The Battle Against Cheating: How Anticheat Systems Shape Gaming

University of Turku
Department of Computing
Bachelor's Thesis
Computer Science
February 2025
Abir Hossain

UNIVERSITY OF TURKU
Department of Computing

Abir Hossain: The Battle Against Cheating: How Anticheat Systems Shape Gaming

Bachelor's Thesis, 38 p., 2 app. p.
Computer Science
February 2025

The gaming industry has many challenges in ensuring game integrity and player experience, which is especially seen in the online environment. The objective of this thesis is to examine a range of anticheating measures that can be implemented both within and outside of anticheat system. Providing an overview of existing anticheat frameworks, including Valve Anti-Cheat (VAC), Easy Anti-Cheat (EAC), and BattlEye. Focusing on techniques and strategies that is currently used on both the server-side and client-side.
The study first looks at the motivations behind cheating activities. Ranging from improving one's own social status within the community to gaining unfair advantages. Data mining, log analysis, and statistical analysis are used to identify suspicious activities on the server side. The client-side approach includes kernel-level monitoring and file integrity checks to prevent unauthorized modifications. The thesis also highlights the relevance of user authentication methods and CAPTCHA systems, which help protect against bot attacks.
Moreover, community-driven approaches, such as player reporting systems, can significantly contribute to creating a positive gaming environment. The topics also includes legal frameworks that explore how cheating is addressed across different jurisdictions. This thesis brings together key insights and approaches to demonstrate the importance of a multilayered strategy in maintaining fairness in the online gaming. The findings suggest that to combat emerging challenges it is necessary to take preventive actions.

Keywords: Anti-cheat systems, cheating, motivations, video games,client-side anticheat, server-side anticheat

TURUN YLIOPISTO
Tietotekniikan laitos

Abir Hossain: The Battle Against Cheating: How Anticheat Systems Shape Gaming

TkK-tutkielma, 38 s., 2 liites.
Tietotekniikka
Helmikuu 2025

Pelialalla on useita haasteita pelien eheyden ja pelaajakokemuksen varmistamisessa, mikä erityisesti koskee verkkoympäristöä. Tämän tutkielman tarkoituksena on tutkia erilaisia huijauksen vastaisia toimenpiteitä, joita voidaan toteuttaa anticheat järjestelmien avulla sekä niiden ulkopuolisilla menetelmillä. Työssä otetaan myös esille olemassa olevat anticheat järjestelmät, kuten Valve Anti-Cheat (VAC), Easy Anti-Cheat (EAC) ja BattlEye. Näiden lisäksi esitellään tekniikoita ja strategioita, joita voidaan hyödyntää anticheat järjestelmien palvelin- ja asiakaspuolella.
Turkielma aluksi tarkastelee huijaamisen taustalla olevia motiiveja, jotka vaihtelevat oman sosiaalisen aseman parantamisesta epäoikeudenmukaiseen toimintaan. Näiden toimintojen havaitsemiseksi hyödynnetään palvelipuolella data louhintaa, kirjautumis- sekä tilastollista analyyseja. Asiakaspuolen lähestymistapaan sisältyy kernelin seuranta sekä tiedostojen eheystarkistukset, joiden avulla estetään tiedostojen luvattomat muokkaukset. Tutkielmassa myös korostetaan käyttäjien todennusmekanismeja sekä CAPTCHA järjestelmiä, jotka auttavat bottien torjunnassa.
Lisäksi yhteisölähtöiset lähestymistavat, kuten pelaajarapontointijärjestelmät voivat merkittävästi edistää positiivisen peliympäristön luomista. Tutkielmaan sisältyy myös lyhyt katsaus olemassa olevista oikeudellisista toimista eri lainkäyttöalueilla.
Tämä opintonäytetyö kokoaa yhteen keskeisiä käsityksiä ja lähestymistapoja huijaamisen estämiseksi sekä korostaa monikerroksisen strategian toteuttamisen tärkeyden peliympäristössä. Työn löydökset viittaavat ennaltaehkäiseviin toimiin nousevien haasteiden torjumeksi.

Asiasanat: Anti-cheat systems, cheating, motivations, video games,client-side anticheat, server-side anticheat

Contents

List of Figures

List of Tables

1 Introduction

In recent years, the gaming industry has undergone significant growth worldwide. The estimated overall revenue in 2023 was $406.2 billion and is projected to rise up to $666.69 billion by 2029 [1], as shown in figure 1.1.

Figure 1.1: Graph based on the Statista.com data [1]

Incorporating online gaming across different platforms have brought forward many new ideas and reshaped the gaming landscape. Furthermore, the rise of competitive gaming and large Esport tournaments have brought opportunities for players to switch from a typical gaming hobby to a professional career, earning significant amounts of fame and monetary rewards.

This has also raised questions about cheating and the role of anticheat systems. These systems are essential for maintaining fair play among the players and have a substantial impact on the overall gaming experience. Additionally, there is an

ongoing battle between cheat developers and creators of anticheat systems, posing multiple challenges in the realm of cheating detection and its prevention. The topic of this thesis combines multiple elements, such as technology, gaming culture, and psychology. Exploring aspects of user behaviour, game framework design, and technical solutions from anti-cheat systems offers insights into the bridge between human actions and technology. As someone who enjoys gaming, I am deeply interested in how these complicated anticheat systems work, while maintaining an engaging in-game environment for the players.

Research question and its purpose

This thesis is a literature analysis of the anticheat software systems in online gaming, namely Valve Anti-Cheat (VAC), Easy Anti-Cheat (EAC), and BattlEye systems. The objective is to examine each of these software mechanisms. Furthermore, looking at the broader impact on the gaming industry and its community. This thesis presents a technical background, its implementation, and the impact of anticheat systems both from a player and industry perspective. The research questions are:

1. What motivates players to cheat?
2. What types of anticheating systems are available?
3. What key client-side and server-side measures are used to combat cheating?

This thesis is based on articles and research publications from IEEE and ACM, with a few additional sources through Google search. The references were selected based on few criterias: relevance to the searched topic, credibility and publication timeline, with the earliest two from 2005 and rest being more recent. The Importance was given to academic papers by authors that have other publication records, as well

as sources from established companies, news media outlets, government websites and larger online communities.

The paper is divided into six different sections. Chapter 2 discusses the relationship between cheating and gaming dynamics, covering factors such as fairness in the gameplay, motivations to cheat, and different types of cheating methods. Chapter 3 introduces the VAC, EAC, and BattlEye systems and examines their mechanisms and detection approaches. Chapter 4 covers client-side and server-side anticheat strategies. Chapter 5 discusses other anticheating measures that can be used to supervise the gaming environment, and finally chapter 6 concludes the thesis. AI tools, such as microsoft's copilot, was used on each chapter to check and correct grammatical errors. This helped to ensure overall clarity and readability of the text.

2 Factors behind game success

Research in game development plays a crucial role in determining the success of a video game. Identifying, understanding, and balancing factors like player engagement, player satisfaction, and game mechanics deeply influences the overall gaming experience. The study titled "What Makes a Game High-rated? Towards Factors of Video Game Success" [2] aims to identify the key elements that distinguish high-rated games from others. It analyses several variables, including team size, different game genres, gaming platforms, and encountered challenges.

The findings suggest that there isn't exclusively a single factor that is linked with games' success. However, smaller developer teams were often associated with high-rated games. These high-rated games also tended to have fewer issues in areas such as budgeting, possible delays, and feature reductions from the game. Despite this, surprisingly, granting freedom to the developers does not have a significant impact on gaming ratings. The study also emphasised how team dynamics, game design, and technical expertise should be prioritised in shaping the successful future of a game. These insights offer valuable guidelines for gaming managers across the industry, enabling effective decision-making from the project inception to its implementation. [2]

2.1 Fairness aspect in the gameplay

One of the key factors contributing to a positive gaming environment is players' perception of fairness, particularly regarding behaviour moderations. Achieving and maintaining fair moderating decisions is both important and challenging for game developers. Several elements influence how players perceive the fairness of the disciplinary actions they may face. The main component is a concept of procedural justice, which relates to the fairness of processes used to determine and apply appropriate punishments to players. This naturally impacts players' understanding of fairness. Those who view these procedures as straightforward and transparent tend to have a fair perspective. To achieve the balanced outcome, distributed justice is also needed. The used processes should be clear while ensuring fairness in the distribution of penalties and rewards. This balance is crucial for maintaining trust within the system, leading to common agreements. Establishing respectful and truthful communication with moderators and designers can foster a sense of understanding and fairness among the players. Increasing transparency often correlates with improved fairness perceptions. [3]

Another important consideration is how players' perceptions of fairness can impact outcomes, including organisational citizenship behaviours and motivation. Adopting a positive perception of fairness can lead to improvement in areas such as problem-solving strategies and social support utilization. Game developers should prioritise fairness in their behaviour moderation systems, aiming for effective procedural and distributive justice. Also, by integrating player feedback and reviews early on during the design stage, they can enhance transparency and clarity in their decision-making and communication processes. Overall, by actively prioritising fairness, it not only helps but also guarantees the players feel respected, rightly understood, and justly treated, thereby enhancing the gaming experience. [3]

2.2 Motivations for cheating

There are several reasons why players cheat in video games. For a deeper understanding of this concept, we need to a take look at the psychological aspects that encourage players desires to engage in illegal activities. The reasons can be categorised into three different sections. Advantage, recognition, or the overall thrill of breaking the rules. Finding ways to overcome game design and its mechanics is one way of doing it. Other reasons include lacking the needed skills and time or using shortcuts to overcome difficult parts of the game. There are also people who are interested in financial gains by selling virtual assets in the real world. [4, p. 2]

Common reasons behind cheating, as identified by Laato, include players *causing havoc* by resorting towards cheating and disturbing the gaming experience. Through *experimentation*, hidden areas are exploited within the game dynamics, which normally would not be possible. Additionally, cheating can lead to new innovative *creations* and experiences for gameplay, such as creating mods. Some players may resort to *non-conformity*, rejecting the norms and regulations within the gaming community and asserting independence from gaming supervisors or authorities. [4]

Gaming communities on online platforms also play a role in cheating. Spreading these cheats through social networks can indicate the player's desire for acceptance and recognition. Putting pressure on players and using dishonest means to enable them to stand out within the community [5]. A Study done by J. Passmore [6] indicates that players emotional and psychological motivations affect cheating. In the survey, the participants were asked about their experience with using extraneous game advantages (EGA). Using the gaming mechanics in ways that were not intended, such as game mods, cheat codes, and exploits during the gameplay. Out of 200 participants, around 94 percent acknowledged engaging with EGA, 82.4 percent in relevant situations and 84.6 percent just to enjoy the gaming experience. The study concludes that reasoning for cheating includes repairing one's mood, relieving

stress, and shifting players focus to other things. Cheating can, in this case, act as self-regulation that satisfies players psychological needs while enhancing the overall gaming experience. [6]

2.3 Cheating types and methods

To understand the overall impact of cheating in the gaming world, it is important to explore the resources available for players to cheat. While these methods can, in some way or another, enhance the gaming experience, they can also undermine the competitive nature of the game. Here, we will provide an overview of some of the most well-known and widely used cheating methods in gaming the industry, aiming to understand their impact on players. Some examples of these can be seen in the mod menu shown in figure 2.1, which unlocks unfair modifications for players to use.

Figure 2.1: Typical mod menu

Aimbots are mostly used in Fist-Person Shooter (FPS) games. The main purpose of this system is to automate and maintain the opponent's targeting. By having direct access to the game's internal states, the bot can recognise the target as soon as there is a direct sight. Players gain the ability to lock onto opponents instantly with speed and precision, which is normally impossible within human ability. Increasing the player's aim accuracy eliminates the need for manual aiming and developing aiming skills. Also, because the cheat can predict opponents beyond the visual range,

it removes the necessity to master other gameplay skills such as understanding map layouts and having special awareness. Like these systems, there are also so-called Automatic Bots (auto bots), but they are designed to only automate the gameplay. The difference is that aimbots have inputs from both the player and the bot itself. [7]

Wallhack is another frequently used cheating method that enables players to gain advantages in online gaming. The cheat allows the player to see the opponents through walls and obstacles, making it easy to pinpoint and eliminate enemies while also avoiding detection. This is visualised in the following Figure 2.2.

Figure 2.2: Example of Aimbot and wallhack usage

Taking a closer look at 2.2, we can see green and red marks representing the players in the game. The cheating software visualizes four green-marked players and two red-marked players on the left side of the picture, while two red-marked players are visible on the right side. At the top of the picture, there is a mannequin within the game, which is not marked as it is not an actual player. Here, we can see how the aimbot and wallhack function allows the user to see the positions of other players, even through walls. The root cause of the wallhacks lies in how the game client application processes data. To enhance the gameplay, the server sends data about both visible and non-visible opponents to the client. These data states points do not have an impact on normal players, as they will not be rendered on their

current screen. However, cheaters can exploit these data states to their advantage, giving them the ability to perform a wallhack.

There are several issues with blocking the unauthorised information at the server level. For instance, attempting to process additional data for each player can overload the server, which may cause unnatural gameplay and continuous lags. On the client side, the applications render the pipeline without checking if the object is meant to be visible or not. This allows the non-authorised data to remain in the system's and GPU's memory, opening exploitation opportunities through the use of cheat engines, game clients and kernel mode exploits. [8]

2.4 Other possible methods to cheat

Distributed denial of service attack, also known as *DDOS attacks* are network disruption methods that primarily target websites and servers. The attacker focuses on flooding the site using botnets that send excessive HTTP requests to the server. This increases the traffic and affects the server's performance, potentially causing it to go offline. The purpose of flooding the website with extra data is to distract server's antivirus software, therefore enabling the attacker to breach the security protocols and gain unauthorised access to valuable assets. Figure A.1 in the appendix A provides a general idea of how a DDOS attack works. In the context of gaming, DDOS attacks significantly disrupt the gameplay, which can be seen as interference to the players. Common disruptions include lags, disconnections, and inaccessibility to the game servers. Attacks themselves can last several hours or days. [9] [10] DDOS attacks can be categorised into three different types.

Volumetric attack floods the target's network layer with an overwhelming volume of traffic. To the server, this appears as legitimate, but the incoming data exhausts the target's bandwidth and renders the services inaccessible to the users. On the other hand, *protocol attack* typically exploits the vulnerabilities within the OSI layer

3 (network protocol) and 4 (transport layer). This method floods the target with excessive protocol packets, which consumes available server resources and disrupts the service availability. *Resource layer attack* is done in the application layer, aiming to disrupt the communication data. The attacker can disrupt the target service by exploiting web applications. This can be done by SQL injections or HTTP protocol violations. [9] [10]

Compared to the above cheating methods, *glitches and bugs* are a type of cheat that does not directly use any third-party software or application. These are common in video games, and they often affect player experience. On the other hand, players may exploit in-game software errors and mechanics to gain advantages. A glitch refers to a malfunction of the game software, where there occur anomalies within the gaming environment. For example, the player may find a glitch that allows him to bypass the game barrier, gaining an unfair advantage. A loophole is another type of method that players can exploit. They are not software errors like glitches but are connected to unintended game design. In this, a player may find a way to acquire in-game exp (experience points) or resources at a much faster rate than it was originally intended. Common types of bugs in video games include *information, game graphics*, and *action bugs*. [11]

Information bugs typically occur when essential information within the game is not properly conveyed to the player. Manifesting inaccuracies, such as a lack of quest information or incorrect game instructions, can result in low satisfaction and hinder the gameplay. Such bugs also limit the players' in-game process, leading to frustrations and, in some cases, causing them to quit.

Graphical bugs often affect the visual aspects of the game world, rendering the game incorrectly. This can be noticed by small problems such as text prompts or bigger ones like disappearance of entire environments. These bugs can make certain parts of the game unplayable, which has a negative effect.

Players may occasionally encounter *action bugs* that prevent them from performing in-game actions correctly, often due to malfunctions in the gaming mechanics. The functions may not work as they were originally designed, or the game may fail to recognise player inputs. For instance, fire button not working in FPS gameplay. Player may need to take further action to address this issue. [11]

3 Comparative Overview of Anticheat systems

As discussed in the above sections, the exponential growth of online gaming has brought significant challenges to maintaining both fair play and competitive aspects. Herewith, various anticheating systems have arisen in the gaming industry. Each having its own unique approach to preventing cheating. This section will look at three of the most popular anticheat systems in the current market: Valve Anti-Cheat (VAC), Easy Anti-Cheat (EAC) and BattlEye.

3.1 Valve Anti-Cheat

Valve Anti-Cheat, also known as VAC, is a popular anticheat system developed and released by Valve in 2002. It was first introduced in Counterstrike gameplay and is designed to prevent cheating in multiplayer environments on the Steam platform. The program analyses players' systems and looks for possible data manipulations and cheat directives to determine if there have been cheating occurrences to gain advantages over other gamers. The most common cheats include memory injections with manipulation software, wall hacks, aimbots, and other methods. The system tries to detect these cheat patterns. When the properties are detected, the player receives a delayed ban, which can be enforced for several days after the cheating

was detected. [12] [13] Often players may resort to using more advanced cheating measures, but the system will most likely prevent them from accessing the server.

The VAC system uses several different techniques to detect and identify the cheating software. For example, it uses heuristic analysis to determine the presence of cheats in the computer's memory. The software is also frequently updated to refine the data on cheats on players' computers. Also, the user is not normally notified as the software is updated automatically. [14] As mentioned earlier, when the system detects a cheat, it notifies the user but delays the ban. This is used to help preserve the integrity of the cheat detection system. It also gives time for players to dispute any false accusations. [12] [13]

Players who get banned do not receive a permanent ban from the Steam platform. They are most often banned from certain games or from in-game activities. For example, if the VAC systems detects a player cheating in CS:GO the player will be banned from CS:GO servers, but they can still play other Steam games. There could also be restrictions on item trading or even game content uploads. However, it does not prevent offline game playing or launching other Steam games. [12] [15]

As illustrated in figure 3.1 Valve has developed an advanced machine learning system called VACnet. The AI system is used to automatically detect cheaters in CS:GO. It works by analysing gameplay data to differentiate between normal player behaviour and irregularities. According to Valve's John McDonald [16] the system uses a specialised form of Recurrent Neural Networks (RNN) with Gated Recurrent Units (GRU) and rectified linear unit (ReLu) functions to evaluate game actions. The systems also integrate human oversight through the overwatch system, where more experienced players can review reported cases. This helps to ensure accuracy and minimise the risk of unjustly penalising players. [16] [17]

Overall, the VAC system makes an honest effort to maintain the integrity of multiplayer games while also preserving the fair play aspect. It is by no means a

perfect system, but it integrates new cheat detection methods regularly to provide an even ground for players to enjoy multiplayer gaming.

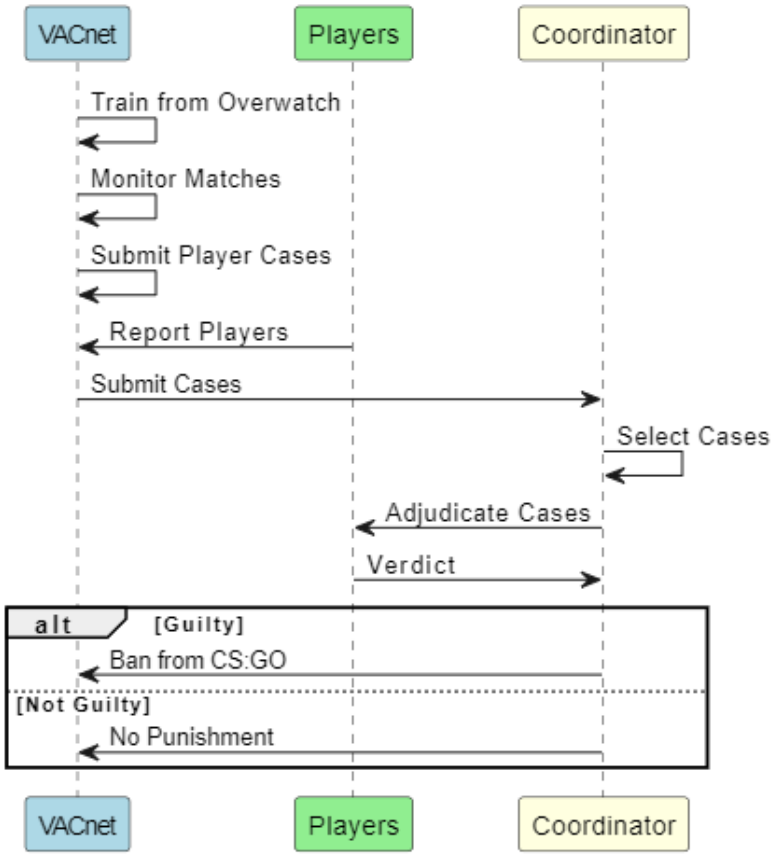


Figure 3.1: Illustration of how VACNET works [16]

3.2 Easy Anti-Cheat

EAC is another popular anticheat middleware program that is widely used in the gaming industry. The most known games include titles such as *Apex Legends*, *Fortnite*, *Dead by Daylight*, and many more. The software was developed by a Finnish company called Kamu, which was later acquired by Epic Games in 2018. [18] As an anticheat system, it is designed to counter hacking and cheating in the multiplayer environment. According to the company's website, the software uses a so-called hybrid anticheat mechanism. They also emphasise their commitment to fostering

a positive gaming environment and valuing each player in the gaming community. Their approach includes rejecting extensive ban lists and avoiding reliance on mass penalisation tactics. They believe that promoting fair play through proactive cheat method tools, which encourages players to shift their mindset in the long term. Having a balance between security and privacy concerns is one example of creating an engaging gaming environment without dividing anyone. [19] [20]

EAC software employs a multilayered approach to ensure fair online gaming environments. The key element in detecting and preventing cheating is kernel-level monitoring within the system. This grants the ability to access and monitor system processes on a deeper level, enabling the software to spot any suspicious activities that indicate cheating. This capacity is enhanced by signature detection methods, which utilise large cheat databases to scan both the memory and processes to find any matches. Even if there aren't any matches, the software uses a heuristic approach to identify potential cheats based on behaviour. Additionally, game files are also checked to confirm the authenticity and ensure that they have not been modified. If a cheater is caught, they can be issued hardware ban that blacklists the hardware IDs. This challenges the cheater to rejoin the gameplay, even if a new account was created. [21]

3.3 BattlEye

BattlEye, founded in 2004 by Bastian Sutrer, is an anticheat software initially created for Battlefield Vietnam game. Since then, it has been implemented into various other game titles, such as *PUBG*, *Destiny 2*, and *Tom Clancy's Rainbow Six Siege*. The system operates based on a two-component architecture, which includes both the server-side and client-side components. Integrating this design allows developers to thoroughly monitor and take actions in cheat detection. According to the company, the program minimally affects system performance, requiring only a small

amount of CPU, RAM and network bandwidth. This ensures a smooth gameplay experience as the anticheat program runs in the background. BattlEye's implementation approach can be divided into three different sections. [22]

BattlEye software employs a *system protection mechanism* to detect and block any cheating and hacking attempts. Continuous monitoring on players' hardware in both user- and kernel mode enables the detection of any suspicious behaviour at the system's core level. Enhancing this with advanced detection algorithms can help identify cheating behaviour patterns, both in existing cheats and new ones. The algorithm employs heuristic and generic approaches, and in some cases, script detection is also used to analyse in-game actions for potential cheats.

Once a cheat is detected, the system *responds* immediately by *enforcing* appropriate actions to eliminate the violations. The game server is completely monitored by BattlEye server. Having full control enhances the ability to continue cross-communications between all the BattlEye clients, making it possible to remove rule-violating players more effectively. The system utilises a global banning system, which can be linked to other ID identifiers like SteamID. Compared to the VAC system, this mechanism makes it possible to take more comprehensive actions.

The company states that all the data transmissions are done through the game's own network system via encrypted packets, ensuring both *security* and *effective communication*. Simplifying the integration process eliminates the need for firewalls or other additional port configurations. Also, an auto-update mechanism with minimal needed network bandwidth maintains the overall system integrity. Furthermore, players are ensured that their privacy is safeguarded, meaning the system collects only the necessary data for cheat detections and does not store any personal information. These qualities, with the ease of integration, make BattlEye a good choice for developers. [22]

3.4 Differences between anticheating systems

Taking a look at Valve Anti-Cheat (VAC), Easy Anti-Cheat (EAC), and BattlEye, each system offers distinct strengths and approaches for ensuring fair gameplay. VAC, developed by Valve, focuses on a combination approach with heuristic analysis and machine learning used in VACnet to detect abnormalities. Additionally, there are delayed bans for players, which are used to preserve detection integrity. Another key element is that VAC bans are game-specific, meaning players can access and play other games on the Steam platform even if they are banned from other titles.

EAC, on the other hand, has a proactive approach. It uses a hybrid system mechanism to detect anomalies within the game environment. The hybrid system implements both kernel-level monitoring and heuristic detection approaches, providing deeper system access to combat cheating. The program may also employ hardware bans, making it more difficult for cheaters to return to the gaming platform.

BattlEye's approach includes a two-component architecture, which allows for both server-side and client-side monitoring. It is known for real-time continuous tracking of in-game activities. Unlike VAC's delayed bans, BattlEye enforces immediate action if anything unusual is detected. This eliminates the rise of cheaters and limits their ability to impact gameplay. The system also utilizes global bans; for example, if the account is linked with a SteamID, it ultimately prevents cheaters from returning to any other games. On the network side, data transmissions are encrypted, which also increases security.

In summary, all three anticheat solutions are similar to each other, but they have their own ways of implementing detection methods and enforcement strategies. VAC relies on a long-term scope for cheat prevention through delayed bans and machine learning. EAC is a mix of both, trying to find a balance between proactive and reactive measures while also maintaining player privacy. BattlEye often takes direct

actions, which include continuous monitoring and immediate bans. As for which one developers should use, it depends entirely on the game's requirements—be it security concerns, the player community, or the type of enforcement developers would like to have.

4 Client and server side anticheat

The anticheating measures can be categorised into client-side and server-side solutions, each having different techniques to combat cheating. Client-side mechanisms focus on player monitoring within the game client, while the server-side evaluates actions done on the game server. Both approaches are crucial for maintaining the a fair gameplay environment. This section discusses some of the implementations of these techniques. There are two main approaches to implementing anticheat solutions in a game system: client-side and server-side anticheat architecture, as represented in figure 4.1.

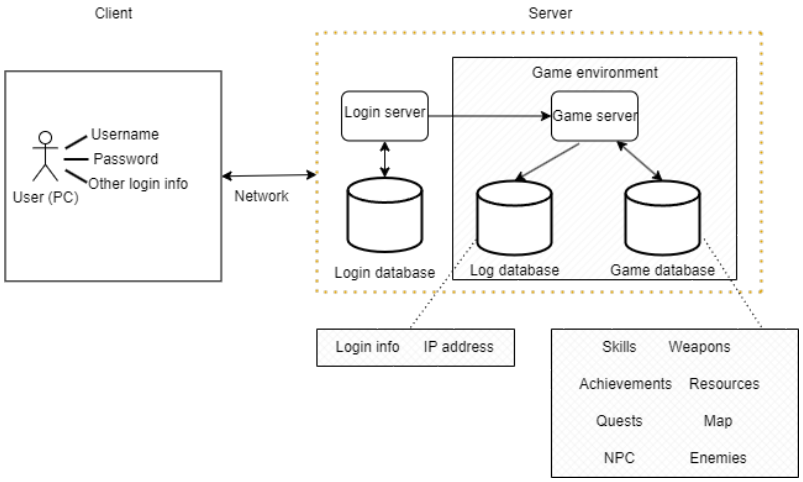


Figure 4.1: An overview of client-side and server-side game server [23]

The figure 4.1 shows the general architecture of a game environment, where the client-side and server-side interactions are represented. The client-side consists of the user's interface, which includes the username, password, and other login creden-

tials. This data is then transferred via the network to the server-side. The server has three main components: the login database, the log database, and the game database. The login server verifies the login information from the login database. After authentication is confirmed, the game server manages the game environment by recording gameplay activities with the help of the log database. The game database contains all the available game elements, such as quests, items, resources and more.

Table 4.1: Client and Server Detection Pros and Cons based on [23]

Method/Detection type	Aspect	Description
Data Mining	Limitation	Limited application in real-time and client-side scenarios due to high computational demands
Statistical Analysis	Limitation	Difficulties with new or rare anomalies because it needs a lot of data to create reliable detection rules
Client-side	Advantage	Fast detection response
	Disadvantage	May conflict with antivirus software
Server-side	Advantage	Accurate fraud detection, logs readily accessible
	Disadvantage	Limited universal application due to game-specific features

Table 4.1 highlights some of the client-side and server-side detection methods, along with their pros and cons. The client-side approach is most often more responsive to take action but may conflict with antivirus software. On the other hand, the server-side approach can be more accurate, though it may be limited by in-game features. The client side often monitors and validates the players behaviours within the client software. This includes, for example, detections of unauthorised modifications within the software or hack detections. In chapter 3 we have discussed a few popular anticheat softwares that are available in the market. These client-side anticheat software are a great way to prevent cheating but can be bypassed by exploiting or modifying the running software on the client's machine.

Similar to the client-sided anticheat method, the server-side also detects and evaluates player behaviours, but this is done directly within the connected game server. This means the server tracks all the communications that are passing through it and verifies player actions. In short, server-side anticheat examines player actions from outside the client software, which makes it difficult for cheaters to bypass.

4.1 Server based anticheat examples

Cheating measures on the server side provide an advantage as they work independently of the player's own machine, eliminating the risk of manipulation through cheats while also providing better protection and resistance. One plus point is that developers get more control over monitoring player behaviour and data integrity, as in-game actions are directly recorded within the server. However, there are a few challenges that the server side faces. The analysis of the collected data can be time- and resource-consuming. Also, detection of minor cheats in the client's environment can pose some challenges. The following section introduces a few server-side techniques and their effectiveness.

4.1.1 Behavior monitoring and statistical analysis

In a study conducted by Bethea, Cochran, and Reite, a server-side anticheat approach is proposed where players' behaviours are verified within the server. The technique uses a symbolic execution approach to create verification conditions. The system then captures in-game behaviours and employs necessary constraints for each server-client interaction. These constraints are then used to determine if the client-server interactions are legitimate user inputs and align with previous server-client communications. [24]

The strong point of this technique is that the process is independent and does not require any client-side software monitoring. This eliminates the need for accessing client-side software code or detecting any made modifications, making more easily scalable across the wide range of different online games. Furthermore, also enhancing scalability across different platforms and programming languages. [24]

The study concludes the robustness and comprehensiveness of the server-side approach and its ability to identify new cheating tactics and behaviours that have not been detected by traditional client-side methods. Demonstrating a practical

applicability and a promising choice for ensuring a fair and secure gaming experience. [24]

4.1.2 Data mining and log analysis

At the server-side, the countermeasures are implemented using the game server's own structural framework. User activities are thoroughly recorded from the time of login to logout. The logs themselves are important for tracking each session. These logs capture essential data such as gameplay, actions, chat exchanges, login times, and transactional activities within the game environment. By having historical record logs, authorised personnel can indicate cheating, such as the use of automated scripts (bots) or real money trading (RMT). [23]

Applying data mining techniques on the server side enables the extraction of crucial insights. By analysing movement trajectories across game maps, performed actions such as monster hunting or quest completion, and interaction patterns with non-playable characters (NPCs), the server can detect deviations. For instance, normal players' movements are often irregular, whereas bots display repetitive actions and moves in a linear manner. Other gaming metrics include playing time, idle time, and resource accumulation patterns. [23]

On the other hand, server-side detection also analyses various aspects of network traffic. This includes the sizes of sent or received packets, the frequency of transmissions, and the timing of sessions between the client program and the game server. By utilising these metrics, it is possible to flag unauthorised transactions or possible bot activities. For example, spikes in packet transmissions or unusual data exchange patterns can indicate the use of automated processes. [23]

4.2 Client based anticheat measures

The difference between server-side and client-side measures is that client-side measures interfere directly with the player's device. This includes more in-depth monitoring of the player's machine, which server-side solutions may overlook. Client-side measures focus on detecting cheats that directly interfere with the gaming environment accessed through the used device. On the other hand, there are several drawbacks, such as privacy concerns and vulnerability to system level cheats. Client-side measures are a great option for keeping the game integrity intact. This section will outline some of the client-side techniques, beginning with kernel level monitoring.

4.2.1 Kernel drive

Before understanding what kernel-level anticheat is, we need to look at the differences between user-mode and kernel-mode. Applications in user-mode operate at a restrictive level within the operating system, preventing them from direct access to processes that are not their own. This limitation makes user-level application defences ineffective against kernel level cheats, such as system calls or memory manipulation, without detection.

On the kernel side, the kernel is the program's core itself and can directly access system resources, including user applications, hardware, and memory. Kernel level anticheat solutions are implemented at the same privilege level as kernel level cheats. By having access to this strategic position, where monitoring and intervening in low system processes are possible, we can prevent game data manipulation attempts at the deeper system level as visualised in figures 4.2 and 4.3. [25]

For instance, in an article about Riot Games' Vanguard anticheat system, which uses kernel level approach to protect game integrity. Enabling the anticheat system to detect any cheating or tampering through direct memory access. The anticheat

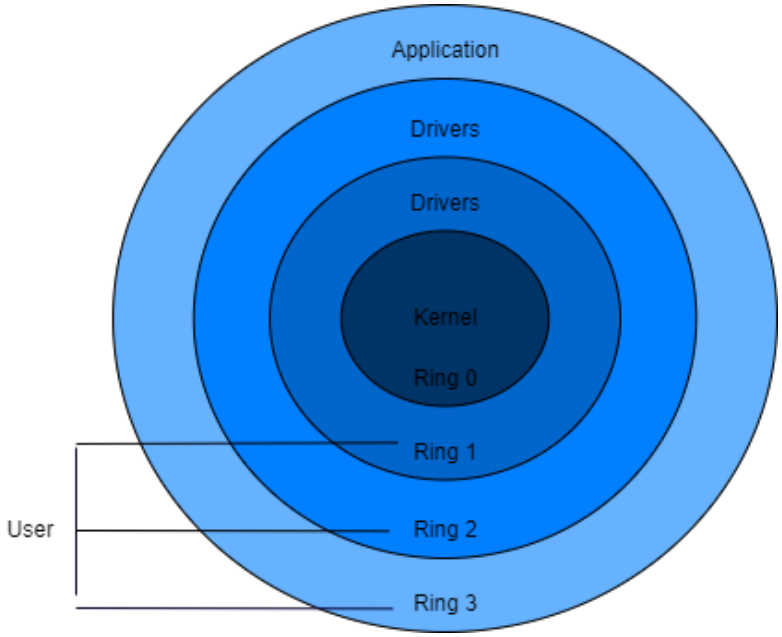


Figure 4.2: Privilege levels organized in rings, with the kernel operating at the highest ring. [26]

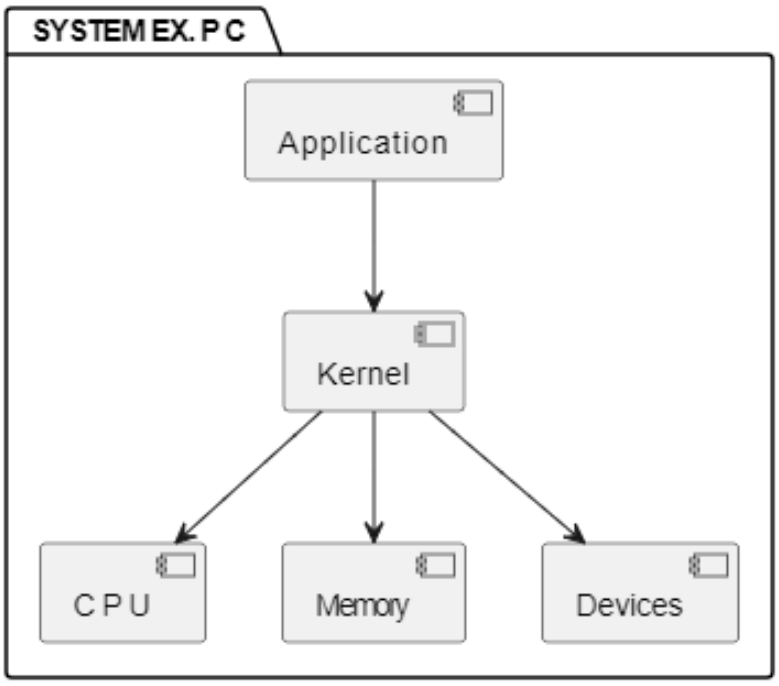


Figure 4.3: Connection between kernel,OS and hardware [27]

system's job is to safeguard a game's integrity, but there are also concerns about privacy, as kernel models have deep access to the system. [26]

4.2.2 File Integrity checking

One of the essential components of the client-side anticheat software is file checking, which ensures the integrity of the game files and prevents any suspicious file modifications. Hash verification and file whitelisting are two types of methods used in file checking.

Hash verification is a commonly used data authentication function that calculates cryptographic hashes, such as MD5 or SHA-1. These hash functions are used to generate unique IDs for each file, which are then used to verify legitimate values. Meaning if even one hash value is different compared to the original, it indicates that the file has been tampered with. This can also be seen as a new hash value is generated if the file contents have been changed. These commonly used MD5 and SHA-1 hash functions are not perfect and are vulnerable to collision attacks. In this type of attack, two types of district inputs produce the same hash output.

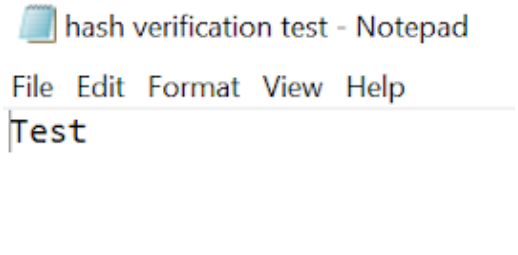
To fight this, Mironov encourages using stronger hash functions, such as SHA-256, and adding an extra layer of security by incorporating HMAC (Hash Based Message Authentication Code), which implements a secret key into the hashing process. This makes it more difficult for the attacker to generate valid hashes. Additionally, combining multiple hash functions can provide further protection and security of the game files. [28] Table 4.2 provides the features of each hash function, and figure 4.4 illustrate how these hash outputs change when original files have been tampered.

Table 4.2: Differences between hash functions [28]

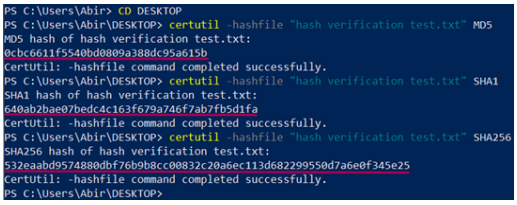
Name	Block size	Output size	Word size	Round
MD5	512	128	32	64
SHA-1	512	160	32	80
SHA-256	512	256	32	64

Table 4.2 compares three common hash funtions: MD5,SHA-1 and SHA-256. All three functions process the data in 512-bit blocks and use 32-bit words. However,

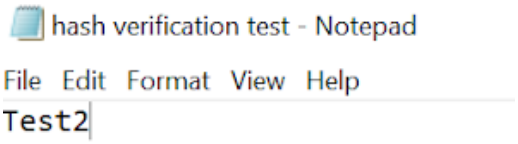
the output size and the number of rounds differ significantly. Due to its larger output size, SHA-256 tends to be more secure than the others. In contrast, MD5 and SHA-1 are considered less secure and may be vulnerable to certain types of attacks. In other words, the output size and the number of rounds contribute to the overall security level of the function, generally larger outputs and more rounds offer stronger protection.



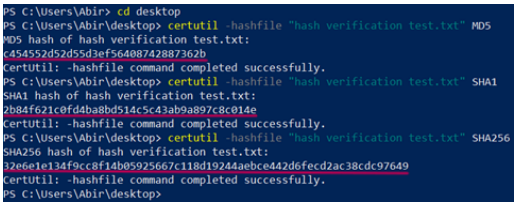
(a) Original text file



(b) Hash verifying the files using different cryptographic hashes



(c) Changed text file



(d) Changing the contents of the text file also changes the hash output

Figure 4.4: Comparison of original and changed text files along with their hash outputs.

The figure 4.4 demonstrates the process of using hash functions to verify the integrity of text files. In subfigure 4.4(a), a text file containing the word "Test" is created. Subfigure 4.4(b) presents various cryptographic hash functions and their corresponding hash outputs, including MD5, SHA1, and SHA256, for the created text file. We can see that each hash function generates its own unique line of data code. The text file is then changed to "Test2", as shown in subfigure 4.4(c). This results in entirely different hash outputs, as illustrated in subfigure 4.4(d), meaning that even a slight change in the text can produce completely different hash values.

The *file whitelisting* method includes creating a list of files that have been approved and verified as safe. Most operating systems have this option, including Windows 11, shown in figure 4.5.

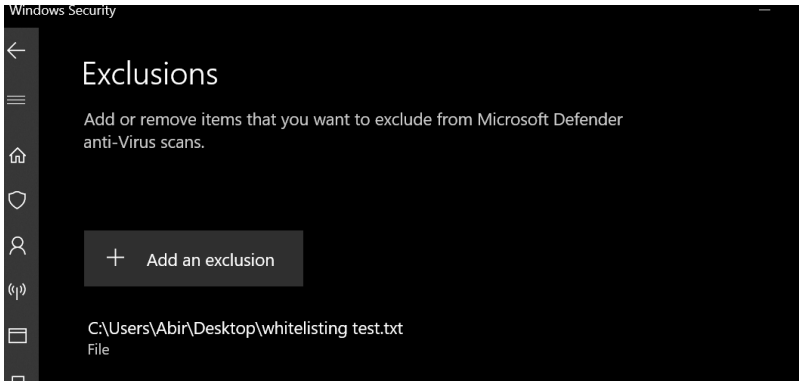


Figure 4.5: Whitelisting files in windows operating system.

The list contains mostly files that are essential for the game's operations. Files that are not on this list could be flagged as suspicious. Depending on the level of risks, these files may be automatically blocked or marked for further inspection, helping to monitor and detect any unauthorised files from compromising the system.

Sudarshan S. Chawathe [29] suggests a more advanced approach that enhances the file whitelisting process. Instead of using the traditional single signature for each file, the method creates a combined signature. It works by first breaking the file into smaller consecutive 512-byte blocks and generating a MD5 signature for each block. The overall combined signature is then generated by taking a small portion of each block's signature. This approach helps in detecting files that are similar even if they have small differences. If a file has small changes, its combined signature will only differ in the parts that have changed, making the whitelisting process more resistant to tampering and more effective. This method's steps is visualized in figure 4.6.

Code injection is a type of security attack where malicious code is inserted into a software application, taking control of the program or accessing data through an

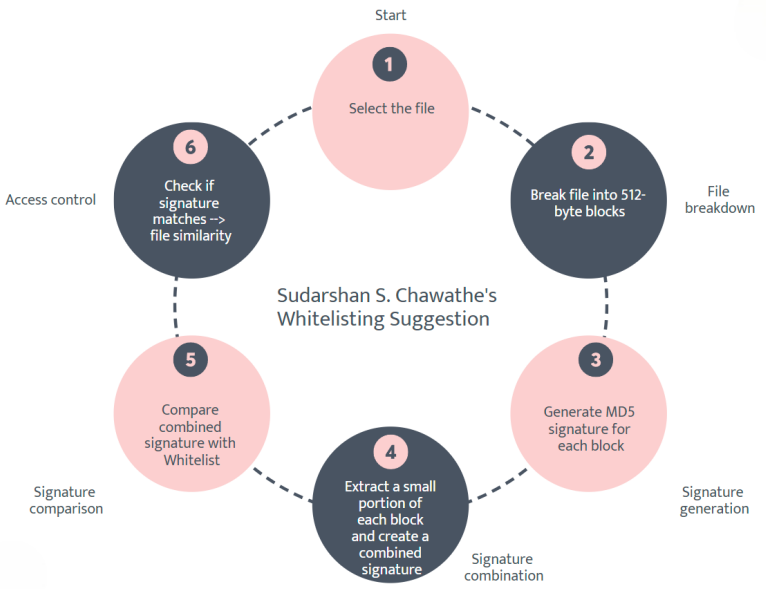


Figure 4.6: Whitelisting suggestion by Chawathe [29]

input or output channel that is not normally available. There are many forms of code injection, such as SQL injection and command injection attacks being examples.

SQL injection works by inserting SQL statements into the user input data of a web application. When these statements are successfully executed, the attacker has the ability to extract, modify, or delete database records, as well as perform unauthorised operations on the database system. Command injection attacks work in a similar way to SQL injections, but there are some differences between the two. SQL injection attacks are specific to database web applications. The attacks are limited to SQL databases with queries that are injected into the application's input field. Command injection attacks are a broader category of code injection attacks and can affect any software application or operating system that accepts user input. The injection can be done with shell or interpreted language commands. Executing code in the underlying system gives attackers the ability to perform actions like file uploads, information downloads, or even control of the system. In short, SQL injection is directed towards the database layer, whereas command injection is directed towards the operating system layer. [30]

Code injection attacks can be prevented in several ways. Filtering the input data to ensure valid characters and sanitising inputs to remove any potentially malicious code from the input fields before execution. Additionally, using taint analysis tools and frameworks can further enhance the prevention of injection attacks.

5 Measures outside of the gaming environment

While in-game anticheating systems directly monitor gameplay itself, the external measures take into account aspects that go beyond gaming. Solving problems that the game's algorithms may not be able to deal with. Incorporating active participation from players, strengthening account security, and creating new law enforcements can ensure a more satisfied gaming experience. The measures that can be put into action include encouraging players to report bad behaviour, introducing multi-factor authentication to protect the accounts, and taking legal actions. These external measures allow developers and authorities much better access and protection in online gaming.

5.1 In-game reporting

While anticheating systems work by utilising algorithms to detect suspicious behaviours, player reporting allows the community to address any inappropriate activities that automated systems might not detect. Implementing intervention mechanisms plays a vital role in creating a positive and toxic-free gaming environment, aiding both developers and gamers in identifying and addressing emerging problems. Hateful and negative speech are examples of toxic behaviours that are common in the gaming space and during the gameplay. The interference system enables players

to anonymously report such behaviours, as presented in figure 5.1, making it easier to raise concerns without fear of discrimination.

Figure 5.1: Example of a player report system. Game: Dead by Daylight

However, most of today's intervention systems act only after toxicity has occurred, potentially limiting their effectiveness. These systems often rely on a clear trigger for intervention. Only a few focus on legal agreements that aim to promote better behaviour before toxicity occurs or on using gender masking as a way to avoid harassment. However, these measures are often reactive. [31]

In-game reporting most often serves as the first step towards online moderation, which is heavily influenced by the social norms. Even though the flagging system gives players the ability to contribute to the reporting system, it by no means is flawless. Yubo Kou suggests that [3] there should be more transparency within the participatory framework, including having clear information on flagging processes and their impact on moderation decisions. Another way to improve the system is by expanding the user role. Recognising and empowering reliable individuals to participate more meaningfully in the flagging process. This could enable new pathways for users to take more significant roles within the community. The system can be further enhanced by integrating feedback mechanisms that address emotional dimensions, potentially reducing stress. Meaning to minimise the false positives, platforms should prioritise aligning practices with the community perceptions. By

refining the understanding of what constitutes toxic behaviour, the classification system can better differentiate between the types of flagged reports. This not only reduces the incorrect flagging but also enhances the overall effectiveness of the moderation system, making it more legitimate and responsive. [32]

5.2 User Authentication

User authentication is an essential protection layer for keeping the online accounts safe beyond the traditional username and password combination. The multifactor authentication (MFA) process involves verifying the identity of the user to a service provider. Since passwords can be guessed or stolen, relying only on them can make user accounts vulnerable to unauthorised access. Many online services, including banking, social media, and gaming platforms, have implemented MFA to enhance security. MFA systems require at least two different factors to verify the user, which helps to reduce the risk of cyber threats and protects valuable assets. [33]

Two-factor verification (2FA) is a security mechanism that requires users to provide two types of identification to gain access to an account. Passwords, tokens, or smart devices such as mobile phones are most often used as the second form of identification. Figure 5.2 at the page 33 represents how a typical 2FA system works.

2FA is a widely used security measure across different systems and platforms for protecting sensitive information. While 2FA provides an additional protective layer compared to single-factor authentication, it also has its limitations. One issue is that it can be vulnerable to attacks, such as brute force or dictionary attacks. The system relies on interactions between the user and the authentication device, which can be inconvenient for users and lead to frustration. There are also discussions about fraud, where criminals can intercept authentication methods by manipulating the authentication processes or implanting malicious software.

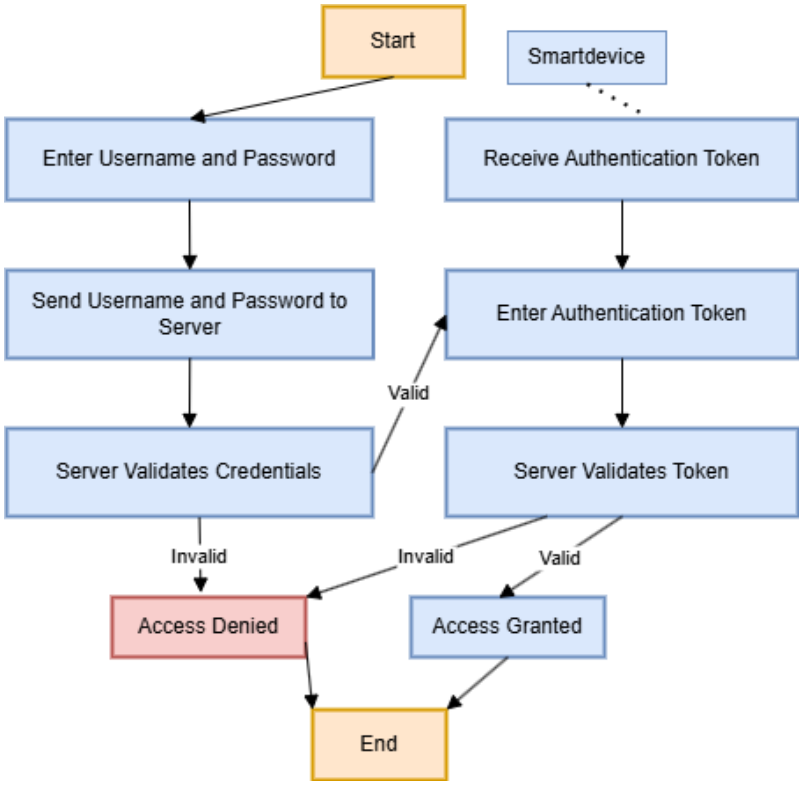


Figure 5.2: Standard Two-Factor Authentication (2FA)

To address this problem, Zhang proposes a transparent two-factor authentication system (T2FA) that solves both the fraud and inconvenience issues in the 2FA. Figure 5.3 provides an overview of this system.

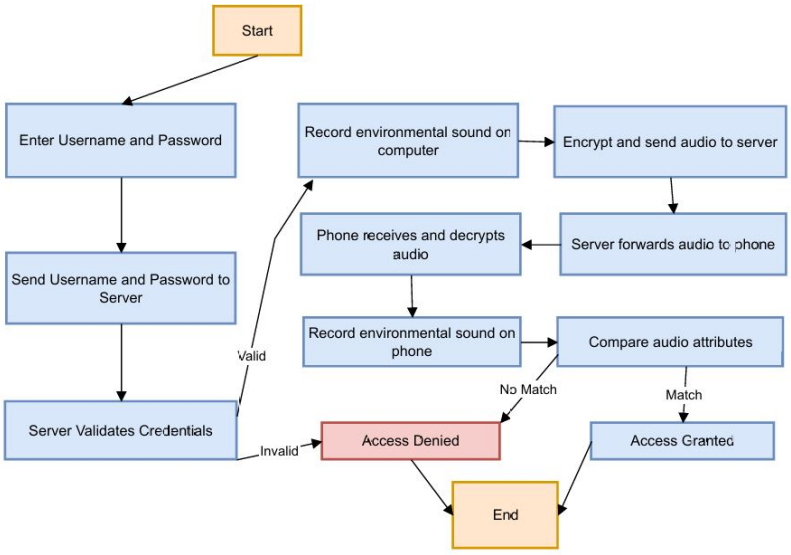


Figure 5.3: Transparent two-factor authentication proposed by Zhang [34]

It uses physical unclonable functions (PUF) to verify the smart devices after the server completes username and password authentication. After this, voiceprints are used to compare the recorded environmental sound with the environmental sounds stored on the smart device. The smart device receives the encrypted audio, which was sent to the server, and compares it with the local audio sample. The comparison is done by extracting attributes, and if there is similarity between them, the smart device notifies the server for legal login. [34]

5.3 CAPTCHA

CAPTCHA (Completely Automated Public Turing Test To Tell Computers and Humans Apart) is a challenge-response security measure that is used to differentiate between human and bot inputs. These tests typically involve puzzles that are simple for humans to solve but are difficult for machines. CAPTCHA tests come in many forms. Typically, the test requires users to solve distorted texts, images, audio prompts, or math problems. The primary purpose of this system is to prevent bots from exploiting online platforms, such as creating fake accounts or automating in-game actions. Thereby preserving the enjoyment and integrity of the digital environment.

One limitation of the CAPTCHAs is their potential to disrupt the user experience. To address this issue, Golle proposes implementing physical CAPTCHAs instead of digital ones. For instance, using tamper-proof devices like joysticks or CAPTCHA tokens. These hardware devices require manual interaction from the user, ensuring that the inputs are provided by humans rather than automated systems. CAPTCHA joysticks can be used to authenticate the communication between the user and the game program. On the other hand, CAPTCHA tokens generate challenges that need to be entered manually to activate the device. Once activated, the token then displays a value that must be submitted to the game server. The

example in figure 5.4 is a prototype for a CAPTCHA token, which resembles a calculator-like device.

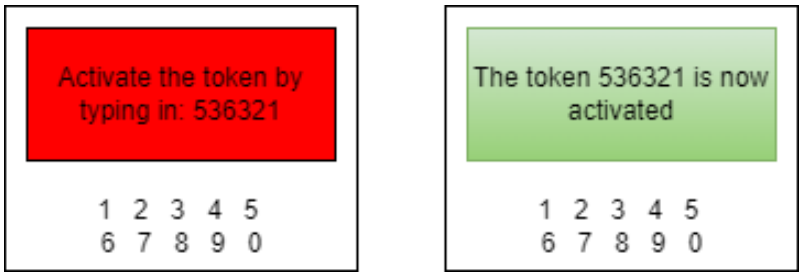


Figure 5.4: CAPTCHA token prototype device [35]

The server sends a token challenge to the the device. The player needs to manually input the token code using the keypad. If inactive, the code is updated over time by generating a new value. This approach reduces the possible bot interferences, as CAPTCHA tokens or joysticks are difficult to outsource or automate. [35]

5.4 Legal actions

The legal actions related to cheating can differ based on the country the player is in and the type of cheating he's done. Some cheats can be considered as copyright infringement due to their ability to alter the files without game developers' authorization. The consequences can vary from typical bans to more serious legal actions such as fines or, in the worst cases, imprisonment. For example, the Digital Millennium Copyright Act (DMCA) in the U.S. allows websites to be removed at the content owner's request. [36] This method is used to identify and take down websites that promote cheats or hacks. Furthermore, this law can be applied to safeguard the intellectual property rights of the game developers, such as banning unauthorised modifications and their distribution. Table 5.1 outlines some general laws that can be implemented to combat cheating in games

Table 5.1: Types of laws that can be implemented [37]

Law Category	Description
Copyright law	Enables game developers to protect their game from unauthorized copying, distribution, or modifications.
Trademark and takedown law	Using trademark law helps developers protect intellectual property rights by stopping unauthorized software and websites from using their trademarks. Takedown laws can be used to remove illegal content or take legal actions against those who violate ownership rights.
Contract law	Developers can apply contract law to prohibit cheating and ensure fair play within the game environment. By inserting anticheating clauses into their end-user license agreements (EULAs) and terms of service (TOS), they can restrict the use of cheats, mods, or hacks.
Criminal law	In some cases, cheating in online games may be regarded as criminal activity, but this is generally considered a last resort and is applied only in certain circumstances.

On the other hand, the Korean government has made game cheating illegal in the country. The law was enforced by the Ministry of Culture, Sports, and Tourism to monitor and crack down on illegal game modifications in collaboration with game companies. Article 32 of the Game Industry Promotion Act outlines that individuals found guilty can face up to 5 years of imprisonment and 50 million won (around 34000 euros) in fines. [38] Like South Korea, China also has a similar system in place. In fact, in 2021, Chinese authorities, in collaboration with Tencent, caught an organisation that sold subscription-based cheating software, making up to $76 million (€67 million) in revenue. In addition to this, the police seized around $46 million worth of assets. [39]

6 Conclusion

The objective of this thesis was to analyse various anticheating measures within the gaming industry, providing an overview of their methodologies and implementation. Through this research, the thesis aimed to answer three questions: 1.What motivates players to cheat? 2.What types of anticheating systems are available? 3.What key client-side and server-side measures are used to combat cheating?

The thesis began by defining the role that anticheat systems play in maintaining the integrity of online gaming environments and preventing any type of unfair advantage. It also delved into motivations behind cheating, addressing the first question: *What motivates players to cheat?* The research identified several key motivations, including the desire for competitive advantage or social status, which often drives players to seek unfair benefits. Psychological factors such as thrill from breaking the rules or exploiting game dynamics significantly encourage players to cheat. The section further discusses a diverse range of cheats, from aimbots to more developed methods like DDoS attacks.

Next, it focuses on examining the techniques used to combat cheating, addressing the second question: *What types of anticheating systems are available?* and the third question: *What key client-side and server-side measures are used to combat cheating?* Systems like Valve Anti-Cheat (VAC), Easy Anti-Cheat (EAC), and BattlEye were analyzed, highlighting how they work to detect and prevent cheating, as well as what differentiates them from each other. After that, client-side and

server-side approaches are discussed. On the client-side, measures such as kernel-level monitoring and file integrity checks offer protection but can be bypassed by advanced cheats. The server-side techniques include behaviour monitoring and data mining, which provide a thorough overview of player activity and network traffic, enabling the identification of irregular patterns that lead to cheating.

Then, this thesis looks at other measures that can be taken outside of the anticheating system framework. In-game reporting is one way to boost community driven moderation, allowing players to flag inappropriate behaviours. The system can be quite reactive, which highlights the need for transparent and proactive measures. From a user security perspective, user authentication methods such as two-factor authentication and CAPTCHA add an extra layer of security, though they also come with their own sets of limitations.

On the legal side, game cheating may have serious consequences for players' lives. To discourage cheating, jurisdictions often enforce types of penalties, such as fines. The legal framework combines both the technological and community approaches, building a comprehensive, multi-layer defence against cheating. The integrations of these strategies demonstrate a thorough approach to anticheat measures, tackling both the technical and social aspects of the problem. While existing systems provide significant protection, new cheating techniques are simultaneously emerging. To fight this, it is crucial for innovation to stay ahead of these techniques and enhance overall effectiveness. However, further research is required, especially in advancing detection techniques and innovating new solutions to maintain fairness and enjoyment in gaming.

References

[1] J. Clement. "Global video game industry revenue 2029". (2024), [Online]. Available: https://www.statista.com/statistics/1344668/revenue-video-game-worldwide/ (visited on 08/27/2024).

[2] G. C. Ullmann, C. Politowski, Y.-G. Guéhéneuc, and F. Petrillo, "What makes a game high-rated?: Towards factors of video game success", in *Proceedings of the 6th International ICSE Workshop on Games and Software Engineering: Engineering Fun, Inspiration, and Motivation*, ser. ICSE '22, ACM, May 2022. DOI: 10.1145/3524494.3527628.

[3] R. Ma, Y. Li, and Y. Kou, "Transparency, fairness, and coping: How players experience moderation in multiplayer online games", in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, ser. CHI '23, ACM, Apr. 2023, pp. 1–21. DOI: 10.1145/3544548.3581097.

[4] S. Laato, S. Rauti, L. Koivunen, and J. Smed, "Technical cheating prevention in location-based games", in *International Conference on Computer Systems and Technologies '21*, ser. CompSysTech '21, ACM, Jun. 2021. DOI: 10.1145/3472410.3472449.

[5] J. Blackburn, N. Kourtellis, J. Skvoretz, M. Ripeanu, and A. Iamnitchi, "Cheating in online games: A social network perspective", *ACM Transactions on Internet Technology*, vol. 13, no. 3, pp. 1–25, May 2014, ISSN: 1557-6051. DOI: 10.1145/2602570.

[6] C. J. Passmore, M. K. Miller, J. Liu, C. J. Phillips, and R. L. Mandryk, "A cheating mood: The emotional and psychological benefits of cheating in single-player games", in *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*, ser. CHI PLAY '20, ACM, Nov. 2020. DOI: 10.1145/3410404.3414252.

[7] S.-Y. Yu, N. Hammerla, J. Yan, and P. Andras, "A statistical aimbot detection method for online fps games", in *The 2012 International Joint Conference on Neural Networks (IJCNN)*, IEEE, Jun. 2012. DOI: 10.1109/ijcnn.2012.6252489.

[8] S. Park, A. Ahmad, and B. Lee, "Blackmirror: Preventing wallhacks in 3d online fps games", in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '20, ACM, Oct. 2020. DOI: 10.1145/3372297.3417890.

[9] Microsoft. "Ddos attacks in gaming". (2023), [Online]. Available: https://www.microsoft.com/en-us/microsoft-365-life-hacks/privacy-and-safety/ddos-attacks-in-gaming (visited on 08/27/2024).

[10] Microsoft. "What is a ddos attack?" (2024), [Online]. Available: https://www.microsoft.com/en-us/security/business/security-101/what-is-a-ddos-attack (visited on 08/27/2024).

[11] A. Truelove, E. Santana de Almeida, and I. Ahmed, "We'll fix it in post: What do bug fixes in video game update notes tell us?", in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, IEEE, May 2021. DOI: 10.1109/icse43902.2021.00073.

[12] Valve Corporation. "Steam support : Valve anti-cheat (vac) system". (2024), [Online]. Available: https://help.steampowered.com/en/faqs/view/571A-97DA-70E9-FF74 (visited on 08/27/2024).

[13] Valve Developer Community. "Valve anti-cheat - valve developer community". (2024), [Online]. Available: https://developer.valvesoftware.com/wiki/Valve_Anti-Cheat (visited on 08/27/2024).

[14] Valve Corporation. "Vac integration (steamworks documentation)". (2024), [Online]. Available: https://partner.steamgames.com/doc/features/anticheat/vac_integration (visited on 08/27/2024).

[15] Valve Corporation. "Steam support: Banned by game developer (game ban)". (2024), [Online]. Available: https://help.steampowered.com/en/faqs/view/46DB-4CEC-F7E9-49E5 (visited on 08/27/2024).

[16] J. McDonald. "GDC 2018: John McDonald (Valve) - Using Deep Learning to Combat Cheating in CSGO". (2018), [Online]. Available: https://www.youtube.com/watch?v=0bhK8lUfIlc&t=101s (visited on 08/27/2024).

[17] Kombucha. "Valve using machine learning and deep learning to catch cheaters on cs:go - technology and operations management". (2018), [Online]. Available: https://d3.harvard.edu/platform-rctom/submission/valve-using-machine-learning-and-deep-learning-to-catch-cheaters-on-csgo-794-words/ (visited on 08/27/2024).

[18] D. Cowley. "Epic games acquires kamu, game security and player services company". (2018), [Online]. Available: https://www.unrealengine.com/en-US/blog/epic-games-acquires-kamu-game-security-and-player-services-company (visited on 08/27/2024).

[19] Epic Games, Inc. "Easy anti-cheat". (2024), [Online]. Available: https://www.easy.ac/en-US (visited on 08/27/2024).

[20] Epic Games, Inc. "EPIC GAMES PRIVACY POLICY". (2024), [Online]. Available: https://www.epicgames.com/site/en-US/privacypolicy (visited on 08/27/2024).

[21] AnonymModz. "Easy anti-cheat: Detection & bypass insights". (2023), [Online]. Available: https://anonymmodz.com/blog/EAC (visited on 08/27/2024).

[22] BattlEye Innovations. "Battleye – the anti-cheat gold standard". (2013), [Online]. Available: https://www.battleye.com/about/ (visited on 08/27/2024).

[23] M. L. Han, B. I. Kwak, and H. K. Kim, "Cheating and detection method in massively multiplayer online role-playing game: Systematic literature review", *IEEE Access*, vol. 10, pp. 49 050–49 063, 2022, ISSN: 2169-3536. DOI: 10.1109/access.2022.3172110.

[24] 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*, vol. 14, no. 4, pp. 1–27, Dec. 2008, ISSN: 1557-7406. DOI: 10.1145/2043628.2043633.

[25] A. Zhou, "A never ending warfare: Fighting cheaters in online video game", *Applied and Computational Engineering*, vol. 5, no. 1, pp. 34–39, May 2023, ISSN: 2755-273X. DOI: 10.54254/2755-2721/5/20230525.

[26] mirageofpenguins. "/dev/null: Anti-cheat kernel driver". (2020), [Online]. Available: https://www.leagueoflegends.com/en-us/news/dev/dev-null-anti-cheat-kernel-driver/ (visited on 08/27/2024).

[27] A. Neekhara. "Difference between operating system and kernel". (2024), [Online]. Available: https://www.geeksforgeeks.org/difference-between-operating-system-and-kernel/ (visited on 10/22/2024).

[28] I. Mironov. "Hash functions: Theory, attacks, and applications". (2005), [Online]. Available: https://www.microsoft.com/en-us/research/wp-content/uploads/2005/11/hash_survey.pdf?msockid=0feb7ac73778687d018068983650699e (visited on 08/28/2024).

[29] S. S. Chawathe, "Effective whitelisting for filesystem forensics", in *2009 IEEE International Conference on Intelligence and Security Informatics*, IEEE, 2009. DOI: 10.1109/isi.2009.5137284.

[30] D. Ray and J. Ligatti, "Defining code-injection attacks", *ACM SIGPLAN Notices*, vol. 47, no. 1, pp. 179–190, Jan. 2012, ISSN: 1558-1160. DOI: 10.1145/2103621.2103678.

[31] M. Wijkstra, K. Rogers, R. L. Mandryk, R. C. Veltkamp, and J. Frommel, "Help, my game is toxic! first insights from a systematic literature review on intervention systems for toxic behaviors in online video games", in *Companion Proceedings of the Annual Symposium on Computer-Human Interaction in Play*, ser. CHI PLAY '23, ACM, Oct. 2023. DOI: 10.1145/3573382.3616068.

[32] Y. Kou and X. Gui, "Flag and flaggability in automated moderation: The case of reporting toxic behavior in an online game community", in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, ser. CHI '21, ACM, May 2021. DOI: 10.1145/3411764.3445279.

[33] Microsoft. "What is: Multifactor authentication - microsoft support". (2024), [Online]. Available: https://support.microsoft.com/en-us/topic/what-is-multifactor-authentication-e5e39437-121c-be60-d123-eda06bddf661 (visited on 08/28/2024).

[34] J. Zhang, X. Tan, X. Wang, A. Yan, and Z. Qin, "T2fa: Transparent two-factor authentication", *IEEE Access*, vol. 6, pp. 32 677–32 686, 2018, ISSN: 2169-3536. DOI: 10.1109/access.2018.2844548.

[35] P. Golle and N. Ducheneaut, "Preventing bots from playing online games", *Computers in Entertainment*, vol. 3, no. 3, pp. 3–3, Jul. 2005, ISSN: 1544-3574. DOI: 10.1145/1077246.1077255.

[36] Digital Millennium Copyright Act Services Ltd. “What is a dmca takedown?” (2023), [Online]. Available: https://www.dmca.com/FAQ/What-is-a-DMCA-Takedown (visited on 08/28/2024).

[37] J. Zetterström. “A legal analysis of cheating in online multiplayer games”. (2005), [Online]. Available: https://gupea.ub.gu.se/bitstream/handle/2077/1948/200528.pdf?sequence=1 (visited on 08/28/2024).

[38] Supreme Court of Korea. “Chapter 32 gambling and gaming crimes”. (2015), [Online]. Available: https://sc.scourt.go.kr/sc/engsc/pdf/Chapter_32_Gambling_and_Gaming_Crimes.pdf (visited on 08/28/2024).

[39] J. Tidy. “Police bust ’world’s biggest’ video-game-cheat operation”. (2021), [Online]. Available: https://www.bbc.com/news/technology-56579449 (visited on 08/28/2024).

Appendix A DDOS attack framework

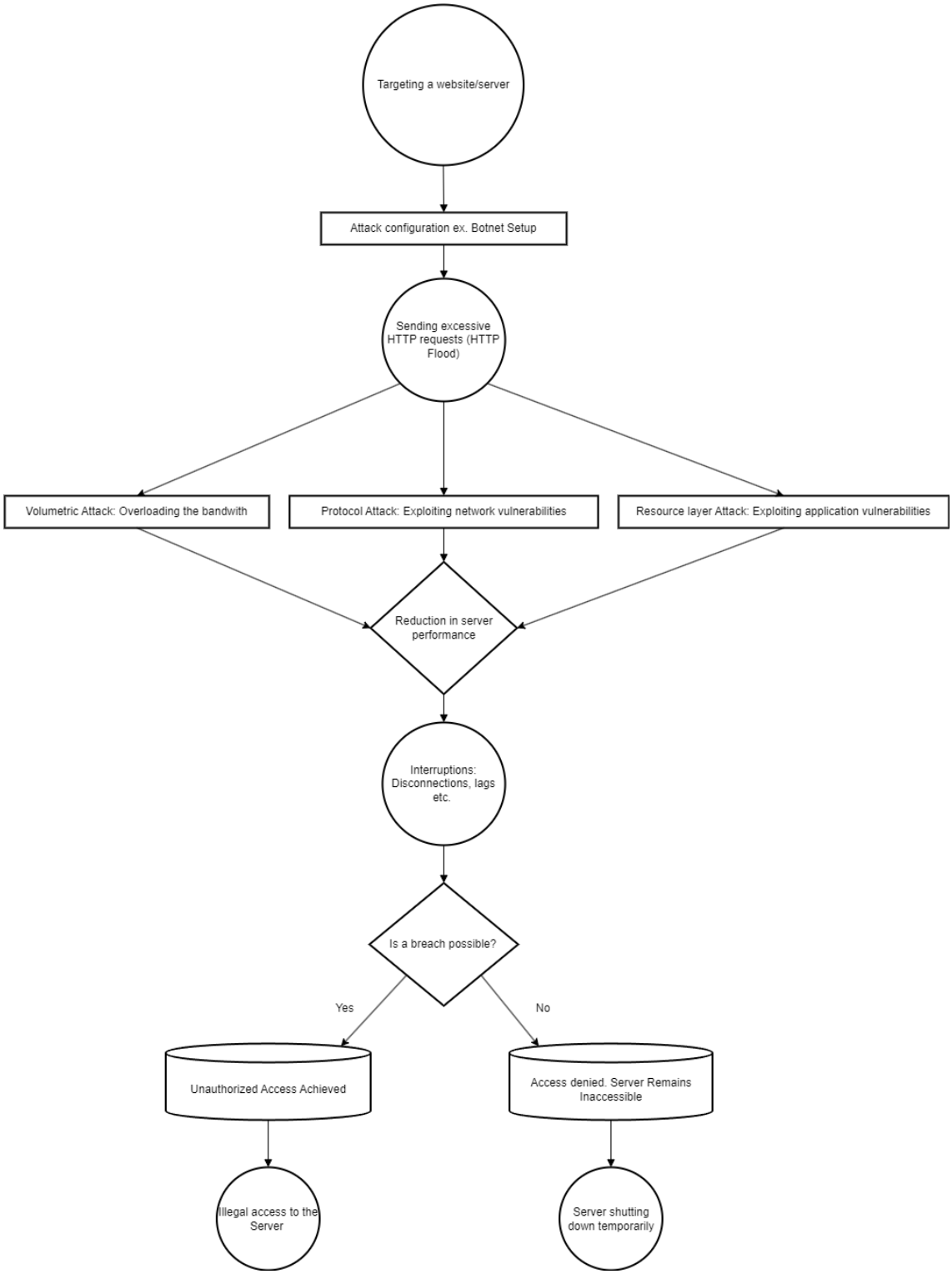


Figure A.1: DDOS attack framework [10]