ingenieur
wissenschaften
htw saar

Hochschule für
Technik und Wirtschaft
des Saarlandes
University of
Applied Sciences

STL SYSTEMTECHNIKLABOR

How Anti-Cheat Systems Try to Save Online Video Games

Jan Bohnerth

Technical Report – STL-TR-2021-01 – ISSN 2364-7167

stl.htwsaar.de

Technische Berichte des Systemtechniklabors (STL) der htw saar
Technical Reports of the System Technology Lab (STL) at htw saar
ISSN 2364-7167

Jan Bohnerth: How Anti-Cheat Systems Try to Save Online Video Games
Technical report id: STL-TR-2021-01

First published: April 2021
Last revision: April 2021
Internal review: André Miede

For the most recent version of this report see: https://stl.htwsaar.de/

Title image source: Pexels, https://pixabay.com/de/photos/ass-karten-mit-kapuze-haube-mann-1869825/

htw saar – Hochschule für Technik und Wirtschaft des Saarlandes (University of Applied Sciences)
Fakultät für Ingenieurwissenschaften (School of Engineering)
STL – Systemtechniklabor (System Technology Lab)
Prof. Dr.-Ing. André Miede (andre.miede@htwsaar.de)
Goebenstraße 40
66117 Saarbrücken, Germany
https://stl.htwsaar.de

How Anti-Cheat Systems try to save Online Video Games

Jan Bohnerth
htw saar – Hochschule für Technik und Wirtschaft des Saarlandes
Seminar "Angewandte Informatik"
Wintersemester 2020/21

***Abstract*—With video games and their respective communities on the rise, sadly the usage of cheating software is increasing as well. One of the countermeasures to that are anti-cheat systems like FairFight, Vanguard, Valve Anti-Cheat System (VAC) and many more. Besides looking at how those work, we will also go into existing and even new concepts of anti-cheat methods, that could change how cheating software in general needs to operate.**

I. Introduction

A. Videogames and the history of Cheating

Analyzing the numbers of video game players in the last few years, we can see that video games seem to become increasingly more popular. If we look at the already high number from 2014, which was 1.8 billion according to Statista [1] and compare it to the 2.6 billion in 2020 [2], we can see that this community is growing really fast. But sadly among those 2.6 billion are people, that don't want to play by the normal rules but rather exploit unintended game mechanics or use third-party software to gain an unfair advantage. We call those cheaters or hackers. But this is nothing new. When it comes to the history of cheating in video games, the most popular example is the so-called "Konami Code". This "code" was implemented by one of the developers of the game "Contra" and would give the player extra lives, 30 to be exact. When it comes to the reasoning to why said developer (Kazuhisa Hashimoto) implemented this cheat code, he replied in an interview, that he felt the game was really difficult, and he inserted this code, so it would be easier for him and others to finish the game [3]. Because this cheat code became so popular, more than 100 other Konami games had some sort of Konami Code implemented [4]. And that is not all, even other video game companies implemented their own version of a Konami cheat code. As examples Bioshock Infinite [5] and even Fortnite (for certain events) [6], to name just a few. [1] So, if cheating is part of the video game history and even really popular looking at the Konami Code, why is it bad, and what is done to prevent it? First we need to differentiate video games before we can give a reason. There is a big difference between online multiplayer games and just local single player games and the same goes for cheating. In single player games one can only cheat for themselves. However, if we look at multiplayer games the cheating aspect impacts the other players as well. What makes this gap even bigger is, if we take competitive multiplayer games into consideration. In this case one player cheating destroys the whole competitive integrity of the match, often resulting in frustration of the affected players. Another big difference is, that in online video games and their respective competitive game modes, there isn't really a Konami Code or any similar thing that can be used. Instead, cheaters will use exploits and bugs or external, third party programs which manipulate the inputs of the player or the information given to the player. [3]

[1] This [7] source is a Wikipedia article that has a List of all the games and also examples of the Konami Code outside of gaming.

B. Cheating in online videogames

If we take some competitive based shooters as an example and look at the number of banned cheaters, we can see why this creates a problem. Looking at Valve's Counter Strike: Global Offensive (CS:GO), in December 2019 alone, more than 600.000 accounts were banned due to cheating [8]. Just to put this number in perspective, the peak player count of CS:GO on December 2019 was nearly 770.000 [9]. If all these cheaters were part of the peak player count, 7-8 out of 10 players per match cheated. If we would do the same for the average player count in December (450.000+ [9]) this would result in the fact, that every one cheated that played the game at the time. Of course this most likely wasn't the case but just shows how immense this number is. The next competitive game would be Tom Clancy's Rainbow Six: Siege (R6S). In the year 2019 altogether, the number of bans reached around 75.000 [10]. This seems relatively small compared to the 600.000 of CS:GO but taking the player count into consideration again, which peaks at around 130.000 players, and averages at 60.000-70.000 this number is still really high [11]. Even the developers of R6S called this a "ANTI-CHEAT WAR" in their blog [10].

But competitive shooters aren't the only genre that is affected. Massive Multiplayer Online Role Play Games (MMORPG's), such as World of Warcraft (WoW) are also plagued by cheaters, but in a different way. In WoW this is mostly because of either "exploits" or bots. In the case of exploits errors in the games code or unintended mechanics are used to for example duplicate items, teleport players and much more. While most of these bugs are fixed (patched) really fast, the problem remains in the bots. In MMORPG's the term "grinding" refers to doing certain quests or actions repeatedly

to progress in the game itself. The repetitive aspect makes those tasks not that attractive for most of the player base. This is where the bots come into play. These are used to automate this progress without the player being at the computer. Basically the program does the grinding for the player, by doing the same actions over and over again. This is also referred to as ”botting”. Beyond that some of those programs can also be used to input keystrokes at inhuman speed, which is referred to as ”Scripts” or ”MACROs” (those can also be seen in other video game genre, and even competitive shooters) [12]. This is used to make the work for the player easier or to get an advantage when it comes to Player versus Player (PvP) interactions. This mostly leaves honest players, that do not cheat in video games, really frustrated and can even cause a decline in the game's player base [13].

As a result, we can see that there is a big discrepancy between using intended cheat codes in local video games, in comparison to unintended exploits and third party tools in online games.

The remainder of this paper is structured as follows: First, we will take a look at the differences in cheating software by dividing it into hard and soft cheats. Following up on this, in Section II-B we will talk about why it is important, to trust the server over the client and the concepts of client-side prediction, networking libraries and tamper resistance. After that we will go back to the client itself and see the impact encryption, obfuscation and tamper resistance in general have. We also look at the example of Riot Games' Vanguard to understand what the term kernel anti-cheat driver means and what they do. We end Section II of with the examples of FairFight as a statistics based anti-cheat as well as VAC and CS:GO's Overwatch system and what makes them unique. Lastly we will talk about the future of gaming and anti-cheat methods with the help of machine learning in the case of VACnet, cloud-streaming services like Google Stadia and how lawsuits are used to try to combat cheat software developers. We close the paper with a conclusion about the potential future with the help of cloud based services.

II. Anti-cheat methods

Since there are not that many papers about this topic, we will mainly look at a Master's thesis by Samuli Lehtonen to understand what anti-cheat systems and methods are and how they work.

A. Differences in cheating software and anti-cheat methods

To subdue or decrease the ability of getting an unfair advantage for players in online video games, there are what we call ”anti-cheat systems”. To analyze how they work, we first need to differentiate again, in this case the severity of cheating . We already talked about external, third party programs, bots, exploits and more in Section I-B. We can divide them like Samuli Lehtonen did in their master's thesis [14, 9–11] into ”Soft cheats” and ”Hard cheats”. Soft cheats are what we called unintended game mechanics or exploits, basically the player using knowledge of the game or even game code to gain an unfair advantage since those were not intended by the developers. Hard cheats on the other hand are the category where the anti-cheat systems are most commonly used. In their case, third party tools are used to modify the game data, gather or modify information that should not be accessible and what we already described as Scripts, MACROs and Bots [12], [14]. To combat these cheats there are basically two ways to do so. One is on the game client itself (client-side or host based) or with the help of the game server (server-side).

B. Server-side anti-cheat

One of the attempts to stop cheaters with the help of the server, is the validation of the data that is sent from the client to the server. Figure 1 is an example from the article [15] about server-side verification. It shows a simple code example that should simulate a game. In this case the player can only interact by pressing the up and down key as well as the ESC key. This means there are only 3 possible, valid outcomes that can be sent from the client to the server (up, down and ESC). What we can now do is check on the server-side, if that is the case. If there were a client that sent a package that did not include any of the three outcomes, we could be certain that it is not a valid client. (Considering the complexity of modern online games this is not as easy to implement as it might seem)

```
100: loc ← 0;
101:
102: while true do
103:     key ← readkey();
104:     if key = ESC then
105:         endgame();
106:     else if key = '↑' then
107:         loc ← loc + 1;
108:     else if key = '↓' then
109:         loc ← loc − 1;
110:     end if
111:     sendlocation(loc);
112: end while
```

Figure 1. Example code on game client

However, a valid client does not mean a trustable client. In Figure 2 we have an example from [14, 13–14], the Master's thesis about anti-cheat methods in video games. This Figure 2 shows, that even though the package includes a valid outcome, it is not the correct one. To prevent this from happening, we can reduce the impact the game client has on the outcome. To do so, rather than letting the game client decide the outcome, the decision is made by our server. In this case, all the client will send is the action rather than the result. With the information of the action, the server can then simulate (in this example the battle) and send an ”action result” back to the client. Similar to the validation method before, this will reduce the potentially malicious impact the client can have but at the cost of server performance [14, 15–17].

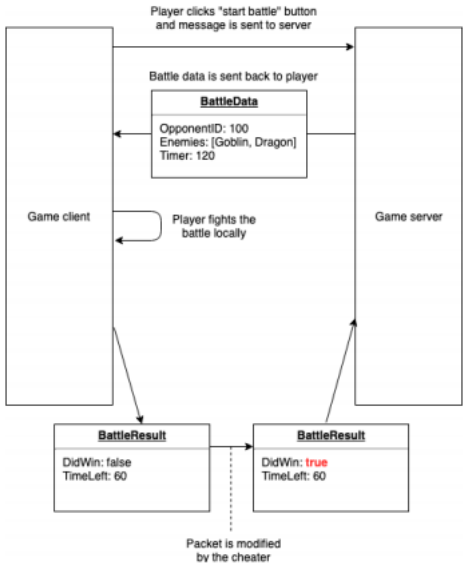

Figure 2. Example for tampering valid result

What could be seen as a result of both of these systems is what we call ”client-side prediction”. The goal of this method is to check if the resulting variables are within certain bounds. This is done by simulating the client-side with an ”authoritative server” and performing the actions of the client on the server-side as well. We do this to prevent things like the tampered battle result from Figure 2. The general idea is, that the server will eventually update the client-side and correct the outcomes which might have been calculated wrong or been tampered. But similar to the other two examples, the drawback is also the server performance and a higher latency since the client needs to wait for the server to answer [16]. To compensate for the delay until the server answers, the client predicts the outcome for the client machine. Here is a concluding Figure 3 from an article from envatotuts on what client-side prediction is [17].

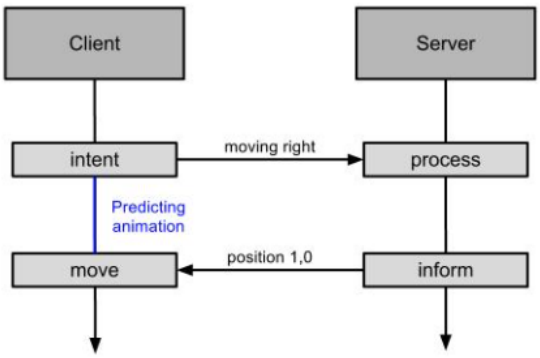

Figure 3. Example for the client-side prediction & lag compensation

Lastly, the protection of the established network and the used application protocol also play a big role in combatting cheaters. For the former, this is mainly done by using a resistant encryption scheme that protects each client-server connection. The idea is, that each connection has for example a unique identifier and a way to authenticate the client. In Lehtonen’s thesis an example for such a networking library, to protect each client-server connection, is RakNet. It basically works like an asymmetrical encryption via a private key (on the server) and a public key (on each game client) [14, 24–27] [18]. For the latter it is important that the designed protocol only sends information that is needed. Coming back to competitive games, this has a big impact. When the protocol sends packages that contain information the player should not have (for example the location of another player behind a wall or out of vision range) the utilization of cheat software could enable a player to get said information. The difficulty in achieving this also lies in the fairness, meaning that the distribution of the packages needs to be at the exact same moment. Otherwise, players might receive a package earlier than others, which would then damage the competitive integrity again [14, 21–27]. Basically the goal is to implement a fair and resistant net code.

To conclude all of this, the best way to mitigate cheating from the server-side is to validate and check the data sent from the client-side. This is done by comparing the results with the ones from the server (client-side prediction). The established connection from each client to the server needs to be protected and encrypted (networking libraries). The overall information sent to the client should be limited to the minimum that is needed (right application protocols) [14, 13–27] [16] [17] [19]. The general rule should be to trust the server rather than the client [20].

C. Client-side/Host based anti-cheat

Compared to the server-side, implementing anti-cheat systems on the client-side can be a lot more difficult. The reason for that is, that client-side anti-cheat systems are on the clients machine itself rather than the server. This means the client has full access to the game code and the anti-cheat. But similar to the server-side anti-cheat system we want to achieve tamper resistance and give a player only the information they need. The former is done by creating a game with as little vulnerabilities as possible while also verifying the legitimacy of the code (this can be done by hashing the files). In addition to that, implementing obfuscation to the code or memory that is used by the application, can make reverse engineering of the code or a statistic analysis a lot more difficult [21]. Since the latter is dependent on the information sent by the server or generated by the client, obfuscation of the heap (memory segment that contains all the important variables of the application) can be a good alternative. To explain how this obfuscation of the memory works, let’s look at an example from Riot Games’ anti-cheat approach for League of Legends. Often times script and bots and their respective developers search for patterns in the games’ memory to detect what useful data is.

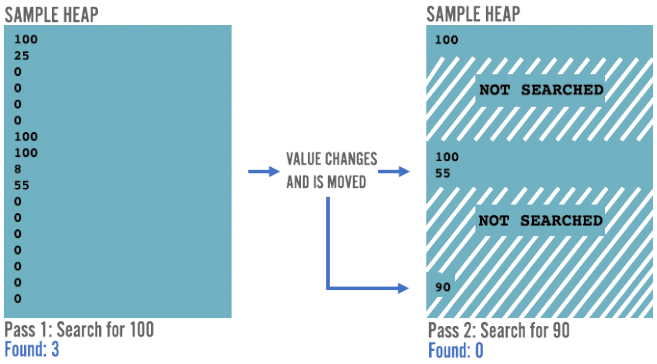

Figure 4. Obfuscated heap example [22]

In their example (Figure 4) useful data would be the memory location of health, mana or something of equal value. If the memory location would not change throughout the progression of a match, tools could then automatically detect what in this case health, mana, etc. would be and read the values directly from the memory. Changing the memory location in addition to changing the values and with that obfuscating the heap, would deny the use of such tools, making it harder to develop scripts or bots for the game. To make the work for cheat software developers even harder, data and code encryption can also be useful in that regard and even increase the effectiveness of obfuscation. The idea is to keep the code sections or game data encrypted until they are needed. That way these tools that try to analyze the code (also statistic analysis), will no longer be able to return a readable version of the application. This again reduces the chances for hackers to find any patterns and also vulnerabilities in the code by a large margin, since they can not obtain interpretable code and neither test it to potentially reverse engineer it [2] [14, 33–38;51].

Since cheat programs themselves become increasingly more intrusive and with that are able to bypass certain anti-cheat systems, there are anti-cheat drivers that operate on kernel level. Figure 5 and 6 show what Kernel (Ring 0) level means. Whereas anti-cheat systems like VAC (Valve Anti-Cheat System) run on the user level or on ring 3 privilege they do not have the same access level/ privileges ring 0 drivers have. This means that user level anti-cheat systems can for example not access hardware directly and only have limited access to memory and CPU instructions [23]. One example for a kernel level anti-cheat system is Riot Games' "Vanguard". The idea of Vanguard is to run in the background even before the main operating system boots up and verify the integrity of the system until the game application is run. By doing so, Vanguard can detect if the system were to be hijacked by cheating software as well as a specific kind of cheat injection on user level [24]. When it comes to the general concept of kernel level anti-cheat drivers, Figure 6 displays exactly that.

[2]The paper of Collberg and Thomborson, on Tamper-Proofing and Obfuscation [21], as well as the Master's thesis of Lehtonen [14, 33–51] go into the details on how this is done

If a cheat program were to write into the memory of the game program or read from it, the kernel driver could intercept and deny any attempts to access it [14, 58–61].

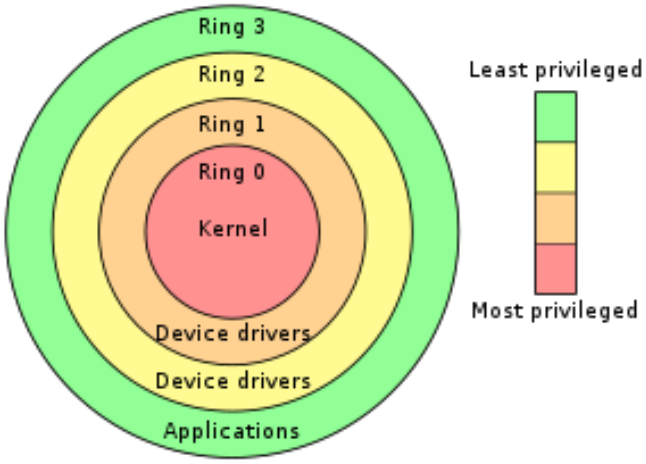

Figure 5. Privilege rings for x86 by Hertzsprung [25]

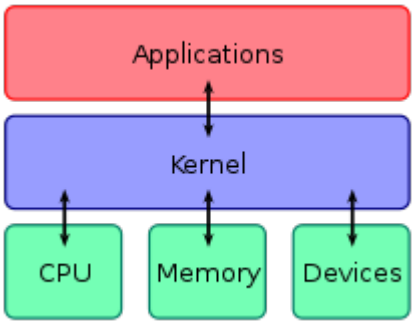

Figure 6. Kernel to connect application and hardware by Bobbo [26]

To summarize the client-side anti-cheat methods, we achieve resistant code, by making it temper-resistant with obfuscation and encryption and with that reduce the ability to find vulnerabilities or patterns in the game data. Lastly, we want to protect the game memory form any unauthorized access, which can be achieved by implementing a kernel level driver [14, 33-61] [21] [25] [26] [23] [24].

There are also methods that utilize client and server. One of those is file-validation with the help of file hashing. Before connecting to a game server the client calculates hashes for the game files. When connecting to the server, the client sends these hashes and the server validates them and either allows a connection or denies it [14, 44–48]. According to an interview by David Kushner Valve's VAC does something similar [27]. The server sends "client challenges" to the client, which it will then solve and send back to the server. What the challenges specifically are is unknown, but it was stated in the interview, that these scan for cheat activities. If the

responses to these challenges are incorrect and VAC finds suspicious activity on the client's machine it will send a incident response to a dedicated team. This team then reviews this response and compares it to a database full of responses (that include cheating or false alarms). If the report were to indentify cheating software on the client-side, it mostly results in the cheater getting banned from the game. However, a VAC-ban will not be immediate, to prevent the cheaters from knowing when exactly VAC and the response team detected the software [27].

D. Other anti-cheat systems

We finished section II-C with looking at VAC. But specifically looking at Valve's CS:GO, this is not the only anti-cheat system in use. Besides them using server-side anti-cheat methods we already discussed, Valve also implemented the so called "Overwatch". This system relies on the community itself. If a player is reported by enough other players (for cheating or similar reasons), the system records the match and stores it as a replayable video. Now experienced players, have the possibility to review such replays in the game client and decide if a player was cheating or not. This overwatch system then decides on the basis of the outcome, if the suspected player gets a penalty (in most cases a ban) or not [28]. But there are other systems aswell. FairFight is an example for an anti-cheat system that relies on statistics [29]. Even though it is considered to be a server-side anti-cheat system, it differs a lot from what we looked at in section II-B. As stated on GameBlocks' webiste: "It does not reside on the player's computer or the game server" [29]. Also, it does not directly interact with either but rather looks at certain, defined gameplay events and how each player completes those. To do that, they use database structures to evaluate a players' gameplay actions and "test them against multiple statistical markers to identify cheating as it occurs". If something is detected that would indicate a client cheating, analysts review the player event(s) and the player is punished depending on the result [29].

III. The future of gaming and anti-cheat methods

A. VACnet

We already talked thoroughly about VAC and CS:GO as one of the examples when it comes to anti-cheat systems. Also, in Section II-D, we briefly looked at the "Overwatch" system from CS:GO. Now over the last years there were also advances when it comes to automatic detection of cheaters, with the use of machine learning. The so called VACnet utilized and still utilizes the vast amount of data that the community helped to gather in the fight against cheaters. With the help of the CS:GO team, VACnet trains itself from monitoring matches and replays and sends potential cheating cases back to the team. That way the system tries to learn how to differentiate between a cheater and a legitimate player [30] [31].

B. Cloudbased gaming

Streaming services became and are still becoming increasingly more popular. But this is not video streaming exclusive. With the likes of Google Stadia even gaming can be done with the help of a cloud-streaming service [32]. In this case, rather than having a client machine that utilizes it's hardware to compute and visualize the game, Stadia puts the workload on their servers. As a result, the servers stream their hosted version of the game to the client and the client solely needs to send back the user inputs [33]. Looking back at Section II-B and especially at Section II-C we can see why this would be a big deal when it comes to combatting cheaters. Google even claims that they want to create more fairness in online video games by implementing systems to prevent cheating software from working [34].

C. Preventing the cheat development

In Section I-B we briefly talked about the frustration cheating can cause, even leading to a decline in the player bases of the affected games. However, what is often times overlooked, is the damages that are caused as a result. Back in 2019 Ubisoft, the developer of R6S, filed a lawsuit against one of the biggest providers of cheating software for said game [35]. This year 2 of the biggest game companies, Riot Games and Bungie, worked together to do the same thing for their games. According to the article by Nicole Carpenter, the damages that were caused by the creators of the cheating software could even reach millions of dollars. But those were not the only lawsuits, other big companies did similar things over the past years to combat cheating overall [36].

IV. Conclusion

With advancements on the side of cheat software as well as anti-cheat systems the ongoing war between those two factions will most likely not stop any time soon. However, the concepts and methods described in Section II, with the likes of client-side prediction, the use of secure networking libraries as well as code-encryption and more intrusive kernel level anti-cheat drivers can make the detection of cheating software a lot easier and even protect from it. Going forward with cloud-streaming services like Google Stadia and machine learning algorithms there might be an alternative that could provide for an environment, where the use of cheating software becomes a lot more difficult or even impossible.

References

[1] C. Gough, "Number of active video gamers worldwide from 2015 to 2023," *Statista*, 2020, visited on 02-01-2021. [Online]. Available: https://www.statista.com/statistics/748044/number-video-gamers-world/

[2] "Number of Gamers Worldwide 2020/2021: Demographics, Statistics, and Predictions," 2020, visited on 02-01-2021. [Online]. Available: https://financesonline.com/number-of-gamers-worldwide/

[3] F. Aboukhadijeh, "Cheating in Video Games," 2011, visited on 02-01-2021. [Online]. Available: https://feross.org/cheating-in-video-games/

[4] "Konami Code For Each Game in The Contra Series," visited on 02-01-2021. [Online]. Available: https://contracentral.com/konami-code/

[5] J. Juba, "How To Unlock 1999 Mode In BioShock Infinite," 2013, visited on 02-01-2021. [Online]. Available: https://www.gameinformer.com/b/news/archive/2013/03/25/how-to-unlock-1999-mode-in-bioshock-infinite.aspx

[6] R. MacLeod, "There's A Konami Code In Fortnite's Black Hole," 2019, visited on 02-01-2021. [Online]. Available: https://kotaku.com/theres-a-konami-code-in-fortnites-black-hole-1839013491

[7] "Konami Code," visited on 02-01-2021. [Online]. Available: https://en.wikipedia.org/wiki/Konami_Code

[8] K. Beck, "Valve bans an astounding number of cheaters after 'CS:GO' goes free," *Mashable*, 2019, visited on 02-01-2021. [Online]. Available: https://mashable.com/article/csgo-record-vac-bans/?europe=true

[9] "Steam Charts Counter-Strike: Global Offensive," visited on 02-01-2021. [Online]. Available: https://steamcharts.com/app/730

[10] "DEV BLOG: RAINBOW SIX SIEGE'S ANTI-CHEAT WAR," 2020, visited on 02-01-2021. [Online]. Available: https://www.ubisoft.com/en-us/game/rainbow-six/siege/news-updates/71mLMFOOVefAO9qlHMLf3O/null

[11] "Steam Charts Tom Clancy's Rainbow Six Siege," visited on 02-01-2021. [Online]. Available: https://steamcharts.com/app/359550

[12] G. McGraw, *Exploiting online games: cheating massively distributed systems*. Addison-Wesley, 2008.

[13] R. Stanton, "Have hackers and cheats ruined The Division on PC?" *The Guardian*, 2016, visited on 19-02-2021. [Online]. Available: https://www.theguardian.com/technology/2016/apr/26/hackers-cheats-ruined-the-division-pc-ubisoft

[14] S. J. Lehtonen *et al.*, "Comparative study of anti-cheat methods in video games," pp. 9–11, 2020.

[15] D. Bethea, R. A. Cochran, and M. K. Reiter, "Server-side verification of client behavior in online games," *ACM Transactions on Information and System Security (TISSEC)*, vol. 14, no. 4, pp. 1–27, 2008.

[16] S. A. Pfeifer, "Real-time multiplayer gaming: Keeping everyone on the same page."

[17] PotHix, "Gamedev Glossary: What is "Client-Side Prediction"?" *envatotuts*, 2013, visited on 12-02-2021. [Online]. Available: https://gamedevelopment.tutsplus.com/articles/gamedev-glossary-what-is-client-side-prediction--gamedev-3849

[18] "RakNet, Secure Connections," visited on 12-02-2021. [Online]. Available: http://www.raknet.net/raknet/manual/secureconnections.html

[19] "UE3, client Server Model," visited on 12-02-2021. [Online]. Available: https://docs.unrealengine.com/udk/Three/ClientServerModel.html

[20] M. Pritchard, "How to Hurt the Hackers: The Scoop on Internet Cheating and How You Can Combat It," *Gamasutra*, 2000, visited on 18-02-2021. [Online]. Available: https://www.gamasutra.com/view/feature/3149/how_to_hurt_the_hackers_the_scoop_.php?print=1

[21] C. S. Collberg and C. Thomborson, "Watermarking, tamper-proofing, and obfuscation-tools for software protection," *IEEE Transactions on software engineering*, vol. 28, no. 8, pp. 735–746, 2002.

[22] M. VanKuipers, "RIOT'S APPROACH TO ANTI-CHEAT," 2018, visited on 20-02-2021. [Online]. Available: https://technology.riotgames.com/news/riots-approach-anti-cheat

[23] S. Khalifa, "Machine learning and anti-cheating in fps games," *Master's thesis, University College London*, 2016.

[24] J. Conditt, "A closer look at Valorant's always-on anti-cheat system," *engadget*, 2020, visited on 18-02-2021. [Online]. Available: https://www.engadget.com/valorant-vanguard-riot-games-security-interview-video-170025435.html?guccounter=1

[25] "Privilege rings for the x86 available in protected mode," by Hertzsprung at English Wikipedia visited on 18-02-2021. [Online]. Available: https://en.wikipedia.org/wiki/Protection_ring#/media/File:Priv_rings.svg

[26] "A kernel connects the application software to the hardware of a computer." by Bobbo visited on 18-02-2021. [Online]. Available: https://en.wikipedia.org/wiki/Kernel_(operating_system)#/media/File:Kernel_Layout.svg

[27] D. Kushner, "Steamed: Valve Software Battles Video-game Cheaters," *IEEE spectrum*, 2010, visited on 18-02-2021. [Online]. Available: https://spectrum.ieee.org/consumer-electronics/gaming/steamed-valve-software-battles-videogame-cheaters

[28] "CS:GO - Overwatch System," visited on 18-02-2021. [Online]. Available: https://support.steampowered.com/kb_article.php?ref=7562-IPJN-1009

[29] "FairFight," visited on 18-02-2021. [Online]. Available: https://www.gameblocks.com/

[30] M. Wilson, "Valve's VACnet anti-cheat system has over 3,000 CPUs powering it," *Kitguru*, 2018, visited on 20-02-2021. [Online]. Available: https://www.kitguru.net/gaming/matthew-wilson/valves-vacnet-anti-cheat-system-has-over-3000-cpus-powering-it/

[31] ——, "Valve to use Machine Learning to detect CS:GO cheaters," *Kitguru*, 2017, visited on 20-02-2021. [Online]. Available: https://www.kitguru.net/gaming/matthew-wilson/valve-to-use-machine-learning-to-detect-csgo-cheaters/

[32] "Stadia," visited on 20-02-2021. [Online]. Available: https://stadia.google.com/

[33] A. Bhatia, "Google Stadia explained: How does it work and where is it available?" *The Indian Express*, 2019, visited on 20-02-2021. [Online]. Available: https://indianexpress.com/article/technology/gaming/google-stadia-all-your-questions-answered-on-the-cloud-gaming-servi/ce-5640119/

[34] C. Wassenar, "Google Stadia Reveals New Anti-Aimbot Technology," *Unikrn*, 2019, visited on 20-02-2021. [Online]. Available: https://news.unikrn.com/article/google-stadia-anti-cheat-aimbot-technology-gdc-n_cw

[35] N. Carpenter, "Ubisoft suing hackers that sold 'hundreds of thousands of dollars' in Rainbow Six Siege cheats," *Polygon*, 2019, visited on 20-02-2021. [Online]. Available: https://www.polygon.com/2019/10/24/20929739/ubisoft-rainbow-six-siege-lawsuit-hackers-mizusoft-cheapboost

[36] ——, "Riot and Bungie teaming up to sue Valorant and Destiny 2 cheat makers," *Polygon*, 2021, visited on 20-02-2021. [Online]. Available: https://www.msn.com/en-us/entertainment/gaming/riot-and-bungie-teaming-up-to-sue-valorant-and-destiny-2-cheat-makers/ar-BB1cECuc