Universidad Nacional Autónoma de México

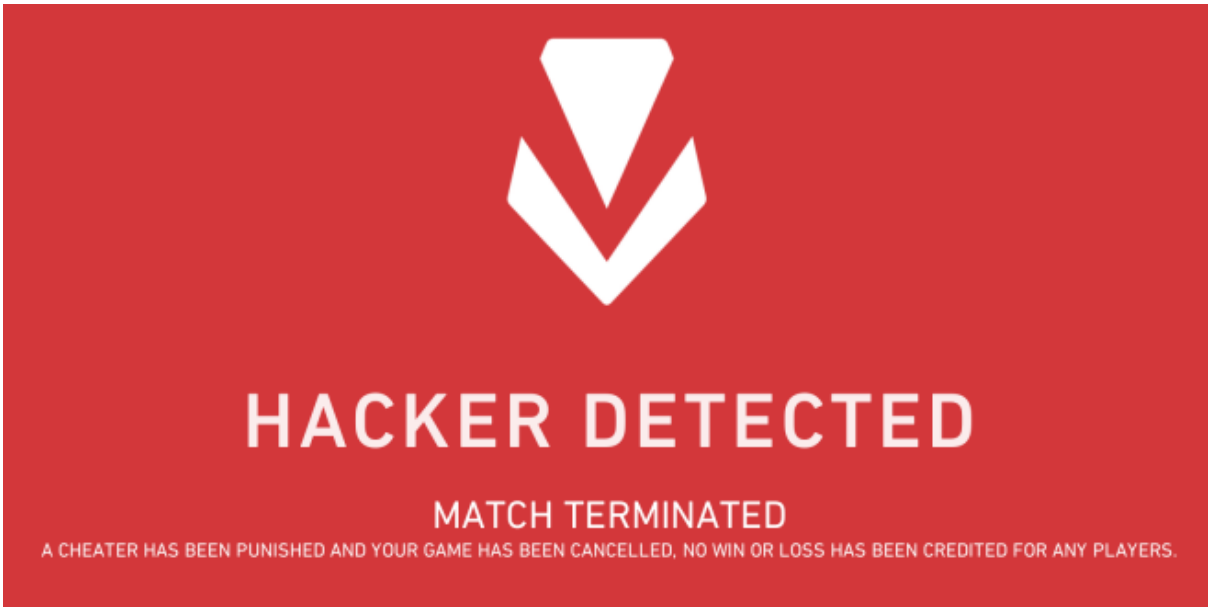
Facultad de ingeniería

Sistemas operativos gpo. 6

Prof. Gunnar Eyal Wolf Iszaevich

Alumno: Eduardo Miguel Paniagua Broca

Presentación “Anticheat y el problema del acceso al kernel”



Introducción

Desde que jugamos juegos unos contra otros siempre han habido personas dispuestas a romper las reglas con el fin de tener la ventaja por encima de sus competidores, ya sea contar cartas, alterar dados o desconectar el control de tu hermanito pequeño, mientras haya una competencia que ganar habrá alguien dispuesto a hacer lo que sea necesario para asegurar la victoria.

Los trucos y trampas en los videojuegos tienen un orígen curioso, ya que en muchas instancias eran creados por los propios desarrolladores para facilitar el proceso de debugging así como para hacer el juego un poco más disfrutable para los jugadores no tan experimentados en lo que dominaban las mecánicas fundamentales del mismo. Un ejemplo bastante conocido de este tipo de trucos es el famoso código konami, implementado por primera vez en el port para la NES del juego Gradius permitía acceder a todas las mejoras así como 30 vidas extras al ingresar la siguiente secuencia:

Arriba, Arriba, Abajo, Abajo, Izquierda, Derecha, Izquierda, Derecha, B, A, Start

Este código fue puesto en el juego para facilitarle a los testers el acceso a los niveles más complicados del juego sin tener que practicar durante horas y horas para poder llegar a estos de manera normal. Por algún motivo este no fue removido antes de lanzar el juego al mercado y los jugadores se encariñaron con este peculiar truco, por lo que haría apariciones en múltiples títulos de la empresa.

La demanda por más maneras de romper los juegos que tanto amamos creó un mercado para la venta de dispositivos que podían insertarse en la consola y estos modificaban los valores almacenados en memoria antes de que el juego terminara de arrancar, permitiendo a los jugadores acceder a powerups de manera inmediata, saltar niveles y en general experimentar el juego en maneras que distaban bastante de lo pretendido por los desarrolladores. Los dos productos más populares de este tipo fueron el Game Genie de Codemasters y el GameShark de la empresa homónima.

El uso de trampas dejaría de estar limitado a la consola del usuario con la proliferación de los títulos con jugabilidad en línea a finales de los 90's y principios de los 2000's, en este caso el uso de trampas no solamente afectaba al usuario del software, sino que también afectaría la experiencia de las demás personas dentro de la partida. Fue este cambio en particular lo que hizo que las compañías desarrolladoras de juegos se tomaran en serio la tarea de combatir estas prácticas con el fin de garantizar una buena experiencia para sus clientes.

¿Cómo funcionan las trampas y el software que las combate?

Existen múltiples tipos de trampas que proporcionan ventajas específicas para el juego al que están destinadas, por ejemplo, a continuación se presenta una breve lista de las trampas más comúnmente empleadas en los juegos first person shooter:

- Wallhack: Le permite al jugador ver a sus contrincantes a través de los muros del mapa, proporcionando una ventaja al momento de tomar decisiones
- Aimbot: En términos generales hace que la mira del jugador se coloque de manera automática sobre el oponente
- Hacks de movimiento: Se saca provecho del motor del juego para que el jugador pueda interactuar con el mapa de manera distinta a la planeada por el desarrollador

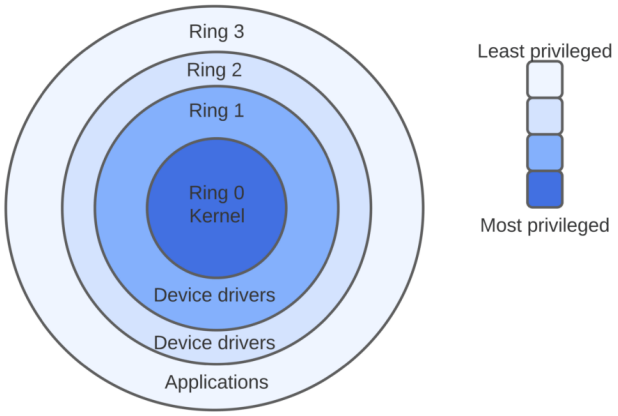
Una clasificación más enfocada en el aspecto técnico del funcionamiento de las trampas las divide en dos tipos: internas y externas.

- Internas: En este tipo de trampas se basan en insertar código dentro de la memoria designada para la ejecución del juego, el método más común es conocido como inyección DDL (Direct Link Library injection). En este método un programa denominado como el inyector le pedirá al SO acceso a las direcciones de memoria correspondientes al juego, una vez consigue el acceso el archivo DDL va a escribir sobre estas direcciones de memoria y el nuevo código será ejecutado como si fuera una parte más del juego, una vez hecho esto el inyector puede ser cerrado e incluso borrado y las trampas seguirán funcionando mientras no se cierre el juego.

- Externas: En este caso en vez de escribir sobre la memoria del juego, el programa externo sólamente lee la memoria para obtener información acerca de la partida y con esa información generar entradas para el teclado y mouse, ya sea a través de los drivers de estos dispositivos o por medio de la API de Windows (por ejemplo, la clase C++ SendKeys.Send).

Las trampas de tipo externas son bastante más difíciles de detectar debido a que no modifica nada dentro del juego, y las entradas de teclado y ratón que generan son indistinguibles de las que generaría el usuario a través de los periféricos. Es esta complicada situación la que ha orillado a múltiples desarrolladores a tomar una decisión que al día de hoy genera controversia entre los interesados en el tema: **Acceder al kernel del SO.**

Los anillos de protección en un sistema operativo son una manera de representar los distintos niveles de acceso a los recursos del equipo que tiene cada programa, para la arquitectura x86 en modo protegido existen 4 anillos de protección:



- Anillo cero: En este se encuentra el kernel del sistema operativo, es decir, aquella parte que opera en modo normal y tiene mayores probabilidades de causar grandes cambios dentro del ecosistema.
- Anillos 1 y 2: Están destinados a los periféticos de entrada y salida, a menudo se presentan situaciones en las que deba pedir permiso.
- Anillo 3: Este es el entorno en el que opera la gran mayoría de aplicaciones que usamos diariamente (tanto consciente como inconscientemente).

Si bien el acceso al kernel abre la posibilidad de crear software anti trampas más efectivo que nunca (aquellos que lo implementan tienden a ser los más efectivos) mucha gente lo ve como un enorme riesgo a la seguridad de su equipo de cómputo y considera que bajo ninguna circunstancia se le debería otorgar ese nivel de poder sobre el equipo a una empresa tan solo por querer disfrutar de un juego.

A continuación se muestra una tabla de algunos programas anti trampas que ocupan acceso al kernel:

Nombre	Empresa desarrolladora	Juegos que lo ocupan
Battleye	BattlEye Innovations	H1Z1, PUBG, Rainbow Six Siege
Vanguard	Riot games	Valorant
EasyAntiCheat	Epic Games	Fortnite, Fall Guys, Battlefield 2049
Punkbuster	Even Balance Inc.	Medal of Honor, Battlefield 4, Far Cry 3
EA anticheat	Electronic Arts	Fifa 23

Conclusiones:

La realización de este trabajo me llevó por caminos distintos a los que tenía planeados, pues en un principio planeaba enfocarse exclusivamente en los programas anti trampas con acceso al kernel del sistema operativo, sin embargo, conforme me adentré cada vez más en el tema fuí capaz de hallar cada vez más información de temas que me parecieron interesantes y que van acorde a lo visto en el curso, llegando así a ser parte de esta exposición.

Bibliografía consultada:

- *Tarantola, A. (2019) "A brief history of cheating at video games". Recuperado el 31/05/2023 de https://www.engadget.com/2019-06-15-a-brief-history-of-cheating-at-video-games.html*
- *"Understanding how cheats work"* (s.f.). Recuperado el 31/05/2023 de https://steamcommunity.com/sharedfiles/filedetails/?id=1873537304