Video Game Cheating and Malicious Hacking: Exploring Technical Overlap and Opportunities for Cross-Industry Learning and Collaboration

Berk Can Altindag[1,†], Linus Eriksson[1,†], Joakim Kävrestad[1] and Görkem Kılınç Soylu[1,*]

[1]Department of Computer Science and Informatics, School of Engineering, Jönköping University, Sweden

Abstract

Video game cheating and hacking are two fields that share numerous similarities in offensive techniques; however, defenses in gaming and cybersecurity contexts are often developed independently, resulting in limited knowledge transfer and collaboration. As several attacks operate similarly across both domains, this study investigates the overlap between two commonly used offensive techniques, code injection and reverse engineering, in cheating and hacking. Building on these technical similarities, the study also examines opportunities and barriers for cross-industry collaboration to strengthen defenses within both gaming security and cybersecurity. The study is based on semi-structured interviews with professionals from the anti-cheat and cybersecurity domains to explore technical overlap and opportunities for cross-industry learning. The findings reveal several overlaps between the identified techniques, including accessibility of attacks and the misuse of trusted systems or processes, such as vulnerable drivers, to conduct attacks and evade detection. Reverse engineering also showed notable similarities, as attackers and defenders use the same tools and similar methods to uncover vulnerabilities in software. The findings identify opportunities for cross-industry learning by sharing defensive knowledge, practices, and expertise, which could improve proactive defense and increase the availability of defensive talent across both fields. However, some barriers hinder the potential for organizational and technical collaboration.

Keywords

Video game cheating, anti-cheat, cybersecurity, code injection, reverse engineering, cross-industry collaboration, cross-industry learning

CEUR-WS.org/Vol-4334/paper17.pdf

1. Introduction

Video games have provided millions of individuals with the opportunity to hone their skills and compete against others. Consequently, competitive gaming has experienced a steady rise in popularity over recent decades, with many games hosting tournaments that offer substantial rewards. However, with increasing competition and prizes to be earned, this has given rise to several unethical practices, mainly cheating [1].

Cheating undermines the competitive integrity of games by exploiting vulnerabilities in a game's code, thereby providing cheaters with unfair advantages over others [2]. Although cheating may result in in-game sanctions when detected [3], the relevance to this study is the technical overlap with malicious hacking rather than any legal equivalence between the two. Some cheating techniques resemble those used in malicious hacking, including Code Injection (CI) and Reverse Engineering (RE), which are used in both domains to discover or exploit weaknesses in software [4, 5, 6].

Several studies have suggested a relationship between video game cheating and cybercrime by comparing the motivations and communities of cheaters and unethical hackers [7, 8]. However, little attention has been given to the technical overlap between the two domains. Understanding these similarities is important for assessing whether technical skills developed through cheat development may transfer to cybersecurity or to cybercrime.

STPIS2026: Socio-Technical Perspectives in Information Systems, September 16-17, 2026, L'Aquila, Italy

*Corresponding author.

†Authors are listed in alphabetical order. The authors contributed equally.

albe25ug@student.ju.se (B. C. Altindag); erli22zx@student.ju.se (L. Eriksson); joakim.kavrestad@ju.se (J. Kävrestad); gorkem.kilinc.soylu@ju.se (. G. Kılınç Soylu)

© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

published 2026-11-19

Defensive mechanisms also share notable similarities. Gaming relies primarily on Anti-Cheat (AC) systems that perform integrity checking and behavioral monitoring, while cybersecurity uses Antivirus (AV) and Endpoint Detection and Response (EDR) systems to achieve comparable goals [9, 3]. Previous work has suggested that cybersecurity defensive techniques could also be adapted to gaming environments [10].

Despite the technical similarities, the gaming and cybersecurity communities have largely evolved independently, with limited sharing of defensive practices, expertise, and knowledge. This separation is not only technical: the domains differ in their organizations, professional communities, and trust relationships, as AC protects a game against manipulation by the device's own user, whereas AV and EDR protect the device against malicious software and unauthorized activity. Understanding where such knowledge can be transferred may help identify opportunities for collaboration while also revealing the barriers that limit it. This study investigates how two common offensive techniques, code injection and reverse engineering, overlap in the contexts of cheating and hacking. Building on the similarities, this study examines how anti-cheat developers and cybersecurity experts perceive opportunities and barriers for cross-industry learning and collaboration. In particular, it explores how defensive practices and expertise from one domain could inform and strengthen the other.
To achieve the aim of this study, the following research questions must be addressed:

RQ1: How do code injection and reverse engineering techniques overlap between cheating and hacking contexts?
RQ2: How do anti-cheat developers and cybersecurity experts perceive potential opportunities and barriers for cross-industry learning and collaboration?

The main contribution of this study is the identification of opportunities and barriers to cross-industry learning and collaboration between the cybersecurity and AC domains. By comparing defenses, attacks, and operational procedures, this study provides recommendations on how collaboration could be facilitated and supported to improve detection and prevention against similarly applied threats, specifically CI and RE.

2. Background

Video game cheating relies on complex technical methods that exploit vulnerabilities in game software [2]. Previous research has suggested that cheating in video games could provide a gateway into cybercriminal activities, such as unethical hacking [7, 8]. Although attack targets differ, cheating and malicious hacking share several techniques. Therefore, this study focuses on two prevalent ones found within both video game cheating and malicious hacking: code injection and reverse engineering.

2.1. Defensive Approaches

AC systems can be broadly categorized into two approaches: client-side and server-side [11]. Client-side AC systems operate directly on a user's system, allowing data from a client to be sent to the game server. They monitor the client's environment for suspicious programs, drivers, and code injection attempts. For example, Riot Games' Vanguard AC terminates matches and penalizes players when cheating is detected [12]. Since these systems run on the player's device, they reduce the load on the game server but can negatively affect device performance due to increased performance overhead [11]. Many modern client-side AC systems also operate at the kernel level, which improves resistance to tampering and allows them to monitor and restrict user-mode access to the game process. However, it also raises several concerns about player privacy, as certain kernel-level ACs exhibit characteristics similar to those of rootkits [13]. This creates a socio-technical trade-off: deeper access can improve cheat detection but requires players to trust software with privileged access to their devices.

In contrast, server-side AC systems operate on the game server and analyze the data available to it, such as player inputs, game state, and behavioral patterns [11]. Detection may occur in real time

or retrospectively. Because server-side AC does not interact with the player's operating system, it is platform-independent, whereas certain client-side AC may not work on Linux-based machines [14]. It is also resistant to client-side tampering and requires no additional software on the player's device [11]. Server-side AC therefore avoids many of the privacy and performance concerns of client-side AC, but has no visibility into the client's processes, memory, or drivers and thus cannot directly detect manipulation such as code injection.

Among cybersecurity solutions, client-side AC is most comparable to endpoint protection, i.e., AV and EDR systems, which also run on the protected device [9]. AV programs often run in the background, continuously scanning incoming data and monitoring system activity for malicious behavior [15]. EDR systems extend this by continuously recording endpoint activity, such as process and network events, to detect and respond to suspicious behavior [16]. Since malicious software developers employ increasingly sophisticated methods to evade AV systems, AV systems must be updated frequently. However, even with frequent updates, systems remain exposed to evolving threats.

2.2. Code Injection and Reverse Engineering

Code injection is one of the most common techniques applied in cheating. Karkallis et al. [17] describe injections as a cornerstone within cheating, since they provide the ability to directly inject untrusted code into a game's memory. This could be done to alter the game's behavior, granting the cheater advantages such as infinite health or accelerated movement [4]. Injection attacks in gaming may also cause disruptive actions, such as crashing a game's server. According to Martinson and Rangel [4], a common way to perform CI attacks in games is through Dynamic Link Library (DLL) injection. By injecting DLLs into the address space of the game process running on the cheater's device, the game could be manipulated to load and execute external code, thereby altering its behavior.

Injection is also common in malicious hacking, but the term covers technically different attacks. Web-application injection, such as SQL injection, Cross-Site Scripting, and shell injection, supplies malicious input that a vulnerable application passes to an interpreter [18]. Process injection, such as DLL injection, instead executes code in the address space of another running process [19]. Process injection therefore provides the closest technical counterpart to injection-based cheating. Injection attacks enable attackers to bypass defense mechanisms, compromise applications, and perform denial-of-service attacks [5].

Both the cybersecurity and the gaming domains defend against CI attacks using similar strategies. In gaming, AC systems are often used to detect CI attempts, employing techniques such as real-time monitoring and anomaly detection to identify suspicious patterns in client behavior [4]. Kernel-level AC is particularly effective because it operates at a higher privilege level than the game itself, allowing it to detect and prevent Windows user-mode attacks [3]. Similarly, cybersecurity defenses often utilize dynamic detection mechanisms to monitor a system's behavior for injection attempts [5].

Code obfuscation is another common defensive measure used to hide the inner workings of software while preserving its functionality, making it more difficult to discover injection points or other vulnerabilities [3]. Common methods of obfuscation used within both the gaming and cybersecurity domains include encryption and control-flow flattening [20]. To bypass these obfuscation techniques, RE is often applied. RE is a technique used in both cheating and hacking contexts to analyze software and systems by deconstructing their code. RE is commonly performed through either disassembly or decompilation of software to understand how it functions internally, in turn, revealing potential vulnerabilities and ways to bypass security systems [6, 20, 21].

Kanervisto et al. [22] state that RE is one of the most challenging aspects of developing hacks due to the obfuscation techniques often applied. However, they note that it is an essential practice, since understanding a program's internal logic is often required before attacks can be developed. Since RE often acts as an initiator for cyberattacks [6], applying strong defenses against RE is therefore important in both industries to hinder the discovery of potential attack vectors. In addition to obfuscation, AC and AV systems often apply process scanning techniques to complicate the RE process [3, 4]. Process scanning involves monitoring running processes on a client's device. It can identify suspicious programs, including cheats, decompilers, and malware, by comparing them with known signatures stored in threat

Table 1
Offensive and defensive techniques across domains

Technique	Hacking/Cybersecurity	Cheating/Anti-Cheat
Code injection	Injecting code into another process to alter its behavior or evade detection	Injecting code, such as DLLs, into games to modify behavior for in-game advantages
Reverse engineering	Deconstructing software to discover vulnerabilities or ways to bypass defense mechanisms, or to analyze malware	Deconstructing game code or client-side AC to discover vulnerabilities, or analyzing cheats to detect them
Code obfuscation	Modifying software code to complicate RE while preserving functionality	Transforming or encrypting game or AC code to complicate RE
Process scanning	Scanning the system or memory for malicious software or tools to detect and prevent attacks	Scanning the memory or system of a client device for suspicious programs or cheats to prevent attacks

databases [11, 23].

Table 1 summarizes how the techniques discussed in this subsection are used in the cybersecurity and gaming domains.

2.3. Cross-Industry Gap in Defensive Approaches

Despite the technical overlap between attacks, defensive approaches remain mostly isolated in both industries. Thomas and Francis [24] observe that current AC research focuses primarily on reactive threat mitigation, detecting attacks only after they have occurred. They therefore recommend strengthening proactive defenses, i.e., detecting or preventing attacks before they take effect. Although kernel-level AC systems can support such proactive protection, concerns regarding performance and privacy may limit their adoption [11, 13]. Pontiroli [10] compares cheating techniques to several malware families and argues that cybersecurity research is therefore highly relevant to the development of modern AC systems.

To the best of our knowledge, little research has examined how defensive knowledge can be transferred between cybersecurity and gaming. Such transfer can take two forms: sharing threat intelligence, such as indicators of known cheats, malware, or vulnerable drivers, and adapting defensive mechanisms, such as the proactive detection employed by AV and EDR systems [16]. Whether either occurs depends not only on technical compatibility but also on the organizations involved, consistent with socio-technical approaches that consider technical systems together with the people and organizational structures in which they operate [25].

2.4. Related work

Studies comparing game security with the broader cybersecurity domain remain limited, and, to the best of our knowledge, no work has systematically compared the underlying offensive techniques. However, related areas, such as the social and motivational factors that parallel game cheating and cybercrime, have been discussed in prior research [7, 8]. Cho et al. [7] use the criminological Routine Activity Theory to analyze similar behaviors and motivations among cheaters and cybercriminals. In their study, they discovered that cheaters often develop deep technical knowledge of programming and library development on cheating-related forums. Furthermore, they note that cheaters not involved in cybercrime often come into contact with cybercriminals through these forums, which they suggest could lead to familiarity with such practices without a proper understanding of their implications. Similarly, Bada and Chua [8] argue that young people might come into contact with hacking tools to use against games while not understanding the legal boundaries of their actions. These studies thus

portray cheating forums as social settings in which technical knowledge is shared and cheaters may encounter cybercriminal practices.

Many techniques used in game cheating were mentioned by Pontiroli as sharing malware-like abilities, for example, the use of function hooking to modify software capabilities or CI to alter process behavior [10]. However, while certain similarities were identified, the extent to which the techniques overlapped across the cheating and hacking domains was not thoroughly discussed.

Without research directly comparing the technical overlap between cheating and hacking, the extent to which knowledge and defensive practices can be transferred between the two domains remains unclear. Understanding these overlaps may help explain how technical knowledge developed in cheating communities could transfer to broader cybersecurity contexts.

Ibrahim [1] discusses trends in cheat development and highlights threats such as the use of Artificial Intelligence (AI) tools, ransomware, attacks targeting IoT devices, and phishing using deepfake technology. These trends mirror developments observed in cybersecurity research. For example, Zaid and Garai [26] describe how hackers use newer technologies, such as AI, to carry out attacks, including ransomware, phishing, or exploitation of vulnerabilities in IoT devices. The similarity of these emerging threats further highlights opportunities for cross-industry learning to strengthen defensive practices in both domains.

3. Method

This research focuses primarily on understanding the similarities and differences between how AC developers and cybersecurity experts operate and respond to similar threats. Therefore, the study adopted a comparative research approach to analyzing the two fields. This comparison was done qualitatively using semi-structured interviews for data collection and thematic analysis for data analysis. The way these two methodologies were applied for this research is described in detail in the following subsections.

3.1. Data Collection

To gain a detailed understanding of potential overlap in techniques across domains, semi-structured interviews were chosen as the data collection method. Semi-structured interviews provided a flexible yet common structure to follow across interviews, ensuring consistency among respondents [27]. Additionally, given the flexibility, further knowledge could be captured by asking follow-up questions to the respondents' answers.

Respondents were purposively sampled for their practical experience with offensive or defensive techniques in their domain: five with AC expertise and six with cybersecurity expertise. After approximately three interviews per group, few new codes emerged and later interviews mainly confirmed existing themes, suggesting code saturation [28]. Saturation was thus observed during data collection rather than used as a stopping criterion: the remaining scheduled interviews were completed, and one additional cybersecurity respondent was recruited to increase the technical depth of that group (Section 5.3). Given the limited research on how both domains defend against these threats, expert insights were valuable for understanding their similarities and differences.

The small sample may have limited the range of perspectives captured. Years of professional experience were not recorded systematically and are therefore not reported.

3.1.1. Interview guide

For this research, interview guides were used during each interview to ensure consistency and structure among respondents [27]. Furthermore, using interview guides helps outline opportunities to ask follow-up questions to go into greater depth and collect additional data relevant to the RQs. The questions in the interview guide were asked of all participants, but given the semi-structured approach, additional questions beyond it could be added based on respondents' answers.

Existing research, some of which is presented in the Background section, helped with the development of the interview guide. The guide featured questions about RQ1, including how RE and CI are applied and mitigated in both contexts, providing insights into potential overlaps between the techniques. Similarly, RQ2 was addressed through questions examining both industries' current defensive practices, including how they operate, their strengths and weaknesses, and respondents' perceived opportunities for collaboration between the fields.

Examples of core questions included in the guide are how CI and RE are used in each industry and what defensive mechanisms are employed to mitigate malicious attempts of compromise. Questions about the benefits and drawbacks of client and server-side security implementations are also included. These questions aim to determine which approach is most feasible from a performance and security standpoint, and whether the two industries have different perspectives. Finally, questions about opportunities and barriers to collaboration are included to assess whether industry professionals believe collaboration is feasible.

Since both gaming security and regular cybersecurity operations differ, questions within the interview guide could not be adapted for both industries simultaneously. Therefore, two interview guides were created, featuring similar questions that encompass the same core themes but adapted to their respective operational contexts. Using different interview guides for both industries could have improved the validity of the research, since it is unlikely that cybersecurity experts have detailed knowledge of game-specific defenses or threats, and vice versa. Not factoring this in could have hurt the data collection process, as the answers may be of low quality because they are based on speculation. Adapting the questions for each industry ensured that the collected data was relevant and grounded in expert insights, enabling accurate and detailed comparisons.

Before the interview guide was applied in practice, an industry professional helped assess whether the questions were feasible and could elicit the information needed to answer the research questions. In addition, a pilot interview was also conducted with a willing peer to gauge whether the questions were clear and understandable. The pilot interview also analyzed whether the included questions posed potential bias, and the questions were then altered based on the feedback. The participant in this pilot had some background in the topics included in the guide, making their feedback valuable. However, because they were not an expert in the field, it was clarified that their answers would not be included in the analyzed dataset.

The findings of the pilot interview revealed a risk of confirmation bias, as some core and follow-up questions tried to confirm previous assumptions about overlaps between the domains. To address this, some questions in the interview guide were reworked to allow more flexible responses rather than leading questions [29]. This allowed themes to emerge more naturally rather than forcing them based on preconceptions, which could have affected the credibility of the results.

The interview guides used for this study are presented in Appendix A for AC respondents and Appendix B for cybersecurity respondents.

3.2. Data analysis

To analyze the data collected from the semi-structured interviews, a thematic analysis was conducted using an abductive approach to theming. Clarke and Braun [30] describe thematic analysis as a flexible way to identify and interpret patterns within qualitative data. These patterns can then be applied across different themes to understand the underlying meanings of the empirical data. An abductive approach was chosen as a middle ground between inductive and deductive approaches [31]: initial analytical concepts were informed by prior research, such as the techniques in Table 1, while new codes could emerge from respondents' answers. Applying the same initial concepts to both groups enabled a systematic comparison of the domains [32].

The first two authors coded all transcripts independently and then compared their codes; differences were discussed until consensus was reached [33]. This investigator triangulation allowed the same data to be examined from different perspectives and reduced the risk of individual bias in theme development [34].

3.3. Ethical considerations

According to Pontiroli [10], some AC systems may employ a security-by-obscurity approach to defend against cheaters. Because of this, all company and game names were kept anonymous, so this study could not be used to assist with cheat development. Information regarding anonymity and how the collected data would be handled was provided to the participants both in the interview invitation and during the interview to ensure they understood that none of their answers could be linked to them. Research in other domains has shown that respondents may be more willing to discuss and disclose detailed and potentially sensitive information if they remain confidential [35]. The respondents in this study may therefore have been more willing to discuss challenges and tools used to defend against different cyberattacks, as this area can involve confidential or sensitive aspects. This could have provided more relevant and detailed results, as respondents may not feel the need to withhold certain information about their organization's operations.

All interviews were audio-recorded to improve transcript accuracy and enhance the reliability of the study [36]. Because recording may influence respondents' answers (the observer's paradox) [37], respondents were offered manual note-taking instead; none chose this option. After transcription, all recordings were erased to prevent linking any data back to the respondents.

In the interview invitation, information about the study, the data planned to be collected, and how it would be handled was clearly stated. If something was unclear, the option to ask further questions before agreement was available. This allowed them to make informed decisions about whether they wanted to participate in the research. Written consent to conduct the interview was also required to ensure that the participant understood what the study was about and their role within it. The interview invitation used for this study is presented in Appendix C.

To improve the reliability and validity of the study, the recommendations of Good Research Practice 2024 [38] were followed in data collection and analysis. Since all interviews were conducted within the EU, the study also complies with the guidelines set out in the General Data Protection Regulation (GDPR) on the processing of personal data, ensuring transparency, confidentiality, and integrity. No data other than what was necessary for the research was collected, and once data, such as recordings and identifiers for transcriptions, had been used, they were deleted. Doing this ensured that no information could be traced back to the original respondents. Respondents were also allowed to avoid answering certain questions and also had the right to withdraw from the interview at any time.

4. Results

This section presents the results of the interviews with cybersecurity and AC experts.

4.1. Participant overview

Using purposive sampling via LinkedIn and contacts provided by the study's supervisor, eleven individuals participated in the interviews. This resulted in a distribution of six Cybersecurity Respondents (CSR) and five Anti-Cheat Respondents (ACR). In Table 2 and Table 3, contextual information about the interviewees is presented. The ACR and CSR labels denote the expertise for which respondents were sampled, not necessarily their current role. All CSRs worked in cybersecurity at the time of the interviews, whereas ACR1 and ACR3 were sampled for their background in cheating and AC and now work in other security roles. Their experience of both domains made them well placed to compare the two.

The sample covers operational (CSR1, CSR6, ACR5), offensive (CSR2, ACR1, ACR3), development (ACR2, ACR4), and academic (CSR3–CSR5) roles, and thus captures both technical practice and the organizational settings in which it takes place.

Table 2
Contextual information - Cybersecurity respondents

Respondent	Current role / relevant background
CSR1	L2 SOC Analyst
CSR2	Penetration Tester
CSR3	Cybersecurity Lecturer
CSR4	Cybersecurity Researcher
CSR5	Cybersecurity Researcher
CSR6	Development and Blue Team Activity

Table 3
Contextual information - Anti-Cheat respondents

Respondent	Current role / relevant background
ACR1	Smartphone Hacker
ACR2	Anti-Cheat Developer
ACR3	Vulnerability Researcher
ACR4	Anti-Cheat Developer
ACR5	IT Security (AC Team)

4.2. Thematic findings

The analysis produced two themes, corresponding to RQ1 and RQ2, each comprising three sub-themes. The RQ1 sub-themes concern the techniques themselves, whereas the RQ2 sub-themes also address the organizational and human conditions for sharing knowledge. Appendix D shows an example of the coding tree.

4.2.1. Overlap of code injection and reverse engineering across industries

This subsection presents the sub-themes on the overlap, or lack thereof, between attacks in gaming and cybersecurity contexts (RQ1). Figures 1 and 2 summarize the main overlaps and differences.

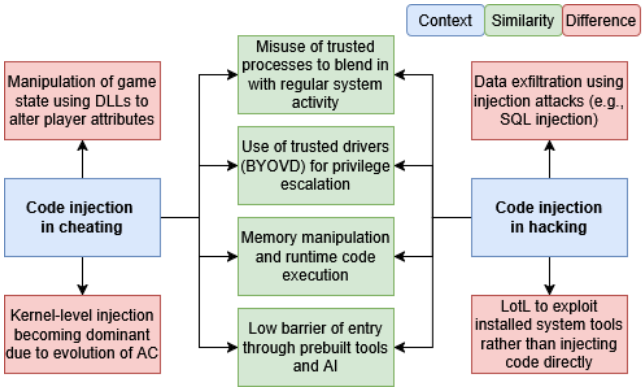


Figure 1: Code injection, similarities and differences

Prevalence and accessibility of techniques. Both domains described CI as widely accessible and as having a low barrier to entry for conducting malicious activities. Multiple respondents also stated that several prebuilt tools and documentation were available, enabling many effective attacks to be carried out with little technical expertise. Among ACRs, DLL injection was often described as the most commonly applied cheating technique. ACR1 described DLL injections as extremely powerful in terms of the advantages they provide, but also relatively easy to execute, depending on the defensive measures in place.

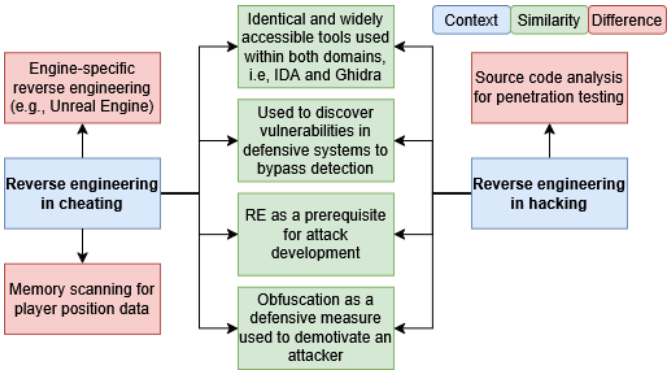


Figure 2: Reverse engineering, similarities and differences

Although some attacks could be simple, the difficulty of CI was described as varying with the desired effectiveness. According to most ACRs, injectors for different games could be publicly found online, making such attacks simple to execute. ACR2 even described that many cheaters often fully rely on code found on forums that does not require a deep technical understanding. ACR3 reinforced this by stating that these types of attack were the most commonly encountered by AC systems and therefore described them as ”low-hanging fruit” for cheaters.

> *“They find whatever code [...] compile it and it just works, or they gather information that someone else got from reverse engineering the game and cobble together a half-working cheat.”* - ACR2

These accounts indicate that accessibility is social as well as technical: forums, shared code, and documentation distribute the knowledge needed to use the tools.

Other ACRs also discussed that these attacks could be performed manually using more complicated methods, such as manual mapping of DLLs within a process memory, bypassing the use of Windows API calls when injecting DLLs. Respondents described manual mapping as providing greater control and stealth of the attack, but would often require more technical expertise and knowledge of the target system. Manual mapping was especially useful against modern ACs, according to ACR2, as these ACs are designed to detect and prevent injection attacks that rely on API calls, which prebuilt tools often use.

> *”[...] they’re not gonna be using Windows hooks or LoadLibrary or whatever because all those are trivially blocked or detected. So the only option left is really to manually map their code into a random portion of memory.”* - ACR2

The accessibility of attacks was also discussed in cybersecurity contexts. CSRs described process injection as relatively rare compared to gaming, whereas some highlighted web-application injection, such as SQL injection, and techniques that are not injection, such as Living-off-the-Land (LotL) and malicious LNK files, as relatively common. Similar to AC, multiple CSRs also raised concerns about the use of prebuilt tools to perform these attacks. CSR2, a penetration tester, noted that using such tools allowed them to easily perform CI-like techniques without prior malware development experience.

CSR2 also noted that while these prebuilt techniques are easy to use, they are often detectable, and as a result, more knowledgeable attackers use more advanced and stealthier alternatives. In contrast, CSR5 further noted that with the emergence of generative AI, the barrier to entry has been lowered, enabling the development of advanced techniques or exploits using simple queries.

> *“Even people who did not have the skills and knowledge by training can now use AI to get quite far with attacks.”* - CSR5

A pattern similar to CI was also observed with RE, where tools for decompilation and debugging, such as Ghidra and IDA, are widely available and useful in both domains for discovering software vulnerabilities. ACR3 described that these tools, while useful for cybersecurity and AC to analyze malware and cheats, are equally accessible to malicious actors who use them for malicious purposes. However, despite the tools themselves being widely available, the respondents described RE as a complicated process, especially against modern software with up-to-date defenses.

Misuse of trusted processes to conduct malicious activity. Another discovered theme was the use of trusted processes or system functions to conduct malicious attacks. In AC contexts, this was primarily identified through DLL injection attacks, in which attackers develop and inject malicious DLLs into a trusted game process. This allows them to directly call internal game functions and modify pages in memory, resulting in attackers gaining extensive control over the process through Windows APIs or more advanced methods such as manual mapping.

All ACRs discussed Bring Your Own Vulnerable Driver (BYOVD), a technique in which attackers use trusted but vulnerable Windows drivers to operate with elevated access, often going undetected by defenses. ACR1 explicitly highlighted, from a cybersecurity perspective, that BYOVD was ”literally one-to-one with what the cheat developers do”, directly linking it to malware tactics used by hackers. In addition, ACR4 noted that as ACs begin moving to kernel-level, using drivers for exploitation becomes increasingly important, as user-mode techniques, such as calling or hooking certain Windows APIs, are often detected and blocked. Exploiting these vulnerable drivers for malicious functionality was therefore useful in both CI and RE contexts.

In cybersecurity, respondents described the misuse of trusted processes mainly through LotL attacks, which use already installed system tools, such as cmd and PowerShell, instead of attacker-developed software. When applying these trusted tools and vulnerable drivers for malicious activity, CSR1 explained that attacks could be carried out while blending in with regular system operations.

> *“They can be used in a way that makes you think that this is intentional or it’s safe because it’s a trusted process that’s being used”* - CSR1

Across both domains, respondents reported that using trusted software and tools often increases stealth, allowing attacks to bypass detection measures that may accept these malicious actions as legitimate behavior. Respondents therefore highlighted a major challenge in differentiating between legitimate and malicious activity, as many programs used in both domains, such as OBS and Discord, may utilize hooks commonly associated with malicious behaviors. CSR6 provided an example in which an employee installed a game with a kernel-level AC on a work computer. The organization’s endpoint protection blocked the AC because it behaved like a potential threat, raising an alert in their Security Information and Event Management (SIEM) system.

> *“Our endpoint protection has detected games and some anti-cheat or other solutions as malicious, usually because they live in kernel space with greater privileges than they should.”* - CSR6

Reverse engineering for discovering vulnerabilities. In both domains, RE was identified as an important technique for discovering software vulnerabilities and understanding their functionality. As noted by ACR3 and CSR4, the same widely accessible RE tools were available for both defensive and malicious purposes in both contexts, resulting in a direct overlap of tools across domains.

In cheating contexts, ACR1 and ACR2 noted that vulnerabilities could be discovered using a multitude of methods with RE. For example, the easiest approach would be to locate patterns or strings within binary code, which would provide context about program functionality. Additionally, discovering leaked source code was also highly beneficial, as all the context about the software was readily available. If these approaches were not feasible, as they were described as rare, ACR1 highlighted more sophisticated methods, such as identifying common patterns in the game engine or scanning the process memory for value changes.

"You'll have to see strings in the binaries [...] if you have an Unreal Engine game, first of all, you are reverse engineering the engine itself, and if you know the version of the engine, there are tools that you can use to get a lot of information from it." - ACR1

In cybersecurity red-team contexts, CSR2 described the use of RE during pentests, in which available source code was analyzed to reverse-engineer a custom hash function by identifying its vulnerabilities. By reversing the function, they were able to access plain-text passwords, essentially bypassing authentication for all accounts. This therefore shares a direct method overlap between the fields, as discovering program functionality and logic was described as essential for identifying vulnerabilities that could be exploited.

In addition to targeting the game or software, respondents also described that attackers focus on conducting RE against the defenses themselves. ACR3 noted that doing RE against AC software could expose security gaps or other vulnerabilities that could be exploited to bypass detection. Similarly, ACR2 described their experience conducting RE against popular AV systems to understand their inner workings, highlighting an overlap in how RE is applied across cheating and hacking contexts. Additionally, as modern defenses begin to operate on client devices, it becomes increasingly difficult to prevent RE, and ACR1 explained that "as long as it's running on your device, there will be a way to bypass it".

Both industries also discussed comparable defensive measures against RE, primarily code obfuscation. Obfuscation was described not as preventing RE completely, but as a way to introduce resistance to RE. This resistance is used to demotivate attackers by increasing the complexity of a game's code and logic, making RE less feasible. As RE is often highly technical, obfuscation can prevent less experienced attackers from reverse-engineering the software.

"It's not a cryptographically secure method of stopping people from interfering with your software. The goal is to demotivate the attacker as much as possible, so they give up, and they move on to another game" - ACR3

ACR4 and ACR5 also described the use of obfuscation, but instead within game engines. They noted that if a game using a widely known engine is reverse-engineered, it may become problematic for other developers, as many of the tools used would be directly transferable to games sharing the same engine. Therefore, continuously modifying and customizing the engine between patches was an effective way to deter this, as it increased the effort needed to develop the appropriate tools.

ACR1 observed a popular game that significantly modified and changed the inner workings of a known engine, making player variables inaccessible to attackers. With this approach, they noted that offensive and defensive techniques began to blend, as both use similar methods to manipulate the game toward opposite goals.

"Think of [game] AC manipulating the game in a similar manner that cheat developers would do, then the technique sort of starts blending." - ACR1

4.2.2. Perceptions of cross-industry collaboration

This subsection presents the sub-themes on barriers and opportunities for cross-industry collaboration (RQ2), summarized in Figure 3.

Barriers to collaboration. Several structural barriers complicate the opportunities for collaboration between AC and cybersecurity industries. One of the recurring concerns that was raised by the respondents is the risk of over-disclosure. Sharing the technical mechanisms underlying a defense system could potentially undermine it. As one respondent noted, if an adversary "knew these five things" considerable effort could be rendered useless in an instant.

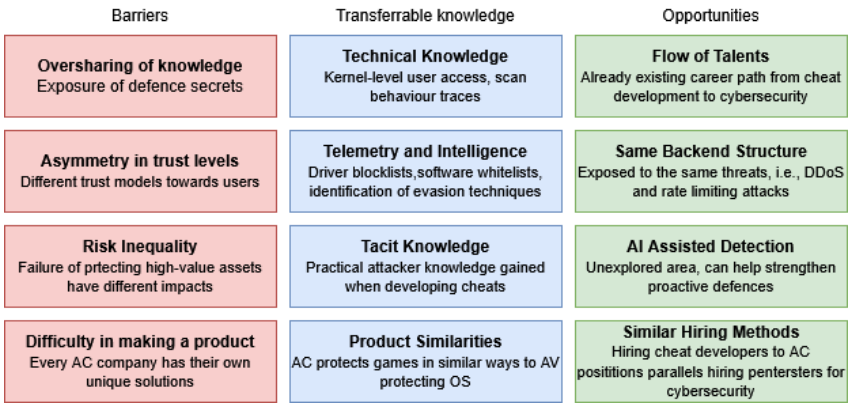


Figure 3: Overview of barriers and opportunities for cross-industry collaboration

> *"[...] let's say everyone collaborates, but what happens if one person uses this information in malicious ways and things like that? So, there's a hesitance in the industry to overshare about all your secret sauce,..."* - ACR4

As ACR1 and ACR4 suggested, many people can return to the cheating scene. Over-disclosure of defense mechanisms could lead to insider attacks, making it easier for cheat developers to find workarounds. And as ACR4 suggested, the industry is reluctant to share this knowledge because building these defenses can take considerable time and money.

Another barrier is that while the high-level techniques might be comparable, the implementation and details are uniquely implemented for the product, and "productifying" them for broader use was described as challenging. This limits the practicality of transferable knowledge, even if techniques have several overlaps.

> *"So, there are a lot of things that are really custom to the company you work at and not so easy to like productify [sic] and like use in a broader context in that sense"* - ACR4

The industries are also divided by trust models and perceived stakes. ACR2 emphasized that EDR and AV software are designed to protect users from malicious attacks, while AC software defends the product against manipulation by its users. CSR1 observed an asymmetry in perceived stakes within both domains. Failures in cybersecurity can have societal consequences; failures in AC can cause financial and reputational damage to the game company, but are not regarded as carrying comparable weight.

Opportunities for collaboration. Despite these barriers, the respondents identified several areas where cross-industry collaboration could be productive. Perhaps the most prominent is through talent pipelines. An informal path already exists from cheat development into AC and wider cybersecurity roles, which ACR1 and ACR3 themselves followed; ACR3 moved via the AC industry into cybersecurity.

> *"I think definitely, because I mean, look at me as an example, for I went from the anti-cheat industry into the cybersecurity industry. So there's a career pathway there..."* - ACR3

As ACR1 emphasized, some AC companies have already begun hiring cheat developers. This form of collaboration mirrors how penetration testers are recruited in cybersecurity contexts. ACR5 also noted that they have run bug bounty programs for their games.

A second area for cross-industry collaboration is strengthening the back-end structure of AC organizations. ACR4 and ACR5 discussed that game servers face many of the same threats as conventional IT systems, such as DDoS attacks and rate-limiting abuse. ACR4 noted that once the game client communicates with its back end, "it becomes more like a classic IT problem." ACR5 also discussed that

many game companies lack in-house IT security expertise and could benefit from working with security firms.

AI-assisted detection was also raised as an under-explored area. CSR4 drew a parallel between AI's emerging capability to detect zero-day attacks and the potential for similar systems to flag cheating behaviors. As with monitoring in traditional cybersecurity, an automated match review system, in which AI analyses recorded gameplay for statistically impossible patterns, was suggested as a complementary approach. Though respondents also acknowledged the risk of false positives, noting that AI systems have misidentified benign objects in certain contexts. The potential for AI-assisted detection, therefore, appears real, but respondents indicated it remains largely uninvestigated in academic and industry literature.

Transferable knowledge. The technical overlap between the two industries is substantial, even if it has not yet been formally leveraged. Both EDR systems and AC software operate at the kernel level, set similar callbacks, monitor thread and process creation, and strip handle access to protect their own processes from being terminated by hostile software (ACR2, ACR5). As ACR2 put it, they "pretty much use the same kind of tech", the difference is not what they do, but who they are protecting against. This similarity extends to products such as Easy Anti-Cheat and Denuvo, which ACR5 described as being similar to Microsoft Defender because they all scan the systems they protect for anomalous activity, differing mainly in the threats they are designed to detect.

Telemetry and threat intelligence collected by AC systems could also be shared across domains. AC developers maintain a detailed knowledge of driver vulnerabilities, CI techniques, and evasion methods, all of which are directly relevant to malware defense. Driver blocklists, for instance, would inform AV providers, who have the advantage of loading at system boot and could therefore prevent vulnerable drivers from initializing before the game and the cheat ever runs (ACR2). This exchange also has the possibility of working in both directions.

> *"And of course the antivirus, basically all of them load at system boot up so they can pretty much stop that driver from loading as soon as possible, because not all anti-cheats run at system boot like Easy Anti-Cheat and BattlEye."* - ACR2

Beyond tools and data, the knowledge produced through cheat and AC development presents a significant form of transferable expertise. As ACR3 stated, individuals who have worked in this area since childhood often have a deep, practical, hands-on understanding of Windows internals that formal education rarely produces. The knowledge of how systems can be manipulated, where defenses break down, and how attackers think is insight that is difficult to teach in conventional training environments, and that several respondents had already carried into professional cybersecurity roles.

5. Discussion

This section discusses the analyzed data in relation to this study's RQs and existing research in the field. The aim of this study was to explore overlaps between hacking and game-cheating techniques and to identify opportunities for cross-domain collaboration to strengthen defenses. The findings of the interviews reveal overlaps, barriers, and potential for collaboration. These findings will be further discussed in the following subsections, which are structured around the main themes presented in the results.

5.1. Overlap of code injection and reverse engineering across industries

Multiple overlapping themes in both CI and RE were identified from responses to the 11 interviewees. The main findings across both domains were that the techniques were easily accessible but varied in difficulty depending on the attacker's goal, often relied on misusing trusted processes, and were used to discover and exploit software vulnerabilities. While previous research has mainly focused on attacker

motivations and communities in cheating and hacking contexts [7, 8], this study instead examines the overlap and transferability of the techniques used by attackers across domains.

The ease of access to both CI and RE was a common theme among respondents. It was also stated that the difficulty of CI could vary greatly depending on the required effectiveness; however, for entry-level attacks, publicly available documentation and tools were often used. Bada and Chua noted a similar finding in their research, and described hacking forums as a popular source to share and discover knowledge and tools for both cheating and hacking activities [8]. Additionally, they described a difference between public and private tools on the forums, which aligns with findings from this study, in which techniques accessible to the public were described as less effective than those requiring greater technical experience.

Despite publicly available tools being less effective, the emergence of AI tools for cheating and hacking [1, 26] could enable attackers to develop advanced attacks using queries, reducing the required technical knowledge, according to respondents. Therefore, these findings suggest that as barriers to entry decrease, the number of low-skilled but effective attacks may increase in both fields. Defenses relying on signatures of known tools may then become less effective, strengthening the case for proactive, behavior-based detection, as also argued by Thomas and Francis [24]. Additionally, given the accessibility to these techniques, knowledge transfer across cheating and hacking could be plausible, as limited technical knowledge is required for entry-level attacks.

Although CI techniques were used in both domains, a major contrast in their application frequency was observed. Several ACRs stated that CI, specifically DLL injection, is commonly used, a point previously acknowledged in the literature [3, 4, 17]. In contrast, CSRs described process injection as rare and gave alternatives such as LotL techniques. This appears to contradict previous research describing CI as among the most widely used hacking methods [18]. However, that research concerns web-application injection, such as SQL injection, which some CSRs also described as common; the contrast therefore applies to process injection only.

This difference could depend on multiple factors; for example, since cybersecurity is a broad field with many varying threats, some respondents might not often encounter these attacks in practice, depending on their role. However, the AC industry is quite contained, allowing greater exposure to techniques such as CI, which are considered dominant. It could also have been the result of improvements in current defensive systems. For example, multiple respondents indicated that direct CI attacks have become increasingly detectable, making them less viable for attackers who require stealth and persistence. In addition, respondents noted that publicly available tools commonly used for CI are often easier to detect, possibly due to improved signature scanning, as these attack signatures are stored in threat databases [15]. This suggests that more advanced attackers instead move away from these approaches to other alternatives. This trend was also highlighted by several ACRs, who stated that as kernel-level defenses become more common, CI must become much more advanced, since relying on API calls or other user-mode methods for code injection will be detected. This results in a technical barrier, in which attackers must operate with Windows kernel-level access to evade defenses and achieve their objectives [3].

Because user-mode attacks are easily detected, many attacks instead focus on using trusted processes and Windows functions to blend in with system activity, improving stealth and reducing reliance on API calls. The use of trusted processes was another common overlap discovered between the fields. One example of this overlap is the BYOVD technique, in which attackers exploit vulnerable but trusted drivers to gain elevated privileges. Drivers often operate at the kernel level, and if they are signed by Microsoft, they are authorized and trusted on Windows machines [11]. Therefore, if a vulnerability is discovered in a driver, attackers could exploit it to bypass security and achieve their objectives, such as manually loading malicious code, which the program then executes [3]. BYOVD can thus be seen as a software supply-chain threat in which the trust granted to signed third-party components is abused; such threats have organizational as well as technical dimensions [39]. One participant even stated that BYOVD techniques were used in the exact same way within both cheating and hacking, suggesting a direct overlap between the fields, as the same vulnerable drivers used for cheating could also be applied in hacking contexts.

In addition to CI techniques sharing multiple similarities, RE was also found to have significant overlap. As noted by the respondents in both domains, the same tools were applied for both offensive and defensive purposes (i.e., Ghidra and IDA). Offensively, it could be used to discover software vulnerabilities; defensively, to analyze malware. For offensive purposes, respondents noted that analyzing a process's functionality and logic was essential to discovering vulnerabilities, a point Kanervisto et al. [22] also made. It is therefore incredibly important to protect against RE as it is often a requirement before attacks can be developed. Both AC and cybersecurity defenses employed similar measures to hinder RE, namely code obfuscation. However, both respondents and prior research indicate that obfuscation does not prevent attacks; rather, it increases complexity [3]. This approach would therefore filter out lower-skilled attackers, but might not prevent those who are highly advanced and dedicated to reverse-engineering a system.

Because both industries apply similar defensive measures and use the same offensive tools, this suggests that highly skilled reverse engineers could transfer some of their knowledge and skills across domains for malicious purposes. However, while technical overlaps, such as scanning process memory and discovering ways to bypass security systems, could be applied in both fields, transferability is not guaranteed. Cho et al. [7] note in their study that although cheating and hacking share parallels, there are numerous differences as well. For example, it is noted that cheaters often commit attacks out of intrinsic motivation, whereas cybercriminals are often driven by extrinsic rewards such as money. Since the techniques could also be used in defensive contexts, Cho et al. [7] also argue that cheaters could transfer skills to cybersecurity instead. Findings from this study also support this, as ACR1 and ACR3 were direct examples of cheaters transitioning to the cybersecurity field, and the knowledge they acquired from cheating communities was transferable to broader security operations.

5.2. Perceptions of cross-industry collaboration

The findings of this study indicate that although the AC and cybersecurity industries share certain technical foundations, cross-industry collaboration largely lags behind due to structural and cultural barriers. This situation coincides with a gap in the literature: defense approaches in both fields were developed in isolation despite the technical similarity of the threats they faced [10, 24].

One prominent barrier mentioned by participants is the risk of sharing too much information. Sharing detailed information about existing defense mechanisms might reveal ways for malicious actors to bypass them. This situation reflects the differences between transparency and operational security in security contexts and is also discussed in the literature as security through obscurity, which is often employed in AC systems [10]. Another related barrier is asymmetric trust models: AV and EDR protect the device against malicious software and unauthorized activity, whereas AC protects the game against manipulation by the device's own user. As ACR2 noted, both use largely the same technology; the main difference thus lies not in the technology but in the trust relationship in which it is deployed. This is consistent with viewing information systems as human activity systems, in which the same technology can serve different purposes depending on the organizational context it supports [40]. The case reported by CSR6 shows both models on one device: kernel access that the game vendor requires to protect its game is treated as a potential threat under the employer's security policy. This difference in threat orientation leads to a disparity in perceived risks: while failures in cybersecurity are evaluated as having societal consequences, failures in AC are not thought to carry similar weight. This different priority structure may make it difficult to sustain close technical collaboration and meaningful information exchange, even when there is technical overlap.

Despite these barriers, the findings also reveal opportunities for collaboration. The most fundamental of these opportunities is the movement of talent between the two industries. Participants noted that individuals with experience developing cheats sometimes transition into AC and broader cybersecurity roles. This informal pipeline transfers knowledge through people rather than technical artifacts, carrying expertise from an informal, adversarial community into formal defensive organizations. The people required for cross-industry collaboration thus already exist, but the pathway has not yet been institutionalized. A second area of potential collaboration is strengthening back-end infrastructure.

As noted by ACR4 and ACR5, game servers face the same threats as traditional IT systems, including DDoS attacks and rate-limit abuse. This means that existing cybersecurity expertise is likely to be directly applicable in these contexts. This is especially important, as some gaming companies lack in-house IT security expertise, according to respondents. Collaboration here would thus be primarily organizational, e.g., through partnerships with security firms, as the required expertise already exists.

AI-assisted detection was also described as a relatively unexplored area in AC contexts. Participants suggested that AI techniques used in cybersecurity to identify previously unseen or anomalous attacks could also be adapted to detect statistically improbable gameplay behavior. This approach aligns with Thomas and Francis' proposal to shift from reactive threat mitigation toward proactive defense [24]. However, participants also highlighted the risk of false positives, noting that AI-based detection systems can misidentify benign activities in certain contexts. In AC, false positives also have a human cost: legitimate players may be wrongly flagged or banned, which can undermine their trust in automated enforcement.

Another finding related to collaboration was the ability to transfer various types of telemetry data on vulnerable drivers between the two industries. ACR1 explicitly described BYOVD attacks as being conducted in the same way in both cheating and hacking contexts. This suggests that sharing information about these drivers could be a valuable and effective way to strengthen proactive detection and prevention. This approach would be bidirectional, meaning blocklists could be shared between domains to gain a more holistic overview of vulnerabilities that could be exploited in trusted drivers.

Since AV systems load at system boot, whereas AC often loads alongside the game itself, strengthening detection of vulnerable drivers through this intelligence sharing could be beneficial. This is because these drivers could be blocked before many ACs can, which could help mitigate their exploitation for malicious purposes. Consequently, this approach would be effective in cases where attackers gain control of a client system, as they would be limited by restricted access to kernel-level drivers, hindering their ability to bypass detection and achieve their objectives.

In addition to sharing driver blocklists, sharing information about trusted software would also be valuable. Respondents described difficulty distinguishing between trusted and malicious software because they commonly use similar functions, such as function hooking or other malware-like capabilities (e.g., rootkit-like behaviors) [10]. As a result, trusted software might generate excessive false positives, potentially overwhelming analysts [9] and increasing the risk of alert fatigue. Therefore, being able to share up-to-date information on software would enhance the ability to differentiate benign from malicious activity, thereby reducing false positives.

Another finding of this study was the opportunity for knowledge transfer. Individuals with experience in cheat development were described as having developed a great hands-on understanding of how systems can be manipulated, where defenses fail, and how attackers think. Cho et al. [7] highlighted that some individuals acquire all their programming knowledge through cheating forums, making such communities valuable for developing specialized technical skills. Cheating communities can thus act as informal, practice-based learning environments that develop low-level systems expertise less commonly emphasized in formal education, which individuals can carry into cybersecurity and AC roles.

The transfer of knowledge about cheating was also identified as advantageous. A respondent highlighted a finding unique to gaming: some developers alter game engines and configure ACs in ways that instead blend offensive techniques for defensive purposes. This included changing the inner workings of an engine to hide or block access to the synchronization of player variables, making reverse engineering and the development of cheats much more convoluted tasks. AC developers, in these cases, therefore have to think like a cheater to understand what the cheat developers aim to target, and then alter the game to prevent access to it. Furthermore, Dorner and Klausner [13] described ACs as sharing similarities with offensive techniques, noting that some popular kernel-level ACs exhibit traces of rootkit-like behavior. They explain that rootkits operate similarly to some AC solutions, using evasion techniques, information exfiltration, and virtualization to prevent RE, evade analysis, and exfiltrate data from a system.

These findings indicate that it would be advantageous for cheaters or others with knowledge and experience of offensive techniques in either domain to transition into security roles. Some companies

have already begun hiring individuals with a background in cheating, which further suggests that this approach, along with knowledge transfer, would be highly valuable and feasible to strengthen defenses. Prior attackers would likely understand how threat actors operate and what they typically target on a system, thereby allowing those attacks to be countered accordingly.

5.3. Method discussion

For this study, semi-structured interviews were used as the data collection method, which suited a niche research area with limited academic literature on the application of offensive and defensive techniques. However, the two groups differed in their direct exposure to the examined techniques: all ACRs discussed CI and RE in technical depth, whereas most CSRs encountered them less directly in their roles. This asymmetry may reflect both the sampled roles and differences between the domains. As a result, ACRs provided richer technical data. To reduce the imbalance, an additional CSR was interviewed, and some ACRs with cybersecurity experience could also speak to that domain. Future studies could include experts in application security or vulnerability research.

The small sample may have limited the generalizability of the findings, and patterns held by individual respondents may have been missed. Recruitment was constrained by the confidentiality of both fields: some contacted professionals were not permitted by their management to participate, despite pseudonymization, owing to the risk of over-disclosure. This gatekeeping mirrors the reluctance to share defensive knowledge identified as a barrier to collaboration and may have narrowed the perspectives represented. Participants may likewise have withheld sensitive information, although pseudonymization and the option to skip questions were intended to reduce this.

6. Conclusions

This section begins with a summary of the study's findings in relation to the two RQs and outlines both the practical and scientific implications of the results. In addition, this section will provide recommendations for organizations and future research to further expand the topics explored.

The purpose of this research was to identify how common offensive techniques used in both cheating and hacking may overlap, and to understand whether knowledge in one area could be transferred to the other. Additionally, it aimed to determine whether cross-domain collaboration could be feasible for strengthening the detection and prevention of these attacks. To achieve this, two RQs were answered, which are summarized below:

RQ1 examined overlaps between CI and RE techniques in the context of cheating and hacking, drawing on insights from industry professionals. The findings highlighted accessibility of both CI and RE, due to widely available prebuilt tools and other resources, such as communities and marketplaces. Although accessible, effectiveness often depended on technical expertise, as public tools relied on methods that modern security solutions can easily detect. Furthermore, the application of these techniques within both industries often relied on trusted Windows functionality to conduct attacks, especially BYOVD, which was used identically in both domains. Using these trusted functions, attackers can remain stealthy by blending with trusted systems. Finally, it was discovered that RE overlaps, as it uses identical tools and techniques for both offensive and defensive purposes, including discovering software vulnerabilities and analyzing malware. The same protective philosophy of obfuscation was also applied across domains to complicate the process of RE.

RQ2 examined the opportunities and barriers to collaboration between the AC and cybersecurity industries from professionals' perspectives. The findings revealed that the two industries share similar technical infrastructure, such as operating on the kernel level and process monitoring. Despite this, barriers such as the risk of oversharing information, asymmetric trust models, and product-specific implementation differences mean the potential of collaboration has not yet been realized at the enterprise level, even though technical overlaps exist. However, the study demonstrates that experience in cheat development, AI-powered detection capabilities, telemetry sharing, and improvements to the back-end

infrastructure are potential areas for collaboration. In addition, knowledge transfer is a finding that has not been adequately addressed in the existing literature.

6.1. Practical Implications and Recommendations

As both industries are affected by techniques that use similar methods to compromise, close collaboration through the sharing of telemetry and intelligence on emerging threats would be highly beneficial. As threats to security constantly evolve, preparation through updated systems and up-to-date threat intelligence is essential for operating proactively and reducing the risk of incidents.

The findings of this study highlight vulnerable kernel-level drivers as a popular method for conducting malicious activities due to the evasion capabilities they provide. Therefore, sharing information and blocklists of vulnerable drivers across domains would have strong practical implications, as it could improve proactive threat detection and mitigation. As AV often loads before AC systems, these vulnerable drivers could be blocked before being maliciously applied. This would be valuable in the context of a system compromise, as the attacker would be limited in their options for privilege escalation and evasion. The opportunity to collaborate by sharing information would also preserve confidentiality and mitigate the risk of over-disclosure, which was identified as a barrier, since sensitive details about security solutions would not need to be provided.

Both industries also described false positives as common, since trusted applications often use system functions that are synonymous with malicious attacks. Since attackers also use the same Windows features, a trade-off is necessary. Increasing detection sensitivity would increase the rate of false positives, while reducing it might allow attacks to go unnoticed, which could be detrimental to an organization's security. Additionally, a high volume of false-positive alerts could lead to increased stress and alert fatigue within security teams and increase the risk of overlooking critical events. Therefore, sharing information about legitimate software used in both domains could help reduce false positives, thereby improving the accuracy of security solutions such as AC, AV, or EDR systems.

As the analyzed techniques overlapped, the potential for cheaters to transition into cybersecurity or unethical hacking is plausible, and some respondents exemplified this shift. Game cheating was described as specializing in areas such as low-level Windows internals and RE, which were noted to be inadequately represented in formal education. Because of this, the cybersecurity industry could engage with cheating communities to identify individuals with in-depth knowledge and practical experience in these areas. This type of knowledge would be highly useful for security operations, including penetration testing and malware analysis. Some AC organizations have already begun doing this, suggesting that it could potentially also be feasible in cybersecurity contexts.

Finally, in the AC domain, the respondents described back-end server security as lacking. Once clients connect to a game server, it becomes vulnerable to attacks such as DDoS and rate-limiting abuse. Because of this, collaboration to acquire cybersecurity expertise could be valuable for securing game servers, as it is closely aligned with standard cybersecurity practices, according to the respondents.

6.2. Scientific implications and further research

Current research in the combined areas of cybersecurity and AC has primarily focused on attacker motivations and other factors that could have a direct influence on an individual's potential transition from cheating to cybercrime or cybersecurity [7, 8]. This study provides scientific implications by analyzing the difficulty of this skill transfer through an examination of how common offensive and defensive techniques overlap across the two domains.

Previous research has discussed the possibility of transitioning from cheating to cybersecurity [7], and this study provides direct examples of individuals who have made this transition. These individuals also provided insights that directly support this study's RQs, noting several overlaps and transferable technical knowledge across industries. This contributes by acknowledging direct similarities in the techniques used by both cheaters and hackers, including the ease of conducting effective attacks and

the emerging use of vulnerable drivers. However, motivational differences (i.e., intrinsic and extrinsic) between attackers across domains persist.

Given the overlap in technical expertise, future research should closely examine this potential career transition, including both opportunities and barriers, to help expand available talent in these fields. This knowledge transfer has not been thoroughly addressed in current academic research to the best of our knowledge. Therefore, future studies could assess the feasibility of this transition across cybersecurity and AC to identify the opportunities and barriers to establishing it as a widely recognized practice in both fields. This could be done through case studies of cheaters who have undergone this change to learn why they did it. It could also be complemented by AC developers who have hired cheaters to understand their reasoning. In addition, conducting interviews or surveys with cheaters on public forums such as UnknownCheats could also be an approach to examine their willingness to transition and the potential barriers that would hinder them.

With the emergence of AI tools for malware development, the barrier to entry for conducting malicious activities is continually decreasing. This enables less skilled attackers to carry out advanced attacks, which means organizations must employ strong, scalable detection and prevention mechanisms to adapt to these threats. Consequently, the cybersecurity industry has also begun implementing AI itself to assist with this. Although AI is also used in AC contexts, there is limited academic and industry literature on how it could be applied in AC environments and how to limit false positives, as some respondents noted. Addressing this would be valuable for keeping up with evolving cheats and for providing highly scalable, proactive security solutions in gaming.

Finally, research on collaboration between AC and cybersecurity is rare, and this study found that such collaboration was not straightforward. Organizations, especially those relying on proprietary security solutions, noted that collaborating by sharing technical details could compromise their defenses. Therefore, it is implied that collaboration involving sensitive details might be difficult to arrange in security-focused industries due to risks of over-disclosure. However, future research could further analyze opportunities for technical collaboration through interviews with cybersecurity and AC professionals. Interviews would provide industry insights into what information could be shared and what technical and trust-related requirements would need to be established before such collaboration could be fostered.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT 5.5 and Grammarly to perform grammar and spelling checks, and improve writing style. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] A. Ibrahim, Guarding the future of gaming: The imperative of cybersecurity, in: 2024 2nd International Conference on Cyber Resilience (ICCR), IEEE, 2024, pp. 1–9.

[2] A. Brandão, J. S. Resende, R. Martins, Employment of secure enclaves in cheat detection hardening, in: International Conference on Trust and Privacy in Digital Business, Springer, 2020, pp. 48–62.

[3] S. Collins, A. Poulopoulos, M. Muench, T. Chothia, Anti-cheat: Attacks and the effectiveness of client-side defences, in: Proceedings of the 2024 Workshop on Research on offensive and defensive techniques in the context of Man At The End (MATE) attacks, 2024, pp. 30–43.

[4] F. Martinson, D. Rangel, A comprehensive analysis of game hacking through injectors: exploits, defenses and beyond, Int J Comput Appl 975 (2023) 8887.

[5] D. Mitropoulos, D. Spinellis, Fatal injection: A survey of modern code injection attack countermeasures, PeerJ Computer Science 3 (2017) e136.

[6] V. Tanwar, K. Ramkumar, A survey on the role of reverse engineering in security attacks, in:

2023 International Conference on Research Methodologies in Knowledge Management, Artificial Intelligence and Telecommunication Engineering (RMKMATE), IEEE, 2023, pp. 1–6.
[7] S. Cho, J. Lusthaus, I. Flechais, The slippery slope: Exploring the parallels between game cheating and cybercrime through routine activity theory, in: 2023 APWG Symposium on Electronic Crime Research (eCrime), IEEE, 2023, pp. 1–13.
[8] M. Bada, Y. T. Chua, Understanding risk and risk perceptions of cybercrime in underground forums, in: 2021 APWG Symposium on Electronic Crime Research (eCrime), 2021, pp. 1–11.
[9] M. Althamir, A. Alabdulhay, K. Riad, A. Albual, Enhancing malware detection efficacy: A comparative analysis of endpoint security and application whitelisting, Journal of Theoretical and Applied Information Technology 102 (2024).
[10] S. Pontiroli, The cake is a lie! uncovering the secret world of malware-like cheats in video games, in: Virus Bulletin Conference, 2019.
[11] S. Lehtonen, et al., Comparative study of anti-cheat methods in video games, University of Helsinki, Faculty of Science (2020).
[12] J. Chavez, Vanguard x valorant, https://playvalorant.com/en-us/news/game-updates/vanguard-x-valorant/, 2024. Accessed: Jan. 27, 2026.
[13] C. Dorner, 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 International Conference on Availability, Reliability and Security, 2024, pp. 1–11.
[14] A. Maario, V. K. Shukla, A. Ambikapathy, P. Sharma, Redefining the risks of kernel-level anti-cheat in online gaming, in: 2021 8th International Conference on Signal Processing and Integrated Networks (SPIN), IEEE, 2021, pp. 676–680.
[15] Fortinet, What is antivirus protection, https://www.fortinet.com/resources/cyberglossary/antivirus-protection, n.d. Accessed: Jan. 28, 2026.
[16] Palo Alto Networks, What is the difference between advanced endpoint security and antivirus (av)?, https://www.paloaltonetworks.com/cyberpedia/advanced-endpoint-security-vs-antivirus, n.d. Accessed: Dec. 30, 2025.
[17] P. Karkallis, J. Blasco, G. Suarez-Tangil, S. Pastrana, Detecting video-game injectors exchanged in game cheating communities, in: European Symposium On Research In Computer Security, Springer, 2021, pp. 305–324.
[18] H. Alnabulsi, R. Islam, M. Talukder, Gmsa: Gathering multiple signatures approach to defend against code injection attacks, IEEE Access 6 (2018) 77829–77840.
[19] A. Abuabid, A. Aldeij, Cyber security incident response: The effectiveness of open-source detection tools in dll injection detection, Journal of Information Security and Cybercrimes Research 7 (2024) 29–50.
[20] J. Van den Broeck, B. Coppens, B. De Sutter, Obfuscated integration of software protections, International Journal of Information Security 20 (2021) 73–101.
[21] Y. Cao, R. Zhang, R. Liang, K. Chen, Evaluating the effectiveness of decompilers, in: Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis, 2024, pp. 491–502.
[22] A. Kanervisto, T. Kinnunen, V. Hautamäki, Gan-aimbots: Using machine learning for cheating in first person shooters, IEEE Transactions on Games 15 (2022) 566–579.
[23] M. Al-Asli, T. A. Ghaleb, Review of signature-based techniques in antivirus products, in: 2019 International conference on computer and information sciences (ICCIS), IEEE, 2019, pp. 1–6.
[24] R. M. Thomas, J. V. Francis, The evolution of anti-cheat systems: A literature review of ai-driven and behavioral approaches, International journal for research in applied science and engineering technology 13 (2025) 977–979.
[25] E. Mumford, Effective Systems Design and Requirements Analysis: The ETHICS Approach, Macmillan, London, 1995. doi:10.1007/978-1-349-13774-9.
[26] T. Zaid, S. Garai, Emerging trends in cybersecurity: a holistic view on current threats, assessing solutions, and pioneering new frontiers, Blockchain in healthcare today 7 (2024) 10–30953.
[27] H. Kallio, A.-M. Pietilä, M. Johnson, M. Kangasniemi, Systematic methodological review: develop-

ing a framework for a qualitative semi-structured interview guide, Journal of advanced nursing 72 (2016) 2954–2965.
[28] S. Rahimi, et al., Saturation in qualitative research: An evolutionary concept analysis, International journal of nursing studies advances 6 (2024) 100174.
[29] M. B. Powell, C. H. Hughes-Scholes, S. J. Sharman, Skill in interviewing reduces confirmation bias, Journal of Investigative Psychology and Offender Profiling 9 (2012) 126–134.
[30] V. Clarke, V. Braun, Thematic analysis, in: Encyclopedia of critical psychology, Springer, 2014, pp. 1947–1952.
[31] J. Thompson, A guide to abductive thematic analysis, The qualitative report 27 (2022) 1410–1421.
[32] V. Chepp, Comparative subgroup analysis in qualitative interview studies: does sample size matter?, Chest 166 (2024) 423–424.
[33] K. A. R. Richards, M. A. Hemphill, A practical guide to collaborative qualitative data analysis, Journal of Teaching in Physical education 37 (2018) 225–231.
[34] H. Noble, R. Heale, Triangulation in research, with examples, 2019.
[35] B. Saunders, J. Kitzinger, C. Kitzinger, Anonymising interview data: Challenges and compromise in practice, Qualitative research 15 (2015) 616–632.
[36] N. Al-Yateem, The effect of interview recording on quality of data obtained: A methodological reflection, Nurse researcher 19 (2012).
[37] C. Gordon, Beyond the observer's paradox: The audio-recorder as a resource for the display of identity, Qualitative research 13 (2013) 299–317.
[38] Swedish Research Council, Good research practice 2024, 2024. URL: https://www.vr.se/english/analysis/reports/our-reports/2025-07-03-good-research-practice-2024.html.
[39] B. Al Sabbagh, S. Kowalski, A socio-technical framework for threat modeling a software supply chain, IEEE Security and Privacy 13 (2015) 30–39. doi:10.1109/MSP.2015.72.
[40] P. Checkland, S. Holwell, Information, Systems and Information Systems: Making Sense of the Field, John Wiley & Sons, Chichester, 1998.

A. Appendix A - Interview Guide Anti-Cheat

A.1. Intro questions

1. Could you briefly describe your current role and what experiences you have with anti-cheat development?
2. What cheating techniques do you encounter most within the game(s) you work with?
    - Why do you think that is?

A.2. Code injection in Gaming

3. How are attacks such as code injection used within the game(s) you work with?
    - What are the most common types of injections?
4. What indicators or behaviors do you rely on to detect code injection attempts?
5. How effective do you consider current anti-cheat measures against injection-based cheats?
    - If bad: Why do you think that is? Is there anything you believe could enhance these defensive measures?
    - If good: What is it that makes it so reliable in your experience?
6. When a legitimate open-source software uses techniques that are commonly associated with cheating, such as OBS using graphics hook, how do you differentiate whether the intent is malicious or not without triggering false positives?

A.3. Reverse engineering in Gaming

7. In what ways are reverse engineering used by cheaters to bypass the anti-cheat methods put in place?
8. What measures are in place to prevent hackers from reverse engineering the games code?
    - How effective are these measures at detecting and deterring attempts of RE?

A.4. Client-side vs Server-side

9. When it comes to client-side and server-side anti-cheat, which one do you believe perform better at detecting attempts of code injection and reverse engineering
    - Why do you think that is?
    - Are there any drawbacks to that implementation?

A.5. Perceived Cross-industry collaboration

10. Do you have any experience with cybersecurity technology, such as antivirus or endpoint, detection, and response systems?
    - Do you think there is room for collaboration between these two industries?
    - In what ways do you think tools from anti-cheat could be adapted to cybersecurity?
    - In what ways do you think cybersecurity tools could be adapted to anti-cheat systems?

B. Appendix B - Interview Guide Cybersecurity

B.1. Intro Questions

1. Could you briefly describe your current role and what experiences you have with cybersecurity.
2. What type of attacks do you encounter most within the area you work in?
    - Why do you think these attacks are so prevalent?

B.2. Code injection in Cybersecurity

3. How are attacks such as code injection used within the areas you work with?
    - What are the most common types of injections?
4. What indicators or behaviors do you rely on to detect code injection attempts?
5. How effective do you consider current cybersecurity defensive measures against injection-based attacks?
    - If bad: Why do you think that is? Is there anything you believe could enhance these defensive measures?
    - If good: What is it that makes it so reliable in your experience?
6. When a legitimate open-source software uses techniques that are commonly associated with malicious behaviors, such as OBS using graphics hook, how do you differentiate if the intent is malicious or not without triggering false positives?

B.3. Reverse engineering in Cybersecurity

7. In what ways are reverse engineering techniques used by attackers to bypass cybersecurity mechanisms put in place?
8. What measures are in place to prevent people from reverse engineering the software or systems?
    - How effective are these measures at detecting and deterring attempts of RE?

B.4. Perceived Cross-industry collaboration

9. Do you have any experience with anti-cheat technology?
   If yes: Do you think there is room for collaboration between these two industries?

   - In what ways do you think tools from anti-cheat could be adapted to cybersecurity?
   - In what ways do you think cybersecurity tools could be adapted to anti-cheat systems?

   If no: Could you imagine some things from your industry, such as tools, that could potentially mitigate cheating, which commonly utilize techniques such as DLL injections or software tampering?

C. Appendix C - Interview Invitation

Information letter about thesis project in cybersecurity

We are two master students at Jönköping University studying cybersecurity. We are currently doing our final year thesis project and searching for participants to interview. Our study is about comparing attacks used in gaming to those used by malicious hackers, to discover potential overlaps between industries. We also aim to understand how both industries defend against the identified threats to discover the potential for cross-industry collaboration. For this study we look to interview both cybersecurity experts, such as SOC analysts and Penetration testers, and anti-cheat developers/researchers. You only need to have knowledge in one of these areas to participate.

Data collection will be conducted using semi-structured interviews and will take approximately 30-45 minutes of your time. Any personal data such as the names of participants and organizations will be anonymized. The only identifiable data that will be presented in the report is whether the interviewee was a cybersecurity expert or an anti-cheat developer/researcher.

The interviews can either be conducted online using a video conferencing tool of your choice, or physically if you are located close to central Jönköping (max 1 hour by bus or train).

Additionally, we would like to audio-record the conversation, if possible, to ensure accuracy of all your answers. This recording will be done using either the built-in recording software in the video conferencing system if done online or using a secure voice-memo application stored on an encrypted device if done in person. Other methods of documentation could also be used if preferred. However, please note that this could result in the interview being longer than expected.

If you agree to being audio-recorded, the recording will be deleted once the transcriptions have been completed, which should not take longer than a week. Transcriptions could also be provided if requested.

Participation in the interview is voluntary, and you are allowed to skip any of the questions if you deem them difficult to answer. It is also possible to withdraw from the interview at any time without providing an explicit reason.

If you have any additional questions, you can contact us at:

Linus Eriksson, ...@student.ju.se

Berk Can Altindag, ...@student.ju.se

You can also contact our academic supervisor, Joakim Kävrestad at ... for questions about the study.

D. Appendix D - Coding Tree

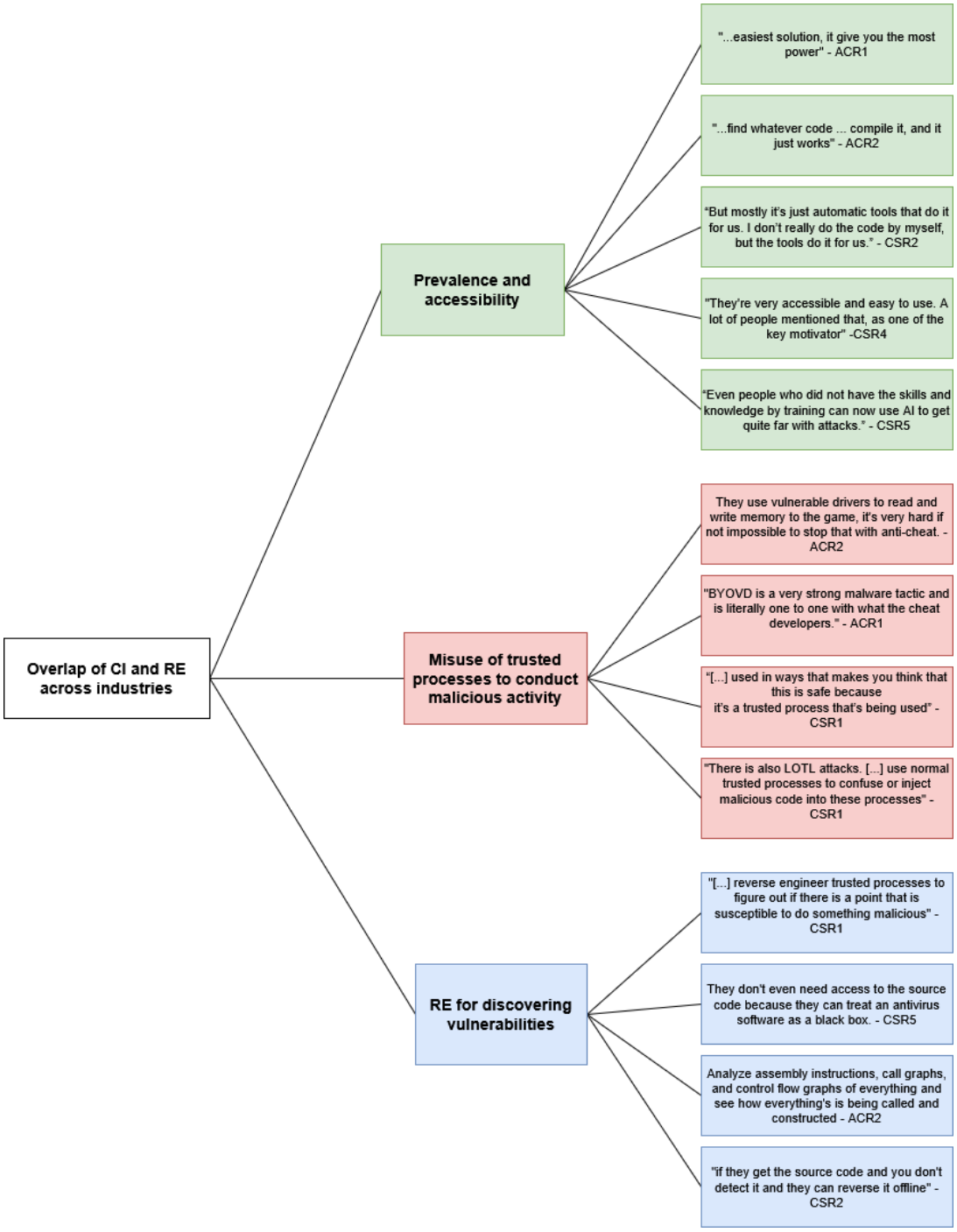


Figure 4: Example of coding tree