. Investigators can either choose 'insufficient evidence' or 'evidence beyond a reasonable doubt'. Players are suspected of cheating if they suddenly get a lot of reports or incur a lot of reports over a long time. Per case, it is not just one investigator but rather many and the majority of the same verdict applies. This helps to get a more accurate verdict on what is or is not cheating without violating the player's privacy. The suspected cheater's name and all other players' names and voice and text chat are omitted. [101]

Even though it is a great tool, cheaters have found a way to abuse this function. Players often go afk[14] after they notice that they are playing with cheaters because leaving the game would badly influence their ranking, but understandably they do not want to play with cheaters. The cheaters then report them for 'griefing', meaning a player plays badly on purpose to infuriate other players, and thus innocent players get bans. Valve has been reported to have fixed the issue [32].

5.3.5 Restrictions of New Users

If the game can be played competitively, one way to stop cheaters from just making a new account and playing competitively with cheats again is to have players play the game for a certain time and make them complete challenges in the game. This might deter cheaters as it takes a longer time to build up the same rank from the previous game. New players might feel that they are put under general suspicion but that period can also be seen as a tutorial.

[14] Afk or 'away from keyboard' means the player is still in the game virtually but not physically in front of their gaming device

5.3.6 Summary

Figure 5.3 again gives an overview of the proposed anti-cheating measures beyond the game in the context of the constraints.

	Prevention	Detection	General Constraints	User Experience	Budget	Privacy	Effectiveness
Reporting		X	applicable to everyone	+	+	0	low to above average
Verifing accounts	X		applicable to everyone	++	++	0	high
Responsible disclosure without retribution	X		applicable to everyone	++	++	++	high
The Overwatch		X	online, multi-player	++	0	+	above average
Restricting new users	X		competitive games	0	++	++	low

Figure 5.3: *Comparison of anti-cheating measures beyond the game proposed*

5.4 Trustworthiness Factor

The here proposed factor is based upon the "trust factor" from CS:GO. The trust factor was introduced in 2017 and is a hidden rating with different factors influencing it. Most of the factors have not been revealed, but these are confirmed ones:

- how often a player has been reported
- how long players play CS:GO
- how long other video games are played on this Steam account
- how much money was spent on that account
- having a unique phone number linked to the account
- having acquired an account from someone else

The FAQ from Valve about the trust factor also indicates to what is taken into account into the rating. An obvious answer on how to improve a player's trust factor is to not cheat and to play the game as intended, what Valve calls a positive member of the CS:GO and Steam community. This could also mean that leaving harassment in reviews and on players' profiles in Steam might rank players worse. The more a player plays the more information the system has to rank the player, meaning newly made accounts might be ranked lower. [102]

These factors reflect anti-cheating measures proposed in Section 5.3 for example reports and having a phone number linked to the account is a way to verify the account. The length at what the game itself and other games are played can be an indicator of a regular player as players that already invested a lot of time into an account might be deterred

from cheating. Similarly, if a player already spent a significant amount of money on the game they might also be deterred from cheating as all progress and items would be lost if they are banned for cheating. But this factor does also come with controversy as using money as a factor could disadvantage people who cannot spend more money than the price for the game. Having acquired the account from someone else or account sharing can be an indication of Goldfarming (see Section 3.2).

What action can be taken based upon the trust factor, is later extensively described in Chapter 6. But first and foremost it is a tool a developer can use to categorize their players on how much they suspect that they are cheaters. Using the factors from Valve, we introduce other possible factors developers could take into consideration.

Anti-cheating measures detected possible cheating This might seem like an obvious point but if there has already been possible cheating detected that has to be taken into account in the rating. This also should have the most impact on the factor as it is probably the most reliable information about who could be cheating. Depending on their accuracy they should have more or less impact.

Using third-party software that intervenes with the game This is a factor that is often discussed heavily in gaming forums as other software that could be intervening with the game does not have to be a cheat application. Besides a cheat application, there is a possibility that the software does modify the game in a way that does not give an unfair advantage but rather just changes the user interface. Additionally, some software that is used for streaming often also intervenes with the game. But distinguishing between these is almost impossible, thus banning the applications from running is often a point of contention. If it only influences the trust factor it could be a good compromise.

Sudden changes in rank Sudden and big upward changes in the leaderboard or rank of the player could be indicative of a cheater and their recent plays should be evaluated. This depends on how much upward movement is possible for the average user.

Odd play hours Odd play hours mean that a player often plays at the same time on the same days for the same length without interruption and could be an indication that the player lets a bot play on their account as their playtime is automated.

Playing with low trustworthy players Likeminded people often flock together. If one player actively plays together with one or more cheaters that could be a sign of a player that wants to cheat themselves or benefit from the cheating players.

Reporting confirmed cheaters Contrary, if a player reports players for cheating and they are then confirmed to be cheating, can be an indicator of a player that does not intend to cheat is thus rather trustworthy.

6 Possible Consequences for Cheating

Now that we have determined what cheating is and how to detect it, we have to determine how to deal with cheaters. First, we have to acknowledge that no system is perfect, so false positives are expected and the system should not punish non-cheaters. Moreover, banning a player as soon as someone even cheats a little, might destroy the user experience. To ensure this does not happen, we should utilize the trustworthiness factor.

When the trustworthiness factor oversteps certain values, steps will be taken accordingly. In contrast to other video games in the market developers could choose to warn users in question that they are flagged as cheaters. This might give them clues to what made them suspicious but also gives them the opportunity to revert to being a non-cheater. Many steps can be taken before permanently banning someone from a game. In this chapter, possible repercussions are outlined. At which point these steps are taken depends on the moral understanding of developers in what they deem as appropriate.

6.1 Matchmaking

Most online games that are multiplayer have some sort of matchmaking if players do not actively choose opponents. Games such as League of Legends match players according to their skill such that the player has a 50/50 chance of winning [60]. The trustworthiness factor can also be taken into consideration when matchmaking. Players with the same trustworthiness get matched to each other, thus maybe giving everyone the opportunity to be part of a fair game with a mostly level playing field. This means players who want to experience a cheat-free game can do so and the user experience of the game is not tainted. Moreover, if not a lot of cheaters are online at the same time matchmaking can take a longer time thus punishing cheaters further. Furthermore, players with a worse factor could also experience engineered longer waiting times. Thus, players are not actively banned but might not want to wait so long to play. CS:GO matches players based upon their rank and trust factor (see Section 5.4). Players can have prime matching if they achieve a certain rank or a service medal and link their account to a valid phone number. After achieving prime they are only matched with other prime accounts. Another way is to pay for the prime account in the Steam store [86].

6.2 Cheat Servers

One of the reasons people might cheat is just because they want to try it out or want to see if they can beat the game. Either giving people the chance to voluntarily join a cheater server, if they want to cheat or make only the cheater server to them available after they reach a certain trustworthiness factor. The point of a cheater server is that everyone that joins is a cheater which levels the playing field and users do not have the wrong expectations of a game.

Cheat servers are a more extreme version of influencing matchmaking but also gives people the opportunity to cheat without badly affecting others and their experiences in the game. The video game Titanfall makes use of this by not banning cheaters but putting them all on the same server [40].

6.3 Revocation of Accomplishments

Another reason that people might cheat is to brag: they have a special item, they are high up on the leader board, they have a higher rank. Most games have some kind of reward for getting better or an option where the player can compare themselves to other players. This might be an incentive to cheat.

Taking away these accomplishments that were not gained fairly nor deserved, helps to combat this. This way, average users regain trust in the accomplishments. The video game Gears of War 2 for example sets back the score to zero as soon as it notices that the player is cheating [9].

6.4 Restrictions of Other Functionalities

Video games provide, in addition to the core uses, other functionalities such as chats, leaderboards, certain items, seasonal events. Not being able to participate in these functionalities, does not stop cheaters from playing the game but does give them a disadvantage. This option is applied in League of Legends: there is an escalation path of repercussions, the first step is a three-day chat restriction, then a seven-day chat restriction, then a fourteen-day ban, and lastly a permanent ban. Chat restriction in this context means that the offending player can only send a restricted number of messages but can still access pings, emotes, private messages and the voice chat as communicating is an important part of the game [61].

6.5 Public Shame

Another repercussion that could be taken is public shaming. In a video game that could take the form of a banner next to the player's name, changes to their outfit or items that cannot be unequipped. Other players now know that the player is a cheater and thus will not feel as frustrated if they lose the game due to cheating. This measure does also come with a downside: making cheaters visible to other players can lead to harassment of the cheater, which should not be encouraged by the game. In GTA 5 for example cheaters get a hat that says "dunce" that cannot be unequipped and their car blows up as soon as they try to exit it [9].

6.6 Temporary/Permanent Bans

Bans are often the last option in what can be done against cheaters. The duration of the ban is based on how much and how severe the cheating was, it could be a day, a week, months or years. Banning temporarily can be somewhat equated to putting a child in time out. The developer hopes that when the cheater has returned they have learned from their mistakes and do not cheat again. Permanently banning is done when the cheating is so severe that there is no hope for a turnaround or the turnaround should not risk other innocent players' enjoyment. Depending on how it is done and how the game is set up, this might not actually stop cheaters. If the game is free-to-play and players are banned based upon their in-game name or email address, cheaters are able to make a new account or people already have 'Smurf' accounts. 'Smurf' accounts are commonly known as second, third, or even fourth accounts with low ranking to either play badly with friends without it impacting the ranking of the first account or try cheating out in the first place. There have already been measures introduced in this paper that make the thought of just creating a new account rather unappealing or even make creating a new account harder such as restricting new users, having a payment structure, or banning players upon more unique identifiers.

The Valve Anti-Cheating (VAC) system from Valve has taken this a step further banning cheaters not just from the servers in the game they cheated but every game that uses VAC. These cheaters can then only play these games if they have non-secured servers, which means servers without VAC [92]. This might deter people even more from cheating. But also comes with the responsibility that the possibility of a false positive has to be really low, if not game developers run into ruining the user experience.

6.7 E-Sport Tournaments

E-sport's viewership and influence has increased massively over the last years, as seen with the League of Legends Worlds tournament where viewership reached four million [28]. Successful e-sport players can live full-time from participating in tournaments because prize money pools have gotten bigger, they can be directly sponsored by for example companies that make gaming peripherals and they can be supported by fans through merch, Twitch subscriptions, or donations. Social media websites such as Twitch or YouTube also make it possible to watch e-sport players participating in the tournaments or even playing and improving in the game outside of tournaments.

Just as with any other sports event or tournament ensuring the integrity and fairness of the game is essential. With bigger events where players play all from the same stadium, it might be easier to ensure that the computers used are free of cheating software. Most players want to play with their own gear but to ensure that no cheating software is installed, the computers should be scanned. This equates to testing for doping. Some players still manage to cheat for example during the LoL Season 2 World Championships a team looked at screens playing the tournament for the audience to know the positions of the enemy [3].

This also indicates that at e-sport tournaments new ways of cheating are possible which tournament organizers need to be aware of. Looking at smaller tournaments, players often play from the comfort of their own home, meaning they are in a less controlled environment which could make it easier to cheat.

Consequences to cheating might also be harsher and cheating might be tolerated less as there are more people impacted than just the other players and the stakes are higher. A company might choose to ban players not just from the game but from future tournaments, take away titles that were won under false pretenses and publicly call out cheaters on the company's official social media. Most e-sport players are deterred from cheating in tournaments because as soon as their cheating is uncovered their career and their reputation is destroyed. But still, there are instances where this has happened.

6.8 Legal Consequences

Even though most countries in Europe or America do not have laws making cheating illegal, in South Korea and Australia it is different. In South Korea, a cheater was fined $10,000 and another one got two years of probation. Game developers can also sue cheat developers or in some countries like China even report them to the police. Blizzard for example is already making use of this and working together with the cybersecurity department in South Korea. [78]

Cheating in an e-sport tournament can be easier to punish through the legal system. Sponsors have the right to pursue action, but most often do not. Match-fixing is a form of cheating that is already established in other forms of sports: "[i]n organized sports, match fixing is the act of playing or officiating a match with the intention of achieving a pre-determined result [...]" [107]. This has also already occurred in e-sports, one of the first instances took place in South Korea in 2011. When it was uncovered, the consequences for the cheaters were fines and even jail time [24].

As much as there often are not legal consequences for the cheaters themselves, those who develop cheats and sell their solutions on the internet as well ass those who sell items or in-game currency on third-party sites can face them. Companies can try to get the sites that host sellers taken offline or get the site owner to ban this content. For example, eBay only allows the sale of digital goods that the "seller is the legal owner of the electronically delivered goods, as applicable, and such ownership is not from, or the result of, any illegal activity in the United States or any other country" [27, Section: Terms and conditions for eBay approved sellers], meaning in-game currency, items and accounts should not be allowed to be sold but more often than not can be found for a lot of games such as Animal Crossing [88].

Companies can also try to investigate who developed their cheats but this might just exhaust resources as hackers try to stay as anonymous as possible. Moreover, as soon as

one website that hosted these cheats is taken offline another one is made or the website is just moved to the dark web. This means this process also could just waste resources that might be better spent elsewhere. If sellers do move exclusively to the dark web this might be an improvement as the hurdle to get the cheats increases. Additionally, if cheats or items are available on sites like eBay users might think that buying and using them is allowed. For example, Take-Two, RockstarGames' parent company, sued developers that made and sold modifications for Grand Theft Auto 5 which gave players for example unlimited ammunition through Copyright Infringement Laws [83].

7 Discussion and Conclusion

7.1 Discussion

Looking back on this thesis, we see that we established different fields of questions that define the basic framework of the anti-cheat engine before development. Different anti-cheating measures were proposed that all have unique goals and try to solve the problem of cheating from different angles.
Still, there are more measures which are not in this work as we cannot feasibly present all anti-cheating measures that can be used. This thesis focuses on measures that are commonly used and have different goals in mind.
Certain proposed measures additionally have limitations in their capabilities as most systems do. Most client-side anti-cheating measures fall under the problem that because the measures are on the computer of the user they are more susceptible to being analyzed and bypassed by the user. This is because the device is inherently owned by the user and they control it. Thus, leaning more into research on server-side anti-cheat might be seen in the future [64]. For some anti-cheating measures such as ring 0 anti-cheats there is not as much of research available and most companies that utilize these kinds of measures are also very secretive about the inner workings of their software. Hence, it is tough to accurately explain how they work.

Additionally, many anti-cheating measures proposed often detect cheaters based upon the anomalous behavior of the cheater compared to the average user. But if cheat developers also utilize machine learning to their advantage the behavior might become unnoticeable to the anti-cheat engine. There have already been instances where machine learning was used to cheat. At first, the user captured gameplay footage of the game CS:GO and trained the program with it. The cheater then plays the game and the program adjusts the aim accordingly to kill the other player [79]. This makes it certainly very hard to detect as a real person is still playing and the program imitates the behavior of existing good players.

Looking at the *trustworthiness factor*, we proposed different factors that could influence the rating but proving their accuracy in a real world setting is out of the scope of this thesis. Further research could take this into account, implement a demo game with a *trustworthiness factor* and observe players while playing and cheating while evaluating the accuracy of the factors.

Another constraint that was not taken into account primarily that could also influence the usability of the anti-cheating measure is how much the measure affects the game's performance. As user experience is an important factor in designing anti-cheat in this work, game's performance is a relevant part of that. For Lockstep, PLS and AS we evaluated that these anti-cheating measures starkly affected the game's performance. Additionally, measures that try to establish tamper-resistance can affect performance as seen with

Denuvo. Denuvo is a company that mainly is concerned with tamper-resistance for Digital Rights Management (DRM) meaning it works to combat video game piracy, but has recently also gone into the field of anti-cheating. In tests it could be showed that Denuvo lead to worse game performance if applied [44]. Further research could look more closely into the effect of anti-cheating measures on video game performance and assess how much of a trade off has to be done between the game's performance and anti-cheating measures.

7.2 Conclusion and Future Work

Through this thesis, we reviewed different anti-cheating measures and laid the groundwork for developers to design an anti-cheat engine that is not just efficient and accurate but also either does not negatively impact or even improves the user experience. This work gives developers the questions needed to evaluate what constraints the engine has. Questions about the budget, their view on privacy, their ethical understanding of cheating, the structure of the game, and how they envision the user experience to look like help to establish what anti-cheating measures are suitable for the game.

Differing anti-cheating measures that support different game structures were proposed and put in context of the constraints. This work took a look at long-established techniques such as code encryption and obfuscation, solutions for very specific cheats such as Lockstep and RACS and took a very close look at game state analysis optionally with machine learning. Beyond the technical options, some solutions are more meta-gaming such as reporting and the Overwatch which are often overlooked within other papers as there aren't new technological advancements associated with them. Contrary, these solutions can impact the user experience positively. A general marker of how efficient every measure is was given. We additionally present the *trustworthiness factor* as an option to give the player a rating of how much they are suspected of cheating. The factor can then be used to be a baseline for what repercussions cheaters face after being detected. What kind of punishments a cheater can face depends on what the developers feel is appropriate and also might need continuous adjusting based upon constructive feedback of players.

The contributions of this thesis are the questions in Chapter 4 that successfully establish a basic framework focusing on different fields that are relevant to the development of an anti-cheat engine. We evaluated the anti-cheating measures that were proposed with a clear overview to ease the decision which measures should be taken. With the *trustworthiness factor* we took an established tool and contributed our own components that should influence the rating.

Even once the development of the game is finished and it is released, the adjustment and improvement never stops with the anti-cheating measures unless the developers want their game to be overflooded with cheaters. Cheat developers see the improvement of the engine often as a challenge to break the system and make cheating possible again. Thus, game developers need to stay on top of new progress made in the space of anti-cheating.

Future research might want to look more thoroughly into using psychology to the advantage of the game developers, meaning for example deterring players through social pressure from cheating. The reason for that is that no system is perfect and cheat developers will try persistently to break the system. Thus, instead of spending great amounts of resources on detecting cheaters through the newest technology, developers might want to incentivize the player not to cheat through psychology.
Additionally, research recently has focused on using machine learning with anti-cheating measures. With the ever-increasing relevance of machine learning in general research this will continue in the next years as machine learning gives the option of non-invasive and highly accurate anti- cheating measures.

While anti-cheating measures move their functionalities to kernel-level, cheat developers find new and creative ways to stay undetected. One such way are Direct Memory Access (DMA) attacks. DMA attacks use the PCI[1] or PCI-E to access directly the memory. In the use case of video games, cheat developers have been able with the use of PciLeeches[2] and Raspberry Pis to read the exact position of the enemy from memory and then adjust the position of the mouse accordingly not writing it into the memory of the game but sending it as valid mouse input [65]. As every part of the cheat is hardware outside of the computer, the computer never registers the adjustment of the mouse as it only gets the correct position via valid USB mouse commands, thus the cheat becomes almost undetectable. DMA attacks are predicted to get more popular over the next years as it gets harder to stay undetected with software cheats and thus research into how to detect and stop DMA attacks on video games will increase.

Trusted computing is a development that can be utilized in order to put an end to DMA attacks. This technology enables an observer to verify that a program is running in verified state [87]. This verified state could include that the I/O Memory Management Unit (IOMMU) is correctly configured, to disable arbitrary DMA from PCIE-devices. Microsoft pushed this technology through a required TPM2 chip with Windows 11 [69], the introduction of secure-core PCs [71] or the incorporation of the Pluton Chip[3] in new AMD chips [68]. This shows that Microsoft builds up the capabilities to enable trusted computing on the Windows platform which in turn then can be used by anti-cheating developers.

Anti-cheating, its research and development will stay relevant as there is an arms race between cheat developers and game developers. Every time game developers manage to lock a game down cheat developers see it as a challenge to break the game. New hackers often have their first contact with hacking due to games, thus there are a great amount of people with a lot of time on their hands in the game hacking space which continuously leads to new developments.

[1] PCI or PCI-E stands for Peripheral Component Interconnect (Express) and connects peripherals to the chip of the computer
[2] PciLeeches use the PCI or PCI-E to read and write memory [35]
[3] The Pluton chip is the same chip that is installed in the Xbox One

List of Figures

Bibliography

[1] L. Ahearn. *Budgeting and scheduling your game*. Accessed last: 11.1.2022. URL: https://www.gamasutra.com/view/feature/3083/budgeting_and_scheduling_your_game.php?print=1.

[2] ArsTechnica. *Valve DNS privacy flap exposes the murky world of cheat prevention*. Accessed last: 29.1.2022. URL: https://arstechnica.com/gaming/2014/02/valve-dns-privacy-flap-exposes-the-murky-world-of-cheat-prevention/.

[3] R. B. *Top 5 worst cheaters in the history of Esports*. Accessed last: 22.1.2022. Aug. 2020. URL: https://www.sportskeeda.com/esports/top-5-worst-cheaters-history-esports.

[4] N. Bar-Yosef. *Examining the threat of Easter eggs*. Accessed last: 28.12.2021. URL: https://www.securityweek.com/examining-threat-easter-eggs.

[5] N. Baughman and B. Levine. "Cheat-proof playout for centralized and distributed online games". In: *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213)*. 2001. DOI: 10.1109/INFCOM.2001.916692.

[6] C. K. Behera and D. L. Bhaskari. "Different obfuscation techniques for code protection". In: *Procedia Computer Science* (2015).

[7] M. Bieg. *Client-Server-Modell*. Accessed last: 9.1.2022. URL: https://commons.wikimedia.org/w/index.php?curid=2551745.

[8] M. Bieg. *Peer-to-Peer-Modell*. Accessed last: 9.1.2022. URL: https://commons.wikimedia.org/w/index.php?curid=2551723.

[9] J. Bruce. *15 video games that severely punish the player for cheating*. Accessed last: 19.1.2022. Dec. 2017. URL: https://screenrant.com/video-games-that-severely-punish-the-player-for-cheating/.

[10] Bugcrowd. *Bug Bounty Program List - all active programs in 2021*. Accessed last: 15.1.2022. URL: https://www.bugcrowd.com/bug-bounty-list/.

[11] M. J. Callaghan, J. Harkin, E. McColgan, T. M. McGinnity, and L. P. Maguire. "Client–server architecture for collaborative remote experimentation". In: *Journal of Network and Computer Applications* 4 (2007).

[12] Cambridge Dictionary. *Cheating*. Accessed last: 10.1.2022. URL: https://dictionary.cambridge.org/dictionary/english/cheating.

[13] J. Cappaert, B. Preneel, B. Anckaert, M. Madou, and K. De Bosschere. "Towards Tamper Resistant Code Encryption: Practice and Experience". In: Apr. 2008. ISBN: 978-3-540-79103-4. DOI: 10.1007/978-3-540-79104-1_7.

[14] CD PROJEKT. *Security breach update*. Accessed last: 11.1.2022. June 2021. URL: https://www.cdprojekt.com/en/media/news/security-breach-update/.

[15] K.-T. Chen and L.-W. Hong. "User identification based on game-play activity patterns". In: *Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for games*. 2007.

[16] K.-T. Chen, J.-W. Jiang, P. Huang, H.-H. Chu, C. L. Lei, et al. "Identifying MMORPG Bots: A Traffic Analysis Approach". In: *EURASIP Journal on Advances in Signal Processing* (2009).

[17] T. Chen. *Guarding against physical attacks: The Xbox One story*. Accessed last: 29.1.2022. URL: https://www.youtube.com/watch?v=U7VwtOrwceo.

[18] M. Consalvo. "Cheating: Gaining Advantage in Videogames". In: 2007.

[19] M. Csikszentmihalyi. "Flow: The Psychology of Optimal Experience". In: Jan. 1990.

[20] S. Daniel, S. Soh, and W. Lau. "RACS: a referee anti-cheat scheme for P2P gaming". In: (Jan. 2007).

[21] brandon the game dev. *10 Elements of Good Game Design*. Accessed last: 19.12.2021. URL: https://brandonthegamedev.com/10-elements-of-good-game-design/.

[22] J. Dibbell. *The life of the chinese gold farmer*. Accessed last: 29.12.2021. June 2007. URL: https://www.nytimes.com/2007/06/17/magazine/17lootfarmers-t.html.

[23] S. M. Doherty, D. Liskey, C. M. Via, C. Frederick, J. P. Kring, et al. "An analysis of expressed cheating behaviors in video games". In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. 1. SAGE Publications Sage CA: Los Angeles, CA. 2014.

[24] Dot esports. *What is match fixing in esports?* Accessed last: 19.12.2021. URL: https://dotesports.com/general/news/what-is-match-fixing-in-esports.

[25] Easy Anti-Cheat. *Don't bear with the Cheaters*. Accessed last: 1.1.2022. URL: https://www.easy.ac/de-de/.

[26] Easy Anti-Cheat. *Specific errors*. Accessed last: 1.1.2022. URL: https://www.easy.ac/en-us/support/fortnite/issues/errors/.

[27] eBay. *Electronically delivered items policy*. Accessed last: 18.1.2022. URL: https://www.ebay.com/help/policies/prohibited-restricted-items/digitally-delivered-goods-policy?id=4289.

[28] Echarts. *League of Legends (lol) tournaments statistics*. Accessed last: 22.1.2022. URL: https://escharts.com/tournaments/lol/worlds-2021.

[29] Electronic Arts. *Ea statements on recent security incident*. Accessed last: 11.1.2022. URL: https://www.ea.com/news/ea-statement-on-june-11-security-incident.

[30] Epic Online Services Developer. *Using the anti-cheat interfaces*. Accessed last: 19.1.2022. URL: https://dev.epicgames.com/docs/services/en-US/GameServices/AntiCheat/UsingAntiCheat/index.html.

[31] Esports Game Rankings :: Esports Earnings. *Top games awarding prize money*. Accessed last: 11.1.2022. URL: https://www.esportsearnings.com/games.

[32] Esports.com. *CS:GO Hacker missbrauchen Overwatch und Trust Factor*. Accessed last: 19.12.2021. URL: https://www.esports.com/de/csgo-hacker-missbrauchen-overwatch-und-trust-factor-116835.

[33] Federal Trade Commission. *Privacy and security enforcement*. Accessed last: 12.1.2022. Aug. 2019. URL: https://www.ftc.gov/news-events/media-resources/protecting-consumer-privacy/privacy-security-enforcement.

[34] S. Ferretti and M. Roccetti. "AC/DC: an algorithm for cheating detection by cheating". In: Jan. 2006. DOI: 10.1145/1378191.1378220.

[35] U. Frisk. *GitHub - ufrisk/pcileech: Direct memory access (DMA) Attack Software*. Accessed last: 9.2.2022. URL: https://github.com/ufrisk/pcileech.

[36] L. Galli, D. Loiacono, L. Cardamone, and P. L. Lanzi. "A cheating detection framework for Unreal Tournament III: A machine learning approach". In: *2011 IEEE Conference on Computational Intelligence and Games (CIG'11)*. IEEE. 2011.

[37] GameSir. *GameSir VX2 Aimbox console Keyboard & Mouse Adapter (audio)*. Accessed last: 15.1.2022. URL: https://www.gamesir.hk/collections/hot-sale/products/gamesir-vx2-aimbox-console-keyboard-mouse-adapter.

[38] GDPR.eu. *What is GDPR, the EU's new Data Protection Law?* Accessed last: 12.1.2022. Feb. 2019. URL: https://gdpr.eu/what-is-gdpr/.

[39] P. Goodliffe. *Code craft: the practice of writing excellent code*. No Starch Press, 2007.

[40] D. N. Griffiths. *Titanfall's anti-cheat system activates, is beautiful and hilarious*. Accessed last: 19.1.2022. Mar. 2014. URL: https://www.forbes.com/sites/danielnyegriffiths/2014/03/27/titanfalls-anti-cheat-system-activates-is-beautiful-and-hilarious/.

[41] Hertzsprung. *Privilege rings for the x86 available in protected mode*. Accessed last: 6.1.2022. URL: https://commons.wikimedia.org/w/index.php?curid=8950144.

[42] O. Hill. *League of Legends Database hacked. riot games "appreciate your immediate attention"*. Accessed last: 11.1.2022. June 2012. URL: https://www.pcgamer.com/league-of-legends-database-hacked-riot-games-appreciate-your-immediate-attention/.

[43] D. Hoppe. *Level up your gaming privacy knowledge: A walk-through of privacy laws*. Accessed last: 12.1.2022. Dec. 2020. URL: https://gammalaw.com/level-up-your-gaming-privacy-knowledge-a-walk-through-of-privacy-laws/.

[44] J. Hruska. *Denuvo really does cripple PC gaming performance*. Accessed last: 30.1.2022. Dec. 2018. URL: https://www.extremetech.com/gaming/282924-denuvo-really-does-cripple-pc-gaming-performance.

[45] Imperva. *2021 ddos Threat landscape report*. Accessed last: 1.1.2022. 2021. URL: https://www.imperva.com/resources/resource-library/reports/ddos-threat-landscape-report/.

[46] Irdeto. *New global survey: Widespread cheating in multiplayer online games frustrates consumers*. Accessed last: 23.1.2022. Oct. 2018. URL: https://irdeto.com/news/new-global-survey-widespread-cheating-in-multiplayer-online-games-frustrates-consumers/.

[47] T. N. Jagatic, N. A. Johnson, M. Jakobsson, and F. Menczer. "Social phishing". In: *Communications of the ACM* 10 (2007).

[48] Jephthah. *How to hack android games using these (9 working methods)*. Accessed last: 15.1.2022. Jan. 2022. URL: https://itechviral.com/hack-android-games/.

[49] D. Johnson and A. Menezes. *The Elliptic Curve Digital Signature Algorithm (ECDSA)*. Tech. rep. 1999.

[50] A. Jonnalagadda, I. Frosio, S. Schneider, M. McGuire, and J. Kim. "Robust Vision-Based Cheat Detection in Competitive Gaming". In: *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1 (2021).

[51] S. Josefsson and I. Liusvaara. *Edwards-Curve Digital Signature Algorithm (EdDSA)*. RFC 8032. Jan. 2017. DOI: 10.17487/RFC8032. URL: https://rfc-editor.org/rfc/rfc8032.txt.

[52] A. R. Kang, J. Woo, J. Park, and H. K. Kim. "Online game bot detection based on party-play log analysis". In: *Computers & Mathematics with Applications* 9 (2013). Advanced Information Security. ISSN: 0898-1221. DOI: https://doi.org/10.1016/j.camwa.2012.01.034.

[53] P. Karkallis, J. Blasco, G. Suarez-Tangil, and S. Pastrana. "Detecting video-game injectors exchanged in game cheating communities". In: *European Symposium on Research in Computer Security*. Springer. 2021.

[54] Karl-Bridge-Microsoft. *Writeprocessmemory function (memoryapi.h)*. Accessed last: 6.1.2022. URL: https://docs.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-writeprocessmemory.

[55] J. Katz and Y. Lindell. *Introduction to modern cryptography: principles and protocols*. CRC Press, 2007.

[56] J. Ki, J. H. Cheon, J.-U. Kang, and D. Kim. "Taxonomy of online game security". In: *The Electronic Library* (2004).

[57] J. Kielpinski. *Hacking unity games with malicious GameObjects*. Accessed last: 28.12.2021. June 2021. URL: https://blog.includesecurity.com/2021/06/hacking-unity-games-malicious-unity-game-objects/.

[58] K. K. Kimppa and A. K. Bissett. "The ethical significance of cheating in online computer games". In: *The International Review of Information Ethics* (2005).

[59] P. Laurens, R. F. Paige, P. J. Brooke, and H. Chivers. "A novel approach to the detection of cheating in multiplayer online games". In: *12th IEEE International Conference on Engineering Complex Computer Systems (ICECCS 2007)*. IEEE. 2007.

[60] League of Legends Wiki. *Matchmaking*. Accessed last: 18.1.2022. URL: https://leagueoflegends.fandom.com/wiki/Matchmaking.

[61] League of Legends Wiki. *Report feature*. Accessed last: 15.1.2022. URL: https://leagueoflegends.fandom.com/wiki/Report_Feature.

[62] H. Lee, E. Kozlowski, S. Lenker, and S. Jamin. "Multiplayer game cheating prevention with pipelined lockstep protocol". In: *IWEC*. 2002.

[63] I. Lee. *What is security?* Accessed last: 28.12.2021. URL: https://www.cis.upenn.edu/~lee/07cis505/Lec/lec-ch9a-security-v2.pdf.

[64] S. J. Lehtonen et al. "Comparative Study of Anti-cheat Methods in Video Games". In: (2020).

[65] Lohouse. *CS:GO Hardware Aimbot (Project Epo)*. Accessed last: 30.1.2022. URL: https://youtu.be/I7WTed7jwvs.

[66] A. Mallik. "Man-in-the-middle-attack: Understanding in simple words". In: *Cyberspace: Jurnal Pendidikan Teknologi Informasi* 2 (2019).

[67] Manfred. *DEFCON 25- Twenty Years of MMORPG Hacking Better Graphics and Same Exploits*. Accessed last: 29.12.2021. 2019. URL: https://www.youtube.com/watch?v=ZAUf_ygqsDo.

[68] *Meet the microsoft pluton processor – the security chip designed for the future of Windows PCs*. Accessed last: 9.2.2022. URL: https://www.microsoft.com/security/blog/2020/11/17/meet-the-microsoft-pluton-processor-the-security-chip-designed-for-the-future-of-windows-pcs/.

[69] Microsoft. *Enable TPM 2.0 on your PC*. Accessed last: 9.2.2022. URL: https://support.microsoft.com/en-us/windows/enable-tpm-2-0-on-your-pc-1fd5a332-360d-4f46-a1e7-ae6b0c90645c.

[70] Microsoft. *Microsoft previous versions of technical documentation*. Accessed last: 27.1.2022. URL: https://docs.microsoft.com/en-us/previous-versions//cc750820(v=technet.10)?redirectedfrom=MSDN#XSLTsection124121120120.

[71] Microsoft. *Windows 11 secured-core PCs*. Accessed last: 9.2.2022. URL: https://www.microsoft.com/en-us/windows/business/windows-11-secured-core-computers.

[72] Microsoft. *Xbox Adaptive Controller*. Accessed last: 7.2.2022. URL: https://www.microsoft.com/de-de/d/xbox-adaptive-controller/8nsdbhz1n3d8.

[73] National Archives. *ECFR :: 16 CFR part 312 – children's online privacy act*. Accessed last: 12.1.2022. URL: https://www.ecfr.gov/current/title-16/chapter-I/subchapter-C/part-312.

[74] No License and Intel Disclaims. “Intel ® 64 and IA-32 Architectures Software Developer ’ s Manual Volume 3 A : System Programming Guide , Part 1”. In: 2006.

[75] J. Noguera. *Black Hat Europe: Unveiling the Underground World Of Anti-Cheats*. Accessed last: 27.1.2022. URL: https://www.youtube.com/watch?v=yJHyHU5UjTg.

[76] S. O’Dwyer. *North American valorant players Phox and W3ak banned for cheating*. Accessed last: 15.1.2022. June 2020. URL: https://dotesports.com/valorant/news/north-american-valorant-players-phox-and-w3ak-banned-for-cheating.

[77] A. Ometov, S. Bezzateev, N. Mäkitalo, S. Andreev, T. Mikkonen, et al. “Multi-factor authentication: A survey”. In: *Cryptography* 1 (2018).

[78] PCgamer. *Arrested Overwatch hacker in South Korea fined* 10,000. Accessed last: 19.12.2021. URL: https://www.pcgamer.com/arrested-overwatch-hacker-in-south-korea-fined-dollar10000/.

[79] PCMag UK. *Machine learning is now being used to cheat in multiplayer games*. Accessed last: 23.1.2022. July 2021. URL: https://uk.pcmag.com/games/134383/machine-learning-is-now-being-used-to-cheat-in-multiplayer-games.

[80] A. Puleo. *Fall guys has a massive cheater problem*. Accessed last: 11.1.2022. Sept. 2020. URL: https://gamerant.com/fall-guys-massive-cheater-problem/.

[81] E. Rescorla and T. Dierks. “The transport layer security (TLS) protocol version 1.3”. In: (2018).

[82] R. L. Rivest, A. Shamir, and L. Adleman. “A method for obtaining digital signatures and public-key cryptosystems”. In: *Communications of the ACM* 2 (1978).

[83] A. Robertson. *Take-two is suing over a Grand Theft Auto cheating mod - again*. Accessed last: 18.1.2022. Mar. 2019. URL: https://www.theverge.com/2019/3/21/18274336/grand-theft-auto-online-evolve-mod-menu-cheating-copyright-lawsuit.

[84] P. Savage. *ESEA release malware into public client, forcing users to farm bitcoins [updated]*. Accessed last: 5.1.2022. May 2013. URL: https://www.pcgamer.com/esea-accidentally-release-malware-into-public-client-causing-users-to-farm-bitcoins/.

[85] J. Schell. *The Art of Game Design: A Book of Lenses*. 3nd. USA: A. K. Peters, Ltd., 2019.

[86] P. Schmidt. *CS:GO Matchmaking*. Accessed last: 18.1.2022. Feb. 2021. URL: https://www.ingame.de/guides/csgo-matchmaking-90221453.html.

[87] C. Shen, H. Zhang, H. Wang, J. Wang, B. Zhao, et al. “Research on trusted computing and its development”. In: *Science China Information Sciences* 3 (2010).

[88] C. Sheridan. *Animal crossing: New horizons players are selling items on eBay and you shouldn’t do that*. Accessed last: 18.1.2022. Apr. 2020. URL: https://www.gamesradar.com/animal-crossing-new-horizons-players-are-selling-items-on-ebay-and-you-shouldnt-do-that/.

[89] S. Shirali-Shahreza and M. H. Shirali-Shahreza. "Accessibility of CAPTCHA Methods". In: *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*. AISec '11. Chicago, Illinois, USA: Association for Computing Machinery, 2011. ISBN: 9781450310031. DOI: 10.1145/2046684.2046704. URL: https://doi.org/10.1145/2046684.2046704.

[90] V. P. Singh and P. Pal. "Survey of Different Types of CAPTCHA". In: (2014).

[91] R. Stanton. *Trackmania Dev analyses 15 years' worth of records to root-out the cheaters*. Accessed last: 7.1.2022. July 2021. URL: https://www.pcgamer.com/trackmania-dev-analyses-15-years-worth-of-records-to-root-out-the-cheaters/.

[92] Steam. *Valve Anti-Cheat-System (VAC)*. Accessed last: 19.12.2021. URL: https://help.steampowered.com/de/faqs/view/571A-97DA-70E9-FF74#consequences.

[93] Steampowered. *VAC-enabled Steam games*. Accessed last: 7.2.2022. URL: https://store.steampowered.com/search/?category1=998&category2=8.

[94] Steamworks. *Vac Integration (steamworks documentation)*. Accessed last: 1.1.2022. URL: https://partner.steamgames.com/doc/features/anticheat/vac_integration.

[95] Support Valorant. *Uninstalling and Disabling Riot Vanguard*. Accessed last: 5.1.2022. URL: https://support-valorant.riotgames.com/hc/en-us/articles/360044648213-Uninstalling-and-Disabling-Riot-Vanguard.

[96] P. Syverson. "A taxonomy of replay attacks [cryptographic protocols]". In: *Proceedings The Computer Security Foundations Workshop VII*. IEEE. 1994.

[97] TMX replay investigation. *TMX Replay Investigation A report by donadigo & Wirtual*. Accessed last: 7.1.2022. URL: https://donadigo.com/tmx1#4-results.

[98] T. Tuna, E. Akbas, A. Aksoy, M. A. Canbaz, U. Karabiyik, et al. "User characterization for online social networks". In: *Social Network Analysis and Mining* 1 (2016).

[99] Ubisoft. *Fairfight anti-cheat software*. Accessed last: 1.1.2022. Sept. 2021. URL: https://www.i3d.net/products/hosting/anti-cheat-software/.

[100] Unity. *Networking.UnityWebRequest-certificateHandler*. Accessed last: 28.12.2021. URL: https://docs.unity3d.com/ScriptReference/Networking.UnityWebRequest-certificateHandler.html.

[101] Valve. *Overwatch*. Accessed last: 19.12.2021. URL: https://blog.counter-strike.net/index.php/overwatch/.

[102] Valve. *The Trust Factor*. Accessed last: 19.12.2021. URL: https://blog.counter-strike.net/index.php/the-trust-factor/.

[103] *Vii. unfair and deceptive practices—federal trade commission act*. Accessed last: 12.1.2022. Dec. 2018. URL: https://www.fdic.gov/resources/supervision-and-examinations/consumer-compliance-examination-manual/documents/7/vii-1-1.pdf.

[104] S. Webb and S. Soh. "A survey on network game cheats and P2P solutions". In: *Australian Journal of Intelligent Information Processing Systems* 4 (2008).

[105] WePC. *Video game industry statistics, Trends and data in 2022*. Accessed last: 23.1.2022. Jan. 2022. URL: https://www.wepc.com/news/video-game-statistics/.

[106] C. Werian. *Die PS4 wurde geknackt: Das steckt hinter dem poobs4-jailbreak*. Accessed last: 15.1.2022. Dec. 2021. URL: https://www.gamepro.de/artikel/ps4-jailbreak-poobs4,3376411.html.

[107] Wikipedia. *Match fixing*. Accessed last: 19.12.2021. URL: https://en.wikipedia.org/wiki/Match_fixing.

[108] T. Wilde. *The controversy over Riot's vanguard anti-cheat software, explained*. Accessed last: 5.1.2022. May 2020. URL: https://www.pcgamer.com/the-controversy-over-riots-vanguard-anti-cheat-software-explained/.

[109] J. Woo and H. K. Kim. "Survey and Research Direction on Online Game Security". In: *Proceedings of the Workshop at SIGGRAPH Asia*. WASA '12. Singapore, Singapore: Association for Computing Machinery, 2012. ISBN: 9781450318358. DOI: 10.1145/2425296.2425300. URL: https://doi.org/10.1145/2425296.2425300.

[110] A. Ziegler and J. Donlon. *04: On Peeker's Advantage & ranked*. Accessed last: 5.1.2022. May 2020. URL: https://playvalorant.com/en-us/news/game-updates/04-on-peeker-s-advantage-ranked/.