How Risky Is It? A Closer Look at Game Anti-Cheat Software

Amanda Viescinski
Fed. University of Paraná
abviescinski@inf.ufpr.br

Tiago Heinrich
Max Planck Institute for Informatics
theinric@mpi-inf.mpg.de

Vinicius Fulber-Garcia
Fed. University of Paraná
vinicius@inf.ufpr.br

Carlos Maziero
Fed. University of Paraná
maziero@inf.ufpr.br

***Abstract*—Anti-cheat software is a system designed to detect cheats (or hacks) in a video game. This paper investigates operations executed by anti-cheat software and their impact on user privacy. We collected data and analyzed three popular anti-cheat solutions: BattlEye, FACEIT, and Vanguard. Our analysis reveals that these programs interact with system files, memory, and users' directories. In addition, the privileged access of these anti-cheat solutions to the operating system and the lack of clarity on what data is collected directly affect user privacy. This conduct risks not complying with current regulations.**

***Index Terms*—Anti-cheat, Software Analysis, Security**

I. Introduction

The video game industry increased its revenue from US$152.1 billion before COVID-19 to US$187.7 billion in 2024, according to Newzoo's Global Games Market Report [1]. This scenario emerged with significant financial appeal in video games and a growing competitive scenario for its players, who can seek victory at any cost, even using cheat software. In particular, cheat software, also known as hacks or simply cheats, is designed to provide an unfair advantage to a player. It manipulates the game's operations and resources, altering its fundamental properties.

Game studios implement their countermeasures against cheating software and malicious players to prevent legitimate players from suffering financial losses and a degraded experience [2]. One well-known strategy is anti-cheat software, which helps prevent and detect cheaters. This software is typically installed on players' machines and actively collects system data to analyze their actions within the game.

However, anti-cheat solutions are becoming increasingly intrusive to improve the detection of malicious players in this cat-and-mouse game, requiring comprehensive permissions and high privileges on players' devices. For example, many anti-cheat alternatives run alongside the system, even when the game is inactive. Moreover, their actions on the device are obfuscated to make reverse engineering by cheat developers as tricky as possible, preventing the discovery of weaknesses that could be exploited [2]. Additionally, to perform detailed analysis of specific data, some anti-cheat software frequently sends files from players' machines for remote verification on a server. All these actions raise significant concerns about player privacy, as it is not clear exactly what data is being collected by anti-cheat solutions [3].

Examples of potential privacy breaches caused by anti-cheat software are widespread and well-documented. As mentioned in [4], after an update to the anti-cheat system of the game World of Warcraft, part of the community felt their privacy was compromised and, in order to continue playing, chose to close all unnecessary applications, with some even abandoning the game entirely. In [5], it is noted that specific anti-cheat solutions have prevented players from connecting to servers simply by using the word "cheat" in web browsers. Furthermore, the Vanguard anti-cheat system, developed by Riot Games, initially erroneously detected software and basic keyboard and mouse drivers as cheats, leading to malfunctions in essential system drivers and interfering with crucial tasks, such as cooling system components, causing computers to overheat [6].

The anti-cheat solutions mentioned and many others used in modern games operate with kernel-level privileges on players' devices, granting them access to various private user information. This situation can pose a significant privacy issue for benign players. Therefore, we must question whether it is worth or necessary to implement such a high level of surveillance to detect unfair activities and whether alternative solutions do not operate at the kernel level yet still effectively address the cheating problem.

This paper analyzes three popular anti-cheat software solutions and their interactions with the operating system. We monitor the activity and operations of these anti-cheat solutions, including access to directories, granted permissions, network communications, and queries or modifications to records, identifying aspects that may compromise player privacy. Our objective is to provide a comprehensive profile of modern anti-cheat solutions, highlighting differences and similarities in their operational characteristics, pointing out privacy risks, and supporting the community with technical evidence that demonstrates the need for a paradigm shift in the anti-cheat software industry, where players' privacy is also a concern.

Specifically, our main contributions are: (a) a detailed analysis of three popular anti-cheat solutions, including uncovering their operational flow; and (b) a comprehensive discussion on the impact of anti-cheat software on user privacy and security.

The rest of this paper is organized as follows: Section II explains the key concepts; Section III discusses the challenges associated with anti-cheat software; Section IV details our proposal; Section V presents our observation; Section VI explains General Data Protection Regulation (GDPR); Section VII discusses the experiments and results; Section VIII relates research in the area; finally, Section IX concludes the work.

II. Anti-Cheat Software

Players use cheat software to gain an advantage in a video game through illegitimate means [3]. It can be categorized into several types. Among them are those that do not alter the game's fundamental rules – that is, they do not perform actions that would be impossible under normal game conditions. An example of this cheat software type is bots, designed to perform automated actions within games. These bots typically accumulate resources over extended periods without requiring player interaction.

The fundamental rules of games can also be exploited by cheat software. This means that the expected interactions between players and the game's digital world can be violated, granting cheaters advantages that should be impossible to achieve. These cheats typically employ code injection techniques to modify memory data, actively altering game behavior or capturing information to execute flawless actions (e.g., aimbots, wallhacks) [3].

Countermeasures against such cheats are known as anti-cheat software. These programs are designed to detect cheats [6]. Once a cheat is detected, an anti-cheat solution can impose penalties on the malicious player, ranging from gameplay restrictions to outright bans [7]. Anti-cheat systems also maintain records of players who have used cheats. Some are more aggressive, enforcing bans across all games that use the same anti-cheat software after a single detection [4].

Anti-cheat solutions employ various strategies to detect cheating. These include detecting anomalous patterns in server data, monitoring for sudden performance spikes, and identifying inconsistent or unauthorized actions. Additionally, they perform file integrity checks and memory scanning while the game is running on the user's machine [8]. Such solutions can also be categorized into two broad types based on where they are executed: server-side and client-side. Server-side solutions operate based on the game rules defined by the server, and every action performed by players and each game state is validated on the server [8]. Client-side anti-cheats are installed directly on players' machines, analyzing the computer's memory and being capable of detecting even discreet and obfuscated cheats – but they are vulnerable to reverse engineering, modifications, and code injections by cheaters [6].

Examples of popular anti-cheat solutions currently available on the market and widely used include Easy Anti-Cheat, FACEIT, BattlEye, Valve Anti-Cheat, and Vanguard [9], [10].

III. The Security Problem

Anti-cheat software may require access to the kernel level of computing systems where they run, raising concerns among users about potential violations of their privacy and the instability such software could introduce to the system. Furthermore, any vulnerability in this software could grant privileged access to attackers, compromising system security [11]. An additional concern is that some anti-cheats are launched before their respective game is even started. For instance, Vanguard runs as soon as the user's operating system starts [6].

Although anti-cheat solutions' consent forms briefly discuss the strategy behind cheat detection and the use of unauthorized software, these terms generally focus on the actions to be performed and the information to be collected. They also warn that only data essential for detecting cheating will be analyzed. However, the scope of these needs is not clearly defined, and the details of what constitutes a fundamental analysis are often unclear or obscured [2].

Anti-cheat solutions have already led to recorded cases of privacy invasion. A well-known example is the case of VAC, Valve's anti-cheat system, which was found to be directly accessing users' search history [8]. There have also been instances of security breaches, such as with the anti-cheat solution used in the game Genshin Impact, mhyprot2.sys, which contains a vulnerability allowing attackers to disable antivirus software running on players' machines [11].

A notable case involving anti-cheats occurred with the ESEA anti-cheat system. Like Vanguard, ESEA remains active even when a monitored game is not running. Because it operates with privileged permissions on players' computers and works hiding its actions, an employee of its developer company inserted bitcoin mining malware, which ran covertly, without users' consent [2]. Such factors influence the risk this type of software can bring to users and demonstrate the importance of understanding how this monitoring is carried out.

IV. Proposal

Our proposal aims to investigate the actions of anti-cheat software and their impact on user privacy. For that, we capture relevant data to evaluate the actions performed by anti-cheat software, such as changes in system registry, permissions granted to processes, directories accessed, dynamic link libraries (DLLs) used, and data transmitted over the Internet.

We deployed the monitoring on a computer with Windows 10 Pro (22H2, KB5046613) using Process Monitor (4.01) and Process Explorer (17.05). Each game version was tested with its corresponding anti-cheat version. Tibia (14.10.db74b1) was considered with the BattlEye anti-cheat, while Valorant (9.06) was selected with Vanguard. Additionally, we monitored the FACEIT (2.0.26) independently, as it does not require a game to run. Three separate captures were made for each anti-cheat analyzed to avoid incompatibility.

We identified a difference in how the anti-cheats integrate with the system. BattlEye runs independently of the system's life cycle, initiating and ending together with the game. In contrast, Vanguard and FACEIT utilize a driver that launches alongside Windows by default. Consequently, some of their behavior was lost until the monitoring tools loaded. To address this, we adjusted the anti-cheats startup times to ensure they launched after both the system and the monitoring tools. This did not affect game behavior. Despite this, monitoring remained effective, as anti-cheats launch a process with their respective games, which served as the monitoring target. Data collection involved: (I) starting the monitoring tools, (II) launching the game/anti-cheat, (III) capturing data, (IV) closing the game/anti-cheat, and (V) saving and stopping the tools.

The BattlEye and FACEIT anti-cheats allowed the monitoring tools to be executed fully alongside the games. The executions conducted for both anti-cheats were about one hour apart. For FACEIT, some data was collected with a user performing standard operations on the system, while others were collected without interference.

Vanguard's data capture presented some challenges. An error was displayed whenever the Process Monitor tool was operating, requiring the system to be restarted for the game to run correctly again. In an attempt to avoid erroneous or unusual system configurations, no changes were made to get around this problem. Therefore, complete captures of quick matches were taken while the game was still running, with the matches being finished before the error was displayed.

V. Observation

We observed that anti-cheats follow a four-phase pattern: *Initialization*, *Setup*, *Loop*, and *Termination*. The **Initialization** phase is the point at which the anti-cheat begins to execute. In the **Setup** phase the anti-cheat software loads dependencies and performs an initial scan. During the **Loop** phase, anti-cheats monitor common actions for cheating software, behavior changes, and unauthorized interactions. The **Termination** phase occurs when the anti-cheat system is deactivated, completing its execution and ending all operations. Table I summarizes these phases for the analyzed anti-cheats.

A. Execution Pattern

This section analyzes the operations executed by the considered anti-cheat solutions. The analysis focuses on the unique behaviors of each anti-cheat, detailing their interactions with the operating system, registers, files, and remote servers.

1) ***BattleEye****:* As previously mentioned, the BattleEye anti-cheat begins executing with the game it protects. It accesses necessary resources, such as registers and DLLs, validating them. One of the registers accessed is called *CodeIdentifiers*, which stores a log file path with details on running applications, their parent process, and execution permission rules. This registry is part of the Windows Safer API, which manages application execution policies [12]. We also observed registers containing network and control panel information being accessed. In the setup phase, the process runs its executable and creates a *.sys* file, usually containing device drivers or hardware configurations.

Unlike other anti-cheats, BattlEye operates through three execution main loops, each performing distinct functions. DLL loading, accessing encryption registers and certificates, and forwarding UDP packets are executed across all loops. The first loop verifies DLL integrity using hashes and the *CryptSIPDllVerifyIndirectData* function. The second loop involves read/write operations in the *CatRoot* directory (which stores system certificates) and disk mapping data. The third loop performs unique tasks such as validating a specific game DLL, accessing records in shared folders, and manipulating files in the *AppData* directory. Notably, these files contain certificates and encrypted data.

2) ***FACEIT****:* FACEIT starts alongside the operating system, initializes threads, checks registers, loads and validates DLLs, and loads web application files (*e.g.*, JS, JSON, node, and ASAR formats). Furthermore, the process creates cache, temporary, and log files. In an isolated test where only Vanguard and FACEIT ran simultaneously, we observed that FACEIT also queried Vanguard log files during these operations. More specifically, FACEIT executed *WriteFile* and *FlushFileBuffers* operations, which write data from buffers to files and then clear the buffers. Still, in the setup phase, the process creates child processes and executes the application's *.exe* file.

During the loop phase, the FACEIT parent process performs I/O operations on the cache and log files while consulting registers with network information. The child processes exchange TCP and UDP packets. An investigation showed that multiple communicated addresses were linked to Google and Cloudflare, suggesting they may be CDNs. At the end of the process, FACEIT stops accessing log files, DLLs, and registers, finally terminating threads before shutting down.

3) ***Vanguard****:* Vanguard anti-cheat has two processes: VGTray, which starts with the operating system, and VGC, which runs with the game. Due to their distinct characteristics and their different operations, VGTray and VGC are represented separately in Table I. The VGTray Setup phase consists of creating threads, consulting registers, accessing and verifying DLL hashes, creating prefetch files, running its executable, and manipulating the *vgkboostatus.dat* file. The *.dat* files can store various types of data, including functions required by DLLs, images, and audio.

In the loop phase, the process creates a thread, queries the executable, and accesses the *vgk* service registry before terminating the thread. It is important to mention that the process remains in this phase for approximately 87% of its total execution time. Towards the end of execution, multiple TCP connections are established, and packets are transmitted. Finally, the registers consulted, DLLs, *.dat* files, threads, and the process are closed.

When the Valorant game starts, the VGC process is executed and starts the setup phase. In this phase, the operations are similar to those of VGTray, plus access files in the *CatRoot* directory and handle application logs. Then, TCP connections are established, and registers containing user names and monitor data are consulted. Additionally, various system directories are accessed, especially those for storing personal files, such as OneDrive and Desktop. This phase includes the generation of new logs and the transmission of TCP packets to Vanguard's servers. Finally, the anti-cheat writes additional logs, closes registers and active threads, and shuts down.

B. Behavior comparison

This section analyzes the similarities and differences between the operations executed by the anti-cheats investigated at each phase of their life cycle. Although the end goal is the same (detecting cheats), each software uses different approaches. This variation directly influences the resources accessed and how much information is collected.

TABLE I
ANTI-CHEAT EXECUTION FLOW AND OPERATIONS.

Anti-Cheat	Initialization	Setup	Loop	Termination
BattlEye	Game	Create threads Query registers Access DLLs Check DLLs hashes Run BEService.exe Create BEDaisy.sys files	Load DLL Access crypto registers & certificates Check DLL hash Close DLL Forward UDP packet Load DLL Read/Write CatRoot files Check DLL hash Create disk mapping Access crypto registers & certificates Close DLLs and CatRoot files Forward UDP packet Load Tibia DLL Access crypto registers & certificates Check DLL hash Access registers about user shell folders Read/Write AppData directory Close Tibia DLL Forward UDP packet	Query registers Read/Write BattlEye's .sys Close DLL Close threads Close open registers Close process
FACEIT	Operating System	Create threads Query registers Access DLLs Check DLLs hashes Load JavaScript/Node.js files Read/Write Cache, Temp & Log files Start child processes RunFACEIT.exe file	Write to cache and log files Query registers with network information Child processes exchange TCP/UDP packets	Close cache and log files Close DLL Closes access to open registers Close threads Close the process
Vanguard (VGTray)	Operating System	Create threads Query registers Access DLLs Check DLLs hashes Create a prefetch file Run vgtray.exe file Read/Write vgkboostatus.dat file	Create threads Query vgtray.exe Accesses vgk service registers Close threads	Establish TCP connections TCP stream Close .dat files Close DLL Close open registers Close threads Close the process
Vanguard (VGC)	Game	Create threads Query registers Access DLLs Check DLLs hashes Create a prefetch file Read/Write vgkboostatus.dat file Run vgc.exe Access CatRoot files Read/Write Log files	Establish TCP connections Query registers Request directory information Write logs TCP packets exchange	Write logs Close open registers Close threads Close the process

*1) **Initialization Phase**:* Anti-cheats activate at different times. FACEIT and VGC/Vanguard start with the operating system, whereas BattlEye and VGTray/Vanguard launch only when the game runs. Regardless of their startup time, all operate at the kernel level, ensuring privileged access to the system. This access is crucial for detecting and mitigating cheats, which can also function at this level [13]. By loading their drivers during system startup, anti-cheats can verify the integrity of other drivers and critical system resources. Anti-cheats that launch with the game use alternative methods to validate the environment, such as scanning RAM and analyzing its contents for known cheat signatures or suspicious activity.

*2) **Setup Phase**:* Overall, BattlEye, FACEIT, and Vanguard all start their execution by executing similar operations. First, they access registers like *CodeIdentifiers*, described earlier, and *CustomLocale*, which defines several aspects of the system's locale. Furthermore, they execute their respective main files in *.exe* format. On the other hand, these anti-cheats also execute different operations. BattlEye is the only one that creates *.sys* driver files, although they all access DLLs or registers related to drivers. FACEIT, however, differs in creating numerous child processes throughout its execution. Although both Vanguard processes (VGC and VGTray) share operations in the setup phase, VGC performs operations such as accessing files in the *CatRoot* folder and reading and writing logs - characteristics not observed in VGTray.

*3) **Loop Phase**:* In the loop phase, most of the operations of BattlEye and Vanguard anti-cheats involve inspecting system files. BattlEye concentrates on checking the hash of DLLs. Notably, it transmits an UDP packet to its server after each check. Among the DLLs analyzed are even those related to Windows Defender settings. In contrast, VGC/Vanguard performs a comprehensive scan of system files, including the user's home directories. FACEIT, on the other hand, primarily manages cache and temporary files from both the application and the system. Additionally, during this phase, anti-cheats continuously exchange packets with their servers.

The directories analyzed by VGC/Vanguard include personal ones such as Documents and OneDrive, as well as operating system directories including *system32*, *programfiles*, and *App-*

data. An official statement from Riot Games [13] emphasizes that Vanguard scans memory signatures looking for patterns compatible with those used by cheating software, just like other anti-cheats. After this operation, there is a significant increase in communication traffic with its server via TCP, similar to the behavior of BattlEye after analyzing the DLLs.

About DLLs accessed, around 21% of the entire list is shared between all the anti-cheats. These DLLs allow access to various components, including data about hardware and devices, as well as events provided by the *ntdll* and *ntmarta* DLLs and kernel-level functions with the *kernel32* and *KernelBase* DLLs. Further libraries retrieve system details, such as the file system, running processes, and loaded drivers. For this purpose, the *SHCore*, *shell32*, and *shlwapi* DLLs enable the execution of PowerShell commands. One of the DLLs responsible for the Remote Procedure Call Protocol (RPC) is also accessed by anti-cheat solutions, *rpcrt4*. RPC allows communication between processes on distinct computers, involving interaction between remote devices. Moreover, the *wldp* DLL, a Windows Defender Application Control component, helps to restrict the execution of programs and scripts on the system.

The DLLs exclusively accessed by each anti-cheat reveal distinct monitoring strategies and system interactions. BattlEye, for instance, utilizes: i) *sfc*, responsible for checking Windows system files, which can be used to analyze file integrity and restore corrupted or damaged files; ii) *mpr*, in charge of managing communication with shared devices and directories in a network; and iii) *ncrypt*, used to manage cryptographic keys, data encryption and decryption operations, authentication and protection of sensitive data. Within the set of DLLs used only by FACEIT, we can highlight *dbghelp*, which provides functions for debugging, reading stack traces, and manipulating executable image symbols in a portable format (e.g., *.exe*, *.sys*). It can, therefore, be used to inspect processes and system memory. Vanguard, in contrast, incorporates the legacy Internet Explorer *iertutil* DLL, which supports creating, manipulating, and closing windows and tabs in the web browser.

4) ***Termination Phase****:* In the termination phase, all the anti-cheat solutions close threads, terminate access to open registers, and finalize their execution. In addition to these operations, BattlEye directly manipulates system files in *.sys* format before concluding its activity. FACEIT, however, closes cache and log files, indicating more detailed management of this data. VGTray, in contrast, establishes TCP connections and transmits data before closing *.dat* files, while VGC, in a more simplified way, records logs before finishing.

VI. Software Licenses against GDPR

The General Data Protection Regulation (GDPR) [14] establishes the privacy laws of European Union citizens and defines the responsibilities of entities when processing personal data. In this case, personal data is information about an identifiable natural person. The GDPR consists of 99 articles that detail its legal requirements, plus 173 recitals that provide context and complementary explanations to these articles. This paper considers the GDPR articles to discuss the potential lack of privacy in anti-cheat solutions.

It is essential to clarify some definitions to better understand the concepts covered by the regulation. Processing refers to any operation carried out with personal data, whether by automated means or not. The controller defines the purpose and processing methods, while the processor acts on their behalf. A third party is any entity other than the data subject, controller, processor, or their authorized agents.

Under the GDPR, software that handles personal data must comply with principles such as data minimization, purpose limitation, and transparency (Article 5th). In addition, the regulation imposes data protection by design and default (Article 25th), requiring solutions to implement data protection measures from the outset and collect only the strictly necessary information. Furthermore, software must guarantee secure data processing (Article 32nd) using encryption, access controls, and pseudonymization wherever possible.

In addition to the requirements for secure processing, the GDPR establishes specific guidelines for storing data collected by software such as anti-cheat systems. Article 32nd requires appropriate security to prevent unauthorized access, loss, or alteration. Furthermore, Article 5th limits retention to what is strictly necessary, in line with the principle of data minimization, while Article 6th defines legal bases such as consent, contracts, or legitimate interest. In addition, Article 17th ensures the right to be forgotten, allowing users to request data deletion when it's no longer needed.

The GDPR also imposes specific governance requirements to ensure data storage transparency. Article 30th obliges entities to keep detailed records of their processing activities, including the purpose of collection, the types of data stored, and the security measures adopted. Article 33rd establishes that if there is a personal data breach, the responsible entity must notify the supervisory authority within 72 hours and notify affected users, if necessary.

VII. Discussion

Anti-cheat software lacks transparency regarding the operations it performs. The analysis of the execution flow we mapped and software licenses reveals that these applications, in fact, access system files, memory, and even users' directories. It is sufficient evidence to state that anti-cheats represent a potential threat to users' privacy. The level of access granted to anti-cheat software makes them critical for system security since these solutions can be attacked and/or compromised by malicious entities. In these cases, attackers can gain comprehensive access and control over the system.

Another point of attention refers to the characteristics that define software as invasive of user privacy. An appropriate definition is to consider as invasive any application that collects and transmits information that can be used to identify an individual [15], which is very similar to the operating pattern of the anti-cheat software analyzed.

The lack of transparency behind anti-cheat software becomes yet another criterion that makes its use dangerous. If this

software is compromised, the detection of malicious operations becomes almost impossible. Our analysis demonstrates a software operation cycle that reaches key elements of the operating system, which directly impacts user privacy.

User data is being collected and stored on external servers. This information, previously only found on the user's machine, is now found on external servers, where data protection is not fully presented to users since not even the data being exfiltrated is disclosed. This situation can leak sensitive data, such as user names, emails, and passwords. Although all servers connected to the Internet are subject to attacks and leaks, the information stored by anti-cheat is particularly compromising, as it can store any file on users' computers, including banking information written down in a simple text file.

GDPR highlights a way to enforce users' privacy respect when considering this niche of intrusive software. However, changes in the execution model are necessary, considering the adoption of less intrusive strategies, such as performing analysis of game rules centrally on servers, which can help to maintain both player privacy and justice in gaming environments.

VIII. Related Work

Recent works discussed anti-cheat operations and their implications for user privacy. In [8], the issue of privacy invasion and security problems caused by *kernel-level* anti-cheats is considered, providing alternative options for detection systems, such as mirroring game rules on the server and statistical analysis of player events and performance. The work points out that *anti-cheat* solutions installed with *kernel* permission can cause problems even with operating system updates and, with the strategy of scanning the entire memory of the device, raise numerous suspicions of invasion of privacy of its users.

In [3], the differences between preventive and detective solutions for cheating in *online* games are explored. The study exposes several privacy and security problems in anti-cheat solutions, such as the memory scanning strategy that checks information from other processes running on users' machines. *BattlEye*'s ability to upload user files to its servers and execute *shell* code directly on users' computers remotely is also mentioned. The paper discusses how a deterrent alternative could be designed and integrated during game development, avoiding the need for intrusive third-party *cheat* monitoring and detection solutions.

In [16], the similarities and differences between anti-cheats and rootkits are analyzed, specifically at the kernel level. To this purpose, seven typical rootkit behaviors are identified, including evasion, execution time, information exfiltration, and network manipulation. The investigation relied exclusively on publicly available data from discussion forums and anti-cheat documentation. As a result, the study concluded that BattlEye and Easy Anti-Cheat did not exhibit enough characteristics to be classified as rootkits. On the other hand, the FACEIT and Vanguard anti-cheats met at least four of the seven criteria established, coming closer to the behavior observed in rootkits.

Although such works conducted discussions, none executed empirical tests to trace the running behavior of the anti-cheat solutions and to analyze the potential privacy issues.

IX. Conclusion

This article presented an analysis of three popular anti-cheat solutions, examining their interactions with the operating system and their implications for user privacy. The life cycle mapped for each anti-cheat indicates that these programs access sensitive data such as system files, personal directories, and memory. In this sense, the method of analyzing the data accessed raises concerns about compliance with privacy regulations. In particular, there is a lack of transparency about what data is collected for processing. In addition, the fact that the data is processed on external servers could potentially result in sensitive user data leaking. These findings underscore the need to revise anti-cheat terms of use to clarify data handling practices, and to develop solutions that minimize system privileges and prioritize local, privacy-preserving processing.

ACKNOWLEDGEMENTS

This study was financed in part by the *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES)* – Finance Code 001 and *Fundação de Amparo à Pesquisa e Inovação do Estado de Santa Catarina (FAPESC)*.

References

[1] Newzoo, "Newzoo's global games market report 2024," 2024.
[2] R. Greidanus, "Client-side anti-cheat in online games: Legal implications from a privacy and data protection perspective," Ph.D. dissertation, Tilburg University, Tilburg, Netherlands, 2017.
[3] W. Ronkainen, "Prevention vs detection in online game cheating," Ph.D. dissertation, University of Oulu, Oulu, Finland, 2021.
[4] E. Wendel, "Cheating in online games a case study of bots and bot-detection in browser-based multiplayer games," Ph.D. dissertation, Norwegian Univ. of Science and Technology, Trondheim, Norway, 2012.
[5] S. Pontiroli, "The cake is a lie! uncovering the secret world of malware-like cheats in video games," *Virus Bulletin*, 2019.
[6] B. Ven, "Cheating and anti-cheat system action impacts on user experience," Ph.D. dissertation, Radboud Univ., Netherlands, 2023.
[7] X. Lan and et al., "An overview on game cheating and its countermeasures," in *2nd Intl Symposium on Computer Science and Computational Technology*, Huangshan, China, 2009, p. 195.
[8] A. Maario and et al, "Redefining the risks of kernel-level anti-cheat in online gaming," in *ISCSCT*, 2021, pp. 676–680.
[9] S. Pilipovic, "Every game with kernel–level anti–cheat software," 2023, https://tinyurl.com/23f2nbn2.
[10] PCGamingWiki, "List of games with anti-cheat technology," 2023, https://tinyurl.com/3cwc2ays.
[11] I. Basque-Rice, "Cheaters could prosper: An analysis of the security of video game anti-cheat," Honours Project Proposal, School of Design and Informatics, Abertay University, 2023, https://tinyurl.com/mpf9km5z.
[12] Microsoft, "Determine allow-deny list and application inventory for software restriction policies," 2024, https://tinyurl.com/5x4eujft.
[13] Riot Games, "/DEV: Vanguard X LoL," 2024, https://tinyurl.com/bvrxxp2t.
[14] GDPR, "Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46," *Official Journal of the European Union*, vol. 59, pp. 1–88, 2016.
[15] M. Boldt and B. Carlsson, "Privacy-invasive software and preventive mechanisms," in *ICSNC*, 2006, pp. 21–21.
[16] C. Dorner and L. D. Klausner, "If it looks like a rootkit and deceives like a rootkit: A critical examination of kernel-level anti-cheat systems," in *Proceedings of the 19th ARES*, 2024, pp. 1–11.