Gaming the System: Kernal Exploits and the Flaws of Anti-Cheat



Noah VanDenBurg
Modesto Junior College

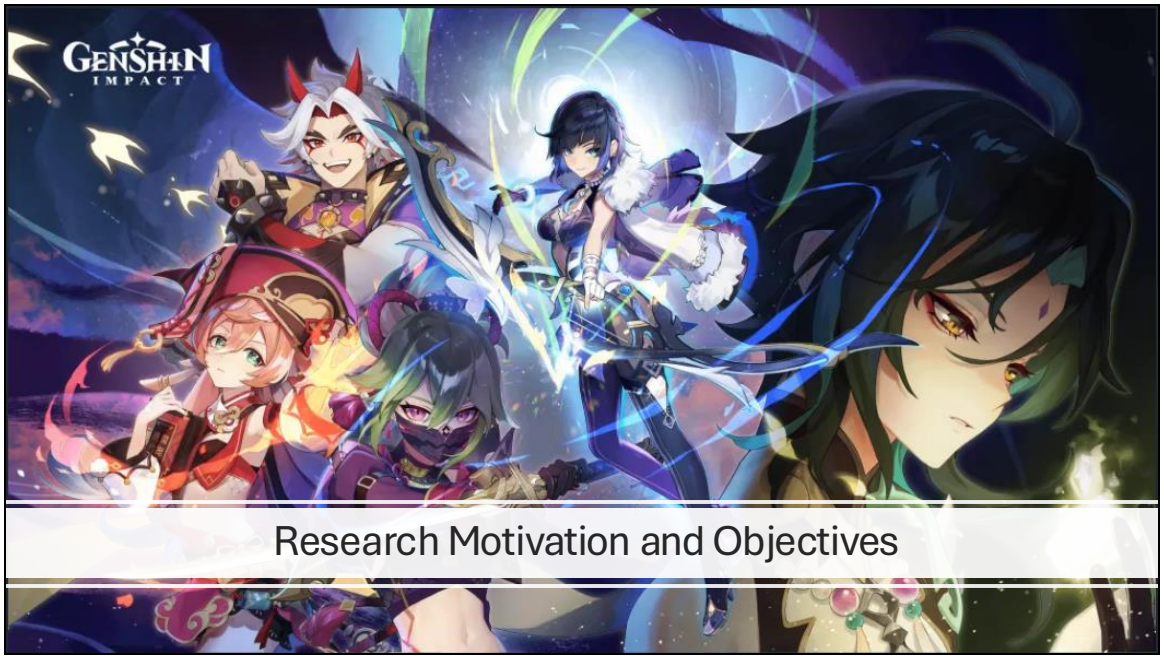




✓As technology evolves, gaming online is becoming more and more popular. The International Journal of Esports states the esports industry alone averages 1 billion dollars in annual revenue.

✓With many games offering rewards for in-game progression. These rewards often encourage fair competition among players, while others seek unfair advantages through unethical means, such as third-party software that enables a player to cheat.

✓Anti-cheat systems now included within modern games have been developed to detect and take actions against the use of these external programs by administering punitive measures such as shadow banning, account bans, hardware and network bans, etc. Therefore anti-cheat systems are essentially an effort to protect fair game play and to keep the ethical user base content.



Research Motivation and Objectives

✓Do they come at a cost?

Well, if you're considering your user's data and or their trust important, absolutely these concerns come at a cost for some individuals.

A lot of players and security enthusiasts find anti-cheat systems to be 'invasive' of their privacy. According to TrendMicro, a well known AntiVirus company, during "the last week of July 2022 a ransomware infection was found in a computer running a popular game called Genshin Impact. After analysis, they found a vulnerability in the anti-cheat system that is built into this game" (TrendMicro)

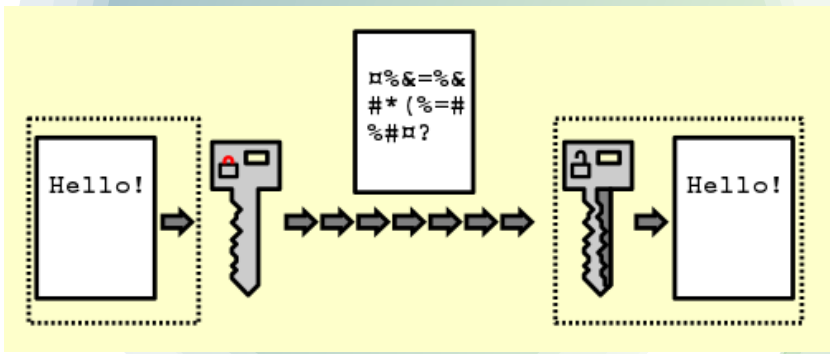
The vulnerability was that it specifically did not restrict unprivileged access, allowing all users to execute malicious code with administrative privileges on Windows systems that had the game installed.

That means our problem has escalated from a video game concern to real world system security concern.



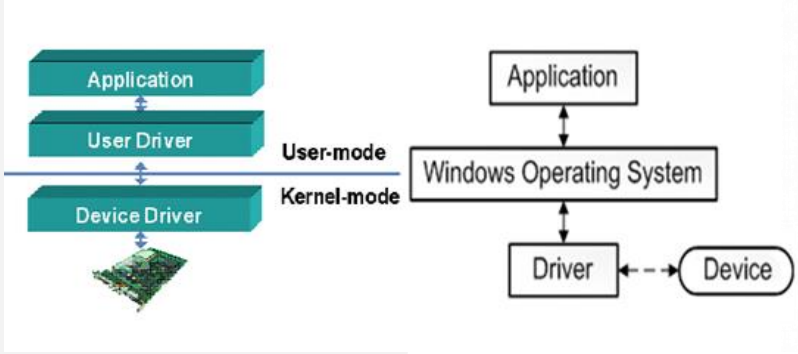
✓The incident with *Genshin Impact* highlights the broader implications of security vulnerabilities in modern technology. This isn't the only incident though as in July of 2024, millions of Windows systems began failing and showing the infamous Blue screen of Death (BSoD) due to a compromised update from a well known security vendor called *CrowdStrike which* cost many Fortune 500 companies an estimated $5.4 billion dollars of loss in just 7 days. This scenario is something I personally felt the impact of when working at Starbucks as a Barista, when we had to completely shut down our drive thru's as the CrowdStrike attack directly affected our own store's computer system.

Anti-Cheat Obfuscation Tactics



What do anti-cheats even do?
Anti cheat systems regularly utilize obfuscation tactics. This is the act of making something difficult or unclear to understand.
Code encryption is one of the most effective methods. By encrypting game code, developers make it difficult for cheat developers to reverse-engineer the game's logic. Tools like **IDA Pro** or **Ghidra** are commonly used to analyze the code, but encryption makes this process much harder, as the code becomes abstract and difficult to understand, as you can see in the example images.
File hashing is a mathematical process that ensures that game files haven't been altered. Every file has a unique hash value, and if even one byte changes, the hash value will be also be affected.
Signature scanning is used to identify known cheats. Anti-cheat systems generate signatures for known cheats like **PhantomOverlay** or **Aimware**. If a player is reported, the system scans for these signatures, and if there's a match, the player is banned.
Another increasingly popular method is using **kernel-based, or root level, anti-cheat system files**. These work by allowing system files, such as a device driver, to monitor interactions with the game process at a lower level, making it harder for the cheats to remain undetected. Anti-cheat systems such as **BattleEye** and **Easy Anti-Cheat** work this way, detecting and blocking cheat programs before they can even interfere with

the game.

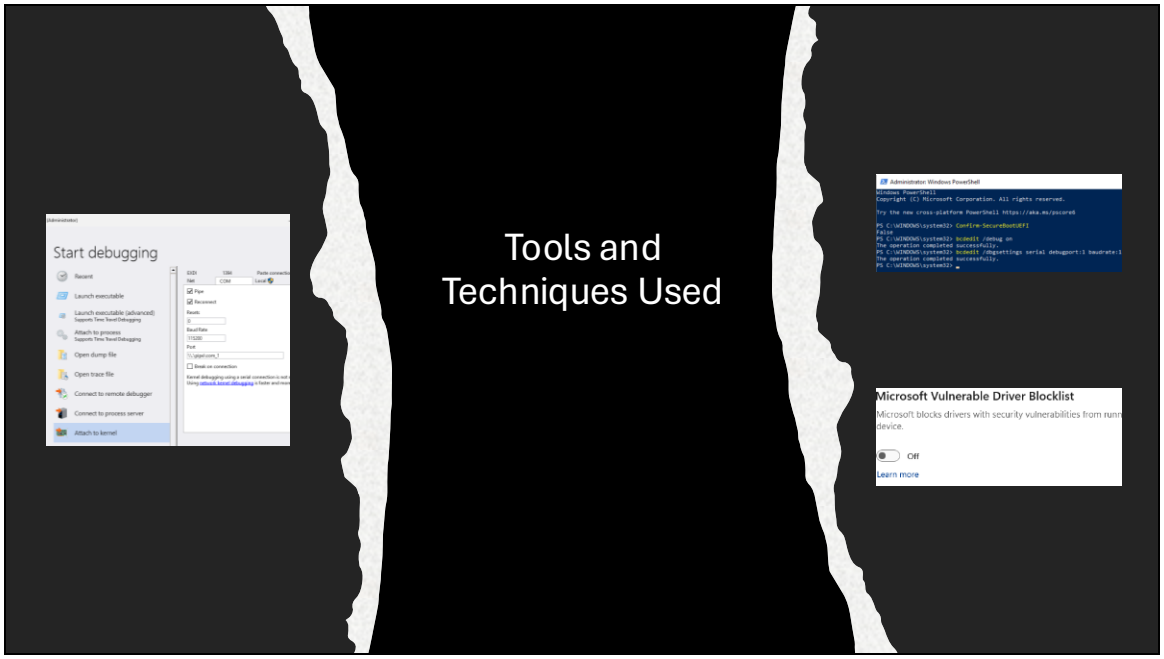


Impact of Device Drivers in Operating System Security

If you're wondering what is a driver? A driver acts like a translator which allows the computer's operating system to communicate with various hardware devices.

✓According to a study published by the Institute of Electrical and Electronics Engineers, also known as IEEE, "Modern operating systems are large and complex, with systems like Linux and Windows containing thousands of bugs. In fact, using a conservative estimate, the Linux operating system is likely to have around 15,000 bugs, while Windows has double that number. This is concerning because about **70%** of the operating system is made up of **device drivers**, which have **error rates** significantly higher than regular code."

Due to the complex nature of Operating Systems and their device drivers, Institute of Electrical and Electronics Engineers has also said that no one person can fully understand the entire systems, leading to challenges in designing and maintaining reliable software. Thus our **operating system**, being the foundation of all software interactions, is often the most vulnerable to security issues. Which allows it to become an easy target for anti cheat systems and those who are creating these cheat systems.

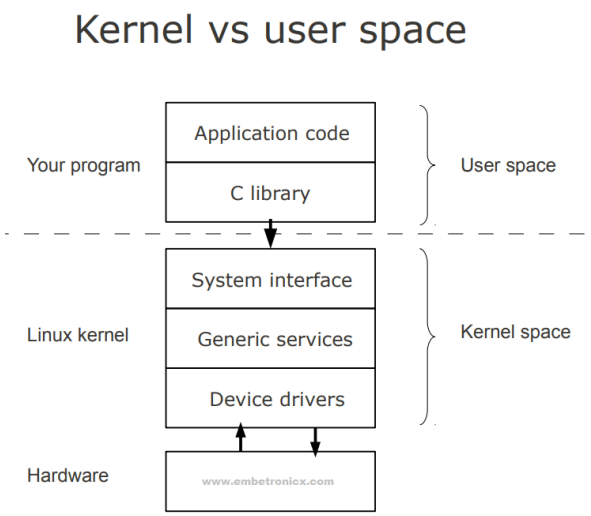


So, let's take a look at how both a user and developer of game cheat systems might use a popular program like KDMapper to bypass an anti-cheat system.
Essentially, KDMapper exploits a known vulnerability in a particular Intel driver. The exploit allows someone to get Kernel level system access which means we now have the ability to affect system memory putting our operating system at further risk for security issues.

✓Before we can proceed using KDMapper, we need to properly configure our system. First, we must **disable Secure Boot**. This is crucial, as it prevents unauthorized changes to the boot configuration data, which is needed for the cheat system to operate.

We'll also need to **disable default security settings**, particularly the **Microsoft Vulnerable Driver Blocklist**, which automatically blocks the loading of vulnerable or potentially unsafe KNOWN drivers.
With these steps in place, we can proceed with the driver development process alongside KDMapper.

Development of Kernel Driver for Exploration



Driver development involves several concepts with the two most important being driver loading and communication.

As for Driver loading: When a driver is loaded into the system, it becomes part of the kernel. It needs to be signed and verified—unless system protections are disabled like we just did.

And for Communication: Drivers must communicate with their user-mode applications using (Input/Output Control codes). These are command-like codes sent through references, telling the device driver exactly what it needs to do.

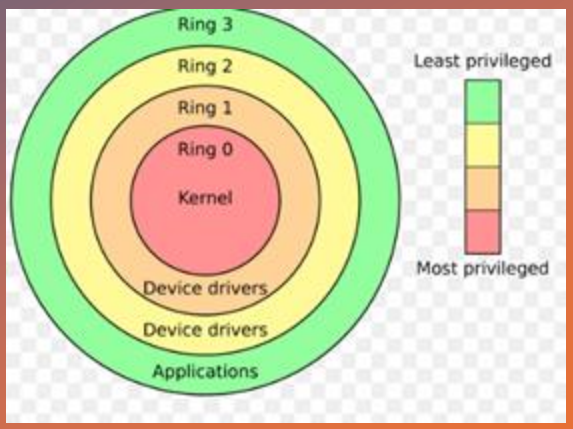
We can consider each of these in different senses:

- **An Input code is** a request from user mode to read/write or manipulate something in the system.
- **An Output code is** the result or data returned from to driver back to user mode.

Understanding (Input/Output Control codes) is key for any successful and legitimate interaction between a user application and a system level device driver.

Creating the User-Mode Counterpart





✓Now that we've covered the technical aspects of driver initialization and communication, let's look at how this applies to bypassing anti-cheat systems like Valve Anti-Cheat (VAC).

Since VAC operates in user-mode (Ring 3), it lacks the permissions to detect any kernel mode driver which lies in Ring 0 (**kernel-mode**). This privilege gap allows our kernel-mode driver to remain undetected by VAC essentially bypassing many of it's security checks.

When creating the usermode counterpart first we need to **obtain the Process ID** of the game and **attach our driver** to it. This attachment will let our driver access the game's memory, enabling memory manipulation.

Using our driver we can now read and write directly to the game's memory while remaining **undetected by VAC** since these operations are occurring in kernel-mode through our driver and being transmitted to our user mode application via the Input/Output codes.

Doing this allowed me to **modify the local_player's jump offset** in Counter Strike, allowing me to **automatically jump** in the game without detection, giving me a slight advantage in my gameplay.

KDMapper is essential for staying undetected here, but how was KDMapper's method even found?

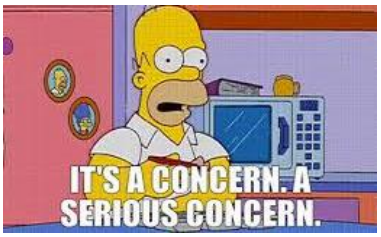
Vulnerabilities in Intel's Network Driver

Metrics CVSS Version 4.0 | CVSS Version 3.x | CVSS Version 2.0

NVD enrichment efforts reference publicly available information to associate vector strings. CVSS information contributed by other sources is also displayed.

CVSS 3.x Severity and Vector Strings:

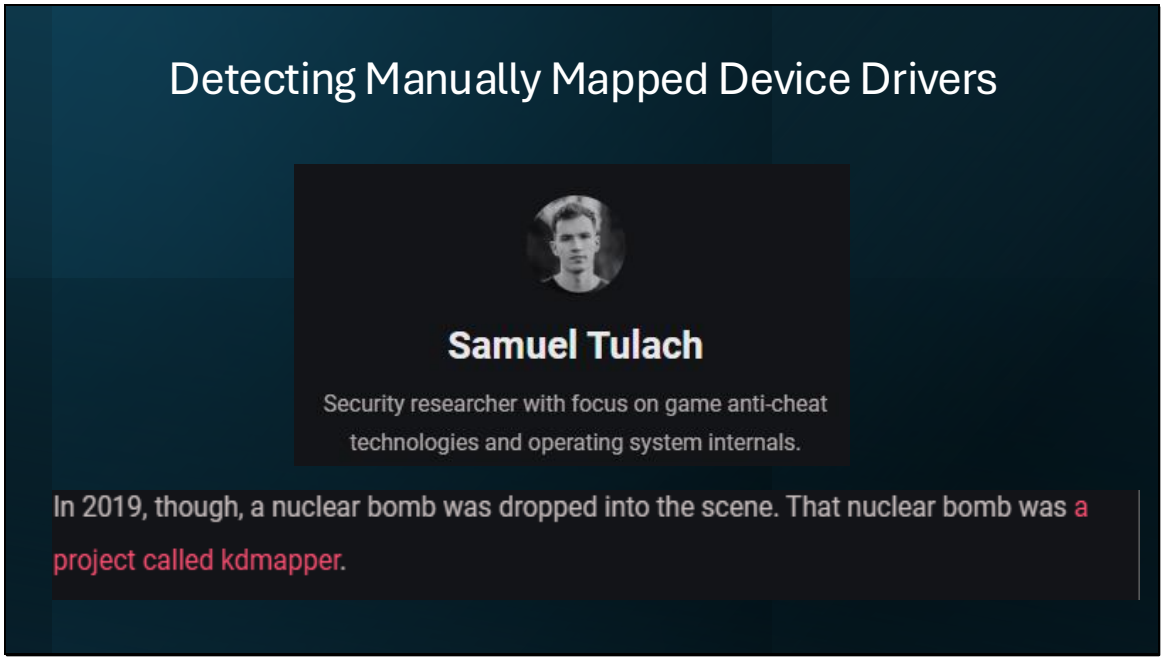
NIST: NVD **Base Score:** 7.8 HIGH **Vector:** CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H



✓In 2016, a vulnerability was found in **Intel's Ethernet diagnostics driver**. This vulnerability allows local users to execute custom code with **system level privileges** via specially coded Input and Output codes. This vulnerability was identified by the **National Institute of Standards and Technology (NIST)** and received a pretty **high severity score** of 7.8 out of 10 due to the potential for attackers to gain full control of the system, like we saw the Genshin Impact exploit do.

Similarly, in our example, we used **Input and Output codes** in our device driver to obtain **read** and **write** functionality at the system level essentially exploiting this vulnerability in exactly this network driver. This showcases that **Input and Output codes** can be leveraged to gain kernel-level permission, showing both the risk and power that comes with using device driver vulnerabilities.

So are device driver exploits just able to bypass everything then?

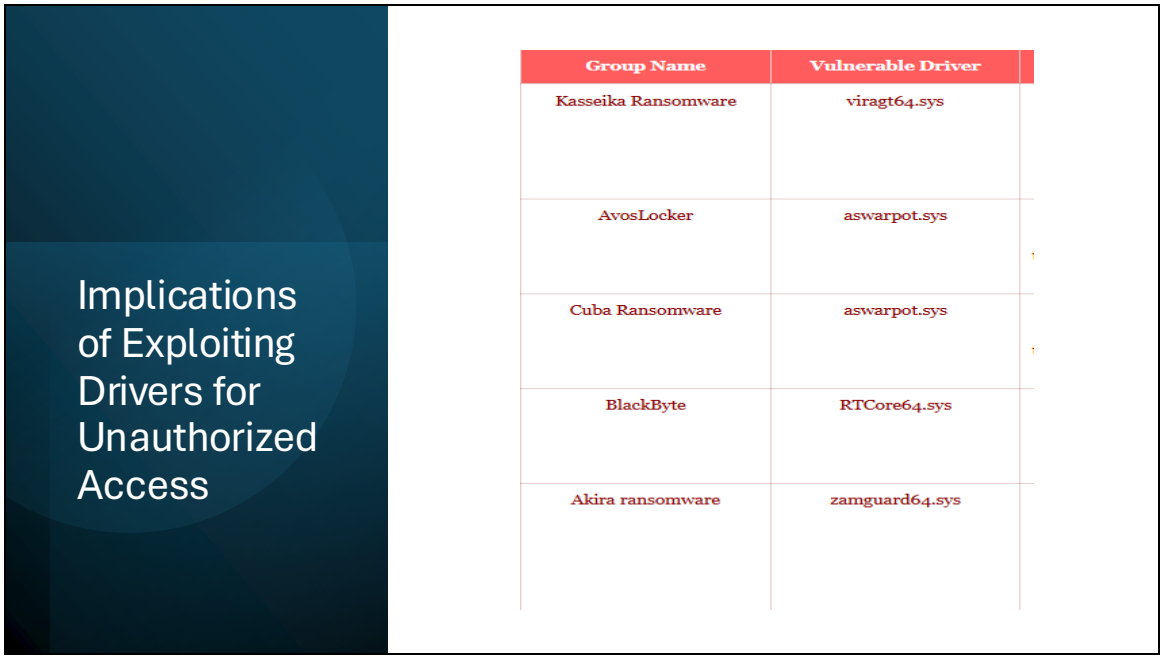


Samuel Tulach a security researcher who focuses on anti-cheat systems and protecting the deepest parts of our computers, the **kernel**. Shares a prominent method for detecting **hidden or unauthorized device drivers, which are** system files that run deep on our systems, and are typically used by cheaters or hackers to gain more control.
Normally, your computer knows what drivers are installed and where they are. But some hackers try to **sneak in modified drivers** manually by injecting them straight into memory, hoping the system won't notice. This is called **manual mapping**, and it's kind of like sneaking into a party without ever having your guest invitation checked.

Samuel's method works by scanning all of our computer's memory and looking for any signs that something has been changed. Specifically, it looks for **"jumps" in code**, which are shortcuts or route redirections that happen when a driver is manually inserted. If the system finds a jump that goes to a suspicious location—outside of typical trusted software, it's a key flag that something may be modified

Once the system finds something suspicious, it runs a deeper analysis to figure out if it's actually malicious. If it turns out to be a cheat or some other kind of malicious device driver the system can flag all users who have ever run that driver and apply

bans in large amounts.

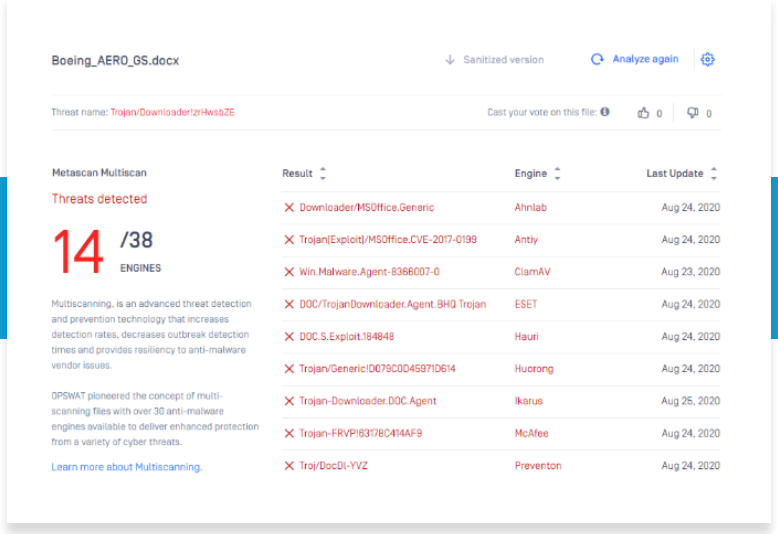


Alright so, is driver hacking even a big issue?
Yes, and it's increasingly important as cybersecurity and gaming evolves. While it may not be entirely visible to mainstream media, Kernel Exploits are steadily becoming more and more popular in the malware and game hacking industry. The Lazarus (hacking group) has used kernel exploits on vulnerable Dell drivers to bypass some Windows security protections and install backdoors like the North Korean Blindingcan Remote Acess Trojan Malware. Based on CISA and FBI malware analysis results, the BLINDINGCAN malware can also remove itself from compromised systems and clean its traces to avoid detection. Essentially meaning it can steal all of your data and leave no trace.



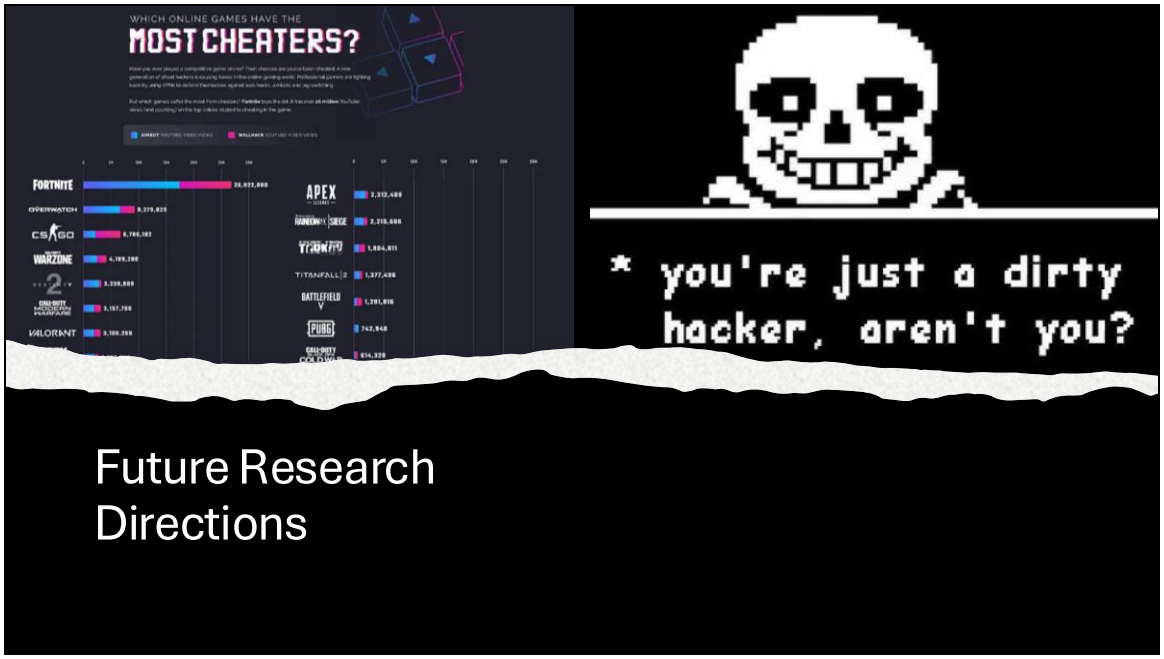
This kind of low-level attack isn't just a concern for national security—it directly ties into the world of anti-cheat systems, which face similar challenges in balancing deep system access with user privacy.

✓According to **Ruben Greidanus** work in his master's thesis, "Where anti-cheat is concerned, the privacy implications stretch far beyond that of the realm of the game itself, and it is imperative to keep that in mind when analyzing the problem and ultimately working towards a solution." This statement really emphasizes how the impact of anti-cheat systems goes beyond gaming, often entering a **gray** area in terms of privacy. I think at this point, it's crucial for us to find a balance between **security** and **privacy**. This may be hard to implement in a fast-evolving technologic world like we face today. As technology advances, so needs our knowledge as we must continuously adjust to new realities in the tech world, which makes it increasingly difficult to maintain the balance of these two concepts when designing anticheats.



Summary of Key Findings

•Recapping what we've learned today.
•**In Anti-cheats we saw** the online gaming industry is amazing, but with it comes significant responsibility and ethics on the cybersecurity side. We want to encourage fair play, so we must implement anti-cheat systems. However, these systems do come with privacy and security risks that we must address as a community.
•Another key topic was **Vulnerable kernel drivers.** We saw they present major security issues in both gaming and real-life scenarios. While these drivers are essential for operating systems, they also present significant risk. Though the IEEE said that these systems are complex, I think as the years progress, we can try and minimize security risks to prevent more vulnerable drivers.
•Lastly **Security and privacy** was also a real challenge when implementing strong anti-cheats because we must balance these two concepts efficiently, which can be difficult.



How can we begin to improve anti-cheats?
We can begin to enhance anti-cheat systems while also protecting privacy: Research must consider the ability to identify cheats and tools without invading players' privacy. We should look for innovative ways to develop systems that would not always need to scan the entire system, but instead utilize statistical analysis instead.

Conclusion
So, today we saw that in the effort to make our gameplay fair, we could really be introducing new vulnerabilities not typically considered. I am reminded of my experience of dealing with the CrowdStrike inccedent when I was working at Starbucks last year. By trusting CrowdStrike's updates, the entire Starbucks chain experienced the unintended consequences of trusting systems to make their business effectively operate, just as a gamer trusts an anti-cheat system to secure their systems and data. We both unknowingly introduce new threats by letting our guard down and trusting these providers.
With that I conclude today's presentation, it was fun and I hoped you learned something. Thank you for your time.

References

- https://www.trendmicro.com/en_us/research/22/h/ransomware-actor-abuses-genshin-impact-anti-cheat-driver-to-kill-antivirus.html
- https://stackoverflow.com/questions/6007176/what-is-a-windows-kernel-driver
- https://www.unknowncheats.me/forum/anti-cheat-bypass/447986-manual-map-mean.html
- https://www.techtarget.com/whatis/feature/Explaining-the-largest-IT-outage-in-history-and-whats-next#:~:text=The%20outage%20was%20not%20a,real%20time%20across%20the%20OS.
- https://helda.helsinki.fi/server/api/core/bitstreams/89d7c14b-58e0-441f-a0de-862254f95551/content
- https://www.linkedin.com/advice/0/what-role-device-drivers-operating-system-xrhpc
- https://nvd.nist.gov/vuln/detail/CVE-2020-36603#:~:text=Description,privileges%20on%20Microsoft%20Windows%20systems.
- https://www.cs.vu.nl/~ast/Publications/Papers/computer-2006a.pdf
- https://www.youtube.com/watch?v=n463QL4cjsU
- https://tulach.cc/detecting-manually-mapped-drivers/
- https://www.securityjoes.com/post/security-s-achilles-heel-vulnerable-drivers-on-the-prowl
- https://www.bleepingcomputer.com/news/security/us-govt-exposes-new-north-korean-blinding-can-backdoor-malware/
- https://amo.uvt.nl/show.cgi?fid=142935;client-side

Photos

- https://imgs.search.brave.com/Ipi0VGzi80v999XL8pMaVfh0ghDxBaTUjifi4GgF0Jg/rs:fit:860:0:0:0/g:ce/aHR0cHM6Ly9kZXZlbG9wZXIuYXBwbGUu/Y29tL2xpYnJhcnkv/YXJjaGl2ZS9kb2N1/bWVudGF0aW9uL01h/Y09TWC9Db25jZXB0/dWFsL09TWF9UZWNo/bm9sb2d5X092ZXJ2/aWV3L2Fyd C9vc3hf/YXJjaGl0ZWN0dXJl/LWtlcm5lbHNfZHJp/dmVyc18yeC5wbmc (kernel)
- https://idstch.com/wp-content/uploads/2022/07/device-drivers-660x319.png (driver explanation)
- https://bpb-us-e1.wpmucdn.com/sites.suffolk.edu/dist/5/1153/files/2021/09/cheating.jpg(bart)
- https://www.techspot.com/images2/news/bigimage/2022/08/2022-08-26-image-21-j_1100.webp (genshin)
- https://static.wikia.nocookie.net/cswikia/images/0/00/VAC_ban.png/revision/latest?cb=20151211194459 (vac)
- https://i.ytimg.com/vi/J84UfRwwTzg/maxresdefault.jpg (BSOD)
- https://hasonss.com/blogs/wp-content/uploads/2023/12/OS-structure.webp (OS)
- https://www.gao.gov/assets/2022-07/Data%20privacy%201.jpg (data privacy)
- https://upload.wikimedia.org/wikipedia/commons/thumb/7/70/Public_key_encryption_keys.svg/500px-Public_key_encryption_keys.svg.png (encryption)
- https://embetronicx.com/storage/2017/08/kernel-space-vs-user-space.png (kernel picture)