Cheaters Could Prosper: An Analysis of the Security of Video Game Anti-Cheat

Isaac Basque-Rice

School of Design and Informatics
Abertay University
DUNDEE, DD1 1HG, UK

ABSTRACT

Context: Video game anti-cheat software is of vital significance to the modern video game industry. Cheating in video games can have a seriously detrimental effect on both the industry itself and users' enjoyment of the industry's products. However many concerns have been raised by a wide variety of parties around the intrusiveness, security, and stability of many of these anti-cheat programs given their perceived intrusion into lower and lower levels of the users' operating systems, including kernel ring 0.
Aim: To evaluate the capability of video game anti-cheat software to effectively identify vulnerabilities that may have a detrimental effect on the end user.
Method: Given anti-cheat developers' well-known aversion to allowing easy investigation of their software, the primary method of analysis will be through reverse engineering the cheats themselves. This has the added benefit of certain vulnerabilities already being identified by the cheat developer, much like a proof-of-concept in the wider security field. Particular attention will be paid to those anti-cheat platforms that make use of kernel ring 0 permissions, as well as those software that have already demonstrated a lack of security. This is because they are the ones that are more likely to cause serious issues should they be compromised.
Results: This project will demonstrate the security of various anti-cheat platforms. Any vulnerabilities discovered will be responsibly disclosed to the relevant parties. Noteworthy findings outwith security concerns, such as novel techniques for performing certain technical actions, will be noted in the report and either disclosed or will form the basis of further work.
Conclusion: The information collected from work during this project will inform users and organisations of the possible risks present within the context of video game anti-cheat. If vulnerabilities are found in a popular anti-cheat platform this may result in significant quantities of devices being open to compromise, and, in the worst case, malicious actors appropriating vulnerable anti-cheat drivers for their malware campaigns independent of whether the games are installed.

Keywords

Anti-cheat, Kernel level, Cheat detection, Video game security, Game cheating & hacks, Games

1 INTRODUCTION

Cheating is a major problem in video games, particularly competitive online games. According to a report by Irdeto (2018), a cybersecurity services organisation for the digital media industry, "60% of online gamers feel [. . .] negatively impacted by other players cheating", and 77% of those same individuals would be likely to stop playing that online game if they believed others were attempting to gain an unfair advantage by cheating, thereby reducing interest in the game and harming the industry as a whole.

In response to this issue, many companies in the video game space have created software to detect and prevent cheaters. However, amongst many of those who enjoy competitive online gaming, the use of certain anti-cheat technologies is controversial. The matter of kernel-level anti-cheat, that is, the usage of kernel ring 0 by anti-cheat programs, is of particular concern to many users for reasons of privacy, system stability, and, of course, whether or not this level of access leaves the users' systems vulnerable to attack (Orallo 2020). This concern is, in fact, not unfounded. The kernel-level anti-cheat used in the popular role-playing game Genshin Impact, `mhyprot2.sys`, is used by ransomware actors in the wild to disable antivirus software before malware deployment (Soliven and Kimura 2022).

This in and of itself provides sufficient justification to undertake a project in this area, as the discovery of security flaws in these kinds of software could be of great concern to a large number of individuals, a total of 320 popular video games make use of this controversial technique (Pilipović 2022), representing a large section of the over 2 billion people strong PC video game market (Karkallis et al. 2021). However, very little is currently known about the internal workings of anti-cheat software. This is, in part, purposeful because anti-cheat developers are unable to discern malicious analysis (i.e., creating cheats) with genuine security analysis. As a result of this, the developers of these programs tend to obfuscate their code to a significant degree (cppcoder10291 2020). Nevertheless, many cheat developers can locate and exploit issues in anti-cheat to develop their programs, and it is within these that vulnerabilities may be discovered.

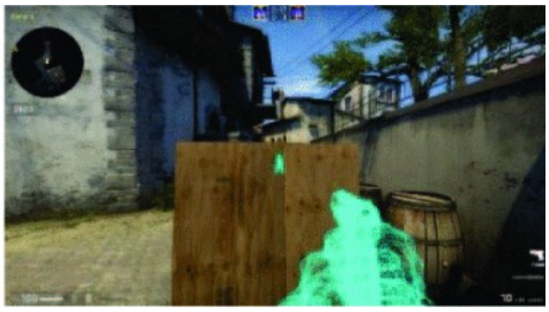

Figure 1 – An example of a user using so-called 'wall-hack' cheats in the video game Counter Strike: Global Offensive, (adapted from Maario et al. 2021)

This project aims to evaluate the capability of video game anti-cheat software to effectively identify vulnerabilities that may have a detrimental effect on the end user. The scope of this project can be deconstructed into 3 specific research questions:

1. How vulnerable are anti-cheat services, particularly the ones that make use of kernel ring 0 permissions, to compromise by a malicious actor?
2. How can the methods used in video game cheat and anti-cheat software be applied in a wider security context?
3. Given anti-cheat platforms' historically opaque approach to their technology, what insight can analysis of cheat software give to vulnerabilities in the anti-cheat software it is attempting to exploit?

2 BACKGROUND

The negative impact that cheating has on the experience of the players and the functioning of the video game industry cannot be overstated. This impact is well established across the literature to cover several domains, from the video game industry's profit, to user gameplay enjoyment, and, of course, the security of both. Hence, the discovery of these flaws (particularly in the field of security) is paramount to the continued success and enjoyment of video games generally.

Research carried out by Maario et al. (2021) concludes that current methods, particularly those which are run on the client side, are fundamentally inadequate to counteract cheating due to both the ”arms race” between anti-cheat and cheat developers (which the authors compare to malware development), and for the inherent security and privacy issues that come with kernel-level anti-cheat drivers, which are becoming ever more prevalent due to the aforementioned arms race.

The authors go into quite some detail regarding the concerns they have around this method of anti-cheat, specifically regarding false flags causing system instability, with the example of Riot's Vanguard system causing the device to overheat if it detected a specific RGB driver, developer misuse, providing the example of a rogue developer for ESEA (E-Sports Entertainment Association) misusing kernel-level access to install bitcoin miners, as well as the obvious possibility of a buffer overflow (or other such vulnerability) being exploited in the kernel-lever driver to allow arbitrary access to the entire target device.

Karkallis et al. (2021), however, were particularly interested in the use of so-called ‘injectors’ in cheat software, as well as the interactions that occur in online game cheating forums. Injectors themselves are of great importance in a wider security context as they may allow for an arbitrary, possibly malicious piece of code to be executed within the context of a separate process (i.e., the video game or kernel-level driver). The authors of this paper also put great emphasis on the community surrounding cheat development, and have thus identified UnknownCheats and MPGH as the two primary cheat distribution platforms, with hundreds of thousands of users on each, as well as the most common games that are being exploited on both, seen in Figure 2.

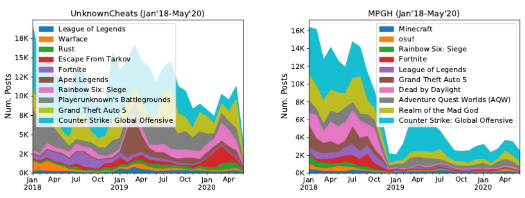

Figure 2 – Top 10 games by number of posts in UnknownCheats (left) and MPGH (right) from January 2018 until May 2020, (adapted from Karkallis et al. 2021)

Karkallis et al. also repeatedly refer to the similarity between cheat software communities and those around malware and other more illegal and/or illicit activities. This is done explicitly, with references to injectors themselves being used in malware deployment, recommended detection software using techniques ”widely proposed [. . .] for malware”, and repeated references to ”underground forums [. . .] similar to other online communities focused on illicit and even illegal activities”. This connection between cheaters and malware developers is continued when the authors refer to the ”strong cooperation which sometimes is promoted by a financial incentive”, akin to malware devs selling zero days and so on. Collectively, this depiction of the cheat community and malware development communities as being closely linked illustrates the potential for technical overlap.

Yan and Randell (2009) give great attention to the matter of video game security. The authors take the approach, in certain sections of the article, that video games are distributed internet applications and, as such, should evoke the same security concerns as other such applications. In many cases, such as in payment methods and hosting system security, these concerns are validated by providing security infrastructure, however, the authors argue that this is only because these concerns are shared across many internet applications. The authors are also careful to make the distinction between *game systems* (i.e., what can be seen on-screen) and *underlying systems*, which include things like targeting networking, operating systems, and so on. Anti-cheat is expected to be able to detect and prevent both of these, and as such both are of equal importance in the context of this work.

The authors conclude, amongst other things, that cheating is largely a result of various security failures, indeed, their taxonomy of online cheating concludes that exploiting design inadequacies in in-game systems is an exceedingly common method of cheating that can result in information theft, service denial, integrity failure, masquerade, and fairness violation.

3 METHOD

3.1 Research

The first phase of this stage of the report will be a review of the literature regarding the current state of the field of game cheat and anti-cheat development. This section will contain a critical analysis of prior research, including both academic sources such as peer-reviewed journal articles, conference papers, etc., as well as non-academic sources such as conference talk videos, articles, and other such online resources. Due to a lack of prior academic research in this field, non-academic sources are of particular utility to this project.

In addition to this, considerable research will be conducted into the online cheating community, their methods, forums, resources, and, of course, the games they are targeting. This is to achieve two goals, the first is to identify in what ways if any, the online cheating community's practises and knowledge can be appropriated into standard security practise, as well as any possible methods that could be used for the technical parts of the report itself, and the second is to gain a greater understanding of the state of the ”scene” to more effectively apply the findings of the report to a real-world context.

Finally, a small amount of research will also be conducted into tooling that can be used to some effect during the analysis phase. In particular, efforts will be made to locate and compare satisfactory reverse engineering tools such as Ghidra, Radare2, and IDA Pro.

3.2 Analysis

As previously mentioned, a result of anti-cheat platforms' desire to not be compromised by technically able cheaters is a considerable amount of effort being put into making sure their software is not truly understood by anyone outwith the team of developers working on said anti-cheat. Whilst this is an almost textbook example of security through obscurity, a disadvantage of this is if there were a serious security (or indeed other) flaw in the code base, it is much harder for legitimate security analysts to discover these issues and allow them to be fixed.

As a result of this, the methodology in this work will place much less emphasis on analysis of the anti-cheat systems themselves and instead focus on reverse engineering the cheats that attempt to exploit these systems. These cheats are more than likely to functionally serve as proofs-of-concept for vulnerabilities either within the anti-cheat itself or some other critical system within the game.

Particular focus will be placed on those cheats and anti-cheats that primarily exist on the client side. This is due to several factors. Firstly, because client-side cheats modify files on disk, any vulnerabilities that occur as a result of either the cheat or anti-cheat will necessarily also be on disk and therefore will present more of a threat to the end user. Secondly, however, the concerns many users have about anti-cheat invasiveness are most clearly realised when client-side services are running, occasionally without clear consent from the user, as an example, Valve's VAC has been reported to have accessed users' browsing history (Maario et al. 2021). Finally, both static and dynamic analysis can more easily be carried out on systems that exist primarily or exclusively locally.

Static analysis will primarily be conducted using a disassembler/decompiler tool such as Ghidra or Radare2, as well as other miscellaneous tools such as VirusTotal to ensure there are no malware signatures present and PeStudio to analyse executables for other possible concerns and Windows API calls that may give some hint as to what the software is doing

Regarding dynamic analysis, however, two methods will be explored, the first will be deploying cheats for specific games in a live environment. This will be done in such a way as to ensure no other users are negatively impacted, such as the creation of private lobbies. The second, however, is through the use of tooling such as KACE (Kernel Anti-Cheat Emulator), which allows users to deploy software developed for Kernel Ring 0 in a Ring 3 (userland) environment.

3.3 Evaluation

Finally, a critical evaluation of the findings in the Analysis section will be undertaken to achieve three goals, which are as follows. Firstly, to determine where weaknesses are in the anti-cheat platforms for them to be patched. Secondly, to more fully understand how, if indeed there is a way, to integrate methods used in the cheat and anti-cheat communities into other areas of cybersecurity and secure development. And lastly, to critically evaluate how successful the paper was in determining the inner workings of selected anti-cheat software via the chosen method of analysing software specifically designed to evade anti-cheat platforms' techniques, to apply this technique more widely.

4 SUMMARY

In summary, in carrying out this project, the aim will be to analyse the security and effectiveness of a selection of anti-cheat platforms through the technique of reverse engineering the very cheats that are attempting to circumvent these software platforms. This work, if successful, is to benefit the security community in several ways.

These ways include the possible discovery of security flaws, new techniques for detection evasion that may be applied in anti-virus or intrusion detection system contexts, and a greater depth of understanding around how to utilise exploits to comprehend the software they are attempting to exploit.

Prior research focuses primarily on improving the effectiveness of the anti-cheat software platforms, be it through newfound techniques, client-server models, machine learning algorithms, and so on. In contrast, this work is less concerned with the effectiveness of anti-cheat in its stated objectives, but more so with the security of the devices on which the anti-cheat is running, as well as what insights the respective communities can offer to the information security industry at large.

5 REFERENCES

cppcoder10291 (Feb. 10, 2020). *How to Reverse Engineer an Anti Cheat?* r/REGames. URL: www.reddit.com/r/REGames/comments/f1rgai/how_to_reverse_engineer_an_anti_cheat/ (visited on Oct. 2, 2022).

Irdeto (2018). *Irdeto Global Gaming Survey: The Last Checkpoint For Cheating*. Irdeto. URL: https://resources.irdeto.com/irdeto-global-gaming-survey/irdeto-global-gaming-survey-report-2 (visited on Oct. 1, 2022).

Karkallis, Panicos et al. (Oct. 4, 2021). "Detecting Video-Game Injectors Exchanged in Game Cheating Communities". In: *Computer Security – ESORICS 2021: 26th European Symposium on Research in Computer Security, Darmstadt, Germany, October 4–8, 2021, Proceedings, Part I*. Berlin, Heidelberg: Springer-Verlag, pp. 305–324. ISBN: 978-3-030-88417-8. DOI: 10.1007/978-3-030-88418-5_15. URL: https://doi.org/10.1007/978-3-030-88418-5_15 (visited on Sept. 28, 2022).

Maario, Anton et al. (Aug. 26, 2021). "Redefining the Risks of Kernel-Level Anti-Cheat in Online Gaming". In: 2021 8th International Conference on Signal Processing and Integrated Networks (SPIN), pp. 676–680. DOI: 10.1109/SPIN52536.2021.9566108.

Orallo, Aim (May 28, 2020). *Why Should You Worry about Kernel-Level Anti-Cheat?* Micky. URL: https://micky.com.au/why-should-you-worry-about-kernel-level-anti-cheat/ (visited on Oct. 1, 2022).

Pilipović, Šerif (Sept. 29, 2022). *Every Game with Kernel–Level Anti–Cheat Software (2022)*. URL: https://levvvel.com/games-with-kernel-level-anti-cheat-software/ (visited on Oct. 2, 2022).

Soliven, Ryan and Hitomi Kimura (Aug. 24, 2022). *Ransomware Actor Abuses Genshin Impact Anti-Cheat Driver to Kill Antivirus*. Trend Micro. URL: https://www.trendmicro.com/en_us/research/22/h/ransomware-actor-abuses-genshin-impact-anti-cheat-driver-to-kill-antivirus.html (visited on Oct. 1, 2022).

Yan, Jeff and Brian Randell (May 1, 2009). "An Investigation of Cheating in Online Games". In: *IEEE Security & Privacy* 7, pp. 37–44. DOI: 10.1109/MSP.2009.60.