O Impacto de Software Anti-cheat na Privacidade do Usuário

Vinicius Matheus[1], Tiago Heinrich[2], Vinicius Fulber-Garcia[1], Newton C. Will[3], Rafael R. Obelheiro[4], Carlos A. Maziero[1]

[1] Departamento de Informática
Universidade Federal do Paraná (UFPR)
Curitiba – PR – Brasil

[2]Max Planck Institute for Informatics (MPI)
Saarbrücken – Saarland – Germany

[3]Departamento de Ciência da Computação
Universidade Tecnológica Federal do Paraná (UTFPR)
Dois Vizinhos – PR – Brasil

[4]Programa de Pós-Graduação em Computação Aplicada (PPGCAP)
Universidade do Estado de Santa Catarina (UDESC)
Centro de Ciências Tecnológicas – Joinville – SC – Brasil

{vmcf19,vinicius,maziero}@inf.ufpr.br, theinric@mpi-inf.mpg.de,
will@utfpr.edu.br, rafael.obelheiro@udesc.br

Abstract. *The video game market is constantly growing, and with it, the use of cheat software, where users cheat to gain unfair advantages. As a consequence, game developers dedicate efforts to developing techniques to detect and prevent malicious users from using such software. One of the strategies used is the adoption of anti-cheat software. However, anti-cheats typically operate intrusively on their users' systems, sometimes requiring execution permissions at the kernel level, raising concerns about the privacy and security of the personal data collected. This paper presents a technical analysis of anti-cheats, investigating their operations looking for potential privacy breaches for users.*

Resumo. *O mercado de jogos eletrônicos está em constante crescimento e, com ele, o uso de* softwares *de* cheat*, onde os usuários trapaceiam para obter benefícios irregulares. Como consequência, desenvolvedores de jogos dedicam esforços ao desenvolvimento de técnicas para detectar e impedir que usuários mal-intencionados usem tais* softwares*. Uma das estratégias utilizadas consiste na adoção de* software anti-cheat*. No entanto, os* anti-cheats *tipicamente operam de maneira intrusiva nos sistemas de seus usuários, podendo requerer permissões de execução até em* kernel-level*, gerando preocupação quanto à privacidade e segurança dos dados pessoais coletados. Este trabalho apresenta uma análise técnica de* anti-cheats*, capturando e investigando suas operações executadas em busca de potenciais brechas de privacidade para os usuários.*

1. Introdução

O número, a variedade e o valor agregado dos jogos eletrônicos vêm crescendo continuamente nos últimos anos [Goh et al. 2023]. Nesse cenário, jogadores que desejam vencer a

qualquer custo recorrem a trapaças para obter vantagem de maneira desonesta e alcançar a vitória. Existe uma variedade de trapaças, desde as inofensivas e previstas, implementadas pelos próprios desenvolvedores de um jogo, até as mais lesivas e agressivas, como *softwares* chamados de *hacks* ou *cheats*, criados com a intenção de manipular ilegalmente operações e recursos em um jogo. Estes *softwares* alteram propriedades fundamentais do jogo, fornecendo vantagens aos jogadores maliciosos, chamados de *cheaters*.

Um agravante que incentiva o uso de *cheats* está relacionado a fraudes financeiras, onde moedas virtuais são vinculadas a moedas reais. Jogos com essas características motivam ainda mais jogadores maliciosos a trapacear. Essa trapaça pode ocorrer, por exemplo, através de *duping*, que é uma maneira desonesta de duplicação de itens, como moedas, por meio da exploração de vulnerabilidades de um jogo [DeLap et al. 2004]. Empresas que desenvolvem e mantêm jogos com esse tipo de moeda devem redobrar a atenção para a prevenção de *cheats*, já que trapaças envolvendo moedas virtuais causam perdas financeiras para jogadores e desenvolvedores [Greidanus 2017].

Para prevenir e detectar *cheaters* em jogos, diferentes estratégias podem ser usadas, tais como: (i) coleta de denúncias de jogadores dentro do jogo; (ii) uso de *software* especializado nos servidores do jogo; e (iii) uso de *software* especializado instalado nas máquinas dos usuários. A primeira técnica requer especialistas para analisar as denúncias, aceitando ou rejeitando as suspeitas levantadas [Eung et al. 2006]. As demais estratégias definem os *anti-cheats*, que podem ser executadas no *server-side*, onde o monitoramento de comportamentos é realizado pelo servidor em busca de padrões que representem *cheaters*, ou no *client-side*, onde o *software* é instalado diretamente na máquina do usuário, monitorando e coletando informações do sistema [Greidanus 2017]. Cabe ressaltar que um *client-side anti-cheat* consome recursos dos computadores dos jogadores e acaba sendo vulnerável a alterações maliciosas feitas por *cheaters*, já que está diretamente acessível no computador do usuário, podendo ser analisado e burlado [Ven 2023].

Assim, visando aperfeiçoar a detecção de usuários trapaceiros, os *anti-cheats* estão se tornando cada vez mais intrusivos, com permissões abrangentes no dispositivo. Eles são executados junto ao sistema, mesmo na ausência de um jogo em execução. Suas ações são tornadas obscuras para dificultar ao máximo o trabalho de engenharia reversa feito pelos desenvolvedores de *cheats*, evitando a descoberta de pontos fracos que poderiam ser explorados [Greidanus 2017]. Além disso, para realizar uma análise detalhada, alguns *anti-cheats* adotam a estratégia de enviar arquivos das máquinas dos jogadores para verificação remota em servidor. Essas ações causam uma grande preocupação com a privacidade dos usuários, já que não é possível saber exatamente quais dados estão sendo coletados [Ronkainen 2021].

Exemplos de suspeitas de invasão de privacidade por parte de *anti-cheats* são vastos e públicos. Como mencionado em [Wendel 2012], após uma atualização no *anti-cheat* do jogo *World of Warcraft*, uma parte da comunidade sentiu sua privacidade afetada e, para jogar, optava por fechar todas as aplicações possíveis, além de abandonar ocasionalmente o jogo por completo. Em [Pontiroli 2019], é apontado que alguns *anti-cheats* já impediram jogadores de se conectar a servidores apenas por usarem a palavra-chave "cheat" em navegadores web. Ainda, o *anti-cheat* Vanguard, criado pela Riot Games, no início do seu desenvolvimento, erroneamente detectou *softwares* e *drivers* básicos de teclado e mouse como trapaças, gerando mau funcionamento de *drivers* fundamentais do sistema

e interferindo em atividades essenciais, como a refrigeração dos componentes, causando superaquecimento de computadores [Ven 2023].

A análise das operações realizadas por *softwares anti-cheat* é crucial para entender seu impacto na privacidade dos usuários. Este trabalho visa monitorar a atividade e operações de *anti-cheats*, incluindo o acesso a diretórios, permissões concedidas, comunicações de rede, e consultas ou modificações nos registros, identificando aspectos que possam comprometer a privacidade dos usuários. As principais contribuições são:

- Rastreamento de interações e recursos acessados por *softwares anti-cheat*; e
- Investigação e discussão ampla sobre o impacto dos *softwares anti-cheat* na privacidade e segurança do usuário e sistemas.

O artigo é organizado da seguinte forma: a Seção 2 explica os conceitos-chave; a Seção 3 discute os desafios associados aos *softwares anti-cheat*; a Seção 4 detalha a proposta considerada; a Seção 5 discute os experimentos e resultados; a Seção 6 relaciona pesquisas na área; e, finalmente, a Seção 7 conclui o artigo.

2. Fundamentação Teórica

Esta seção apresenta os principais conceitos que fundamentam o presente trabalho.

2.1. Segurança & Privacidade

Na área de computação e desenvolvimento de *software*, a segurança contra ameaças engloba um conjunto de cinco propriedades fundamentais: confidencialidade, integridade, disponibilidade, autenticidade e irretratabilidade [Maziero 2020]. Em geral, um *software* é considerado seguro em relação a ataques de usuários maliciosos se essas cinco propriedades são garantidas.

A privacidade, por sua vez, é definida como o direito ao isolamento de informações, onde os indivíduos devem ser protegidos da publicação de qualquer detalhe considerado confidencial [Doneda 2006, Iachello et al. 2007]. Atualmente, a privacidade é um direito fundamental, incluída na Declaração Universal dos Direitos Humanos.

Com o aprimoramento da tecnologia e surgimento da Internet, a privacidade se tornou um recurso fundamental [Machado 2014]. Particularmente, em computação, a privacidade da informação é frequentemente discutida, sendo uma área identificada como a medida de quanto as pessoas decidem quais informações são coletadas sobre elas; ou, ainda, como o risco de informações pessoais serem usadas indevidamente [Schellens 2021].

No trabalho de [Shin 2010], destaca-se que, no contexto de *software*, privacidade e segurança são frequentemente confundidas. Esse equívoco ocorre porque problemas de segurança muitas vezes resultam em violações de privacidade, vinculando os dois conceitos. Por exemplo, uma falha na confidencialidade expõe inevitavelmente dados pessoais, comprometendo a privacidade dos usuários.

Alguns países possuem leis que regulamentam a privacidade no desenvolvimento de *software*. No Brasil, por exemplo, existe a Lei Geral de Proteção de Dados Pessoais (LGPD), que é semelhante à *General Data Protection Regulation* (GDPR) em vigor na União Europeia. Essas legislações determinam regras que devem ser observadas por todos os *softwares* que circulam em seus países.

Na LGPD, regras definem que usuários devem ser capazes de solicitar a exclusão de suas informações pessoais; as empresas devem fornecer um relatório detalhado de como serão utilizadas as informações dos clientes; e os usuários devem poder de recusar o tratamento e manipulação de qualquer um de seus dados [Dias Canedo et al. 2020].

2.2. *Cheat Software*

Cheat é um tipo de *software* utilizado para obter vantagem em jogos eletrônicos, fornecendo benefícios aos seus usuários de maneira ilegitima [Ronkainen 2021]. Esses *softwares* podem ser divididos em diversos tipos, como, por exemplo, aqueles que não afetam o comportamento fundamental do jogo, como *bots*. *Bots* são *cheats* que executam ações automáticas dentro dos jogos, permitindo que o jogador humano se ausente. Estes geralmente são usados para adquirir recursos automaticamente durante longos períodos, sem a necessidade de interações do usuário com o jogo.

Outros *softwares* de *cheats* podem alterar diretamente as regras fundamentais dos jogos, modificando suas leis ou automatizando ações essenciais, o que oferece vantagens significativas aos *cheaters*. Esses *cheats* podem injetar código malicioso durante a execução do jogo para alterar dados em memória e modificar o comportamento do jogo, ou capturar dados durante a execução para realizar ações perfeitas. Exemplos incluem *Aimbot*, *Triggerbot*, *Wallhacks*, *Maphacks*, *No Recoil*, *Speedhack* e *Lag Switches* [Ronkainen 2021].

2.3. *Software Anti-cheat*

Anti-cheat é um tipo de *software* desenvolvido especificamente para detectar *cheats* e alertar sobre, ou punir, *cheaters* [Ven 2023], aplicando desde restrições na jogabilidade até o total banimento do jogador [Lan et al. 2009]. Os *anti-cheats* guardam uma lista de todos os jogadores que já utilizaram *cheats* em seus jogos; alguns são mais agressivos, e quando detectam o uso em um jogo, banem o jogador em todos os jogos que usam o mesmo *anti-cheat* [Wendel 2012].

Os *anti-cheats* utilizam várias estratégias para impedir que jogadores trapaceiem em seus jogos. Essas estratégias incluem o reconhecimento de padrões recebidos pelo servidor, verificando se o jogador não apresenta picos de desempenho em determinados momentos ou se ações inconsistentes e não permitidas foram enviadas. Além disso, realizam a análise de integridade dos arquivos e o escaneamento de memória durante a execução dos jogos na máquina dos usuários [Maario et al. 2021]. Os *anti-cheats* podem ser divididos em duas grandes categorias, baseadas no local de execução: *server-side*, que são executados apenas no servidor, e *client-side*, instalados nas máquinas dos jogadores.

Atualmente, existem soluções *anti-cheat* sendo vendidas e usadas em jogos, tanto do tipo *server-side*, quanto *client-side*, ou ainda soluções híbridas. As soluções *server-side* funcionam conforme as regras do jogo definidas pelo servidor. Assim, cada ação executada pelos jogadores e cada estado de jogo têm sua validade verificada no servidor, evitando que modificações e injeções de código nos clientes afetem o servidor [Maario et al. 2021]. Já os *anti-cheats client-side* são instalados diretamente nas máquinas dos jogadores, analisando a memória dos computadores e podendo detectar até os *cheats* discretos e ofuscados. Estes também liberam recursos dos servidores, pois utilizam os recursos das máquinas dos clientes; contudo, acabam ficando vulneráveis a

estudos, alterações e injeções de código pelos *cheaters* [Ven 2023]. Alguns dos *anti-cheats* mais populares são *Easy Anti-Cheat*, *BattlEye*, *Valve Anti-Cheat* e *Vanguard* [Pilipovic 2023, PCGamingWiki 2023].

3. Problemática

Alguns *softwares anti-cheat* requerem acesso ao nível de *kernel*, gerando preocupações dos usuários quanto à violação de sua privacidade e à instabilidade que esses *softwares* podem causar no sistema. Além disso, qualquer vulnerabilidade nesses *anti-cheats* pode fornecer acesso privilegiado a atacantes, comprometendo a segurança do sistema [Basque-Rice 2023]. Um agravante é que alguns *softwares anti-cheat* são iniciados antes mesmo de seu respectivo jogo ser aberto. Um exemplo é o Vanguard, executado assim que o sistema operacional do usuário é iniciado [Ven 2023].

Formalmente, empresas que oferecem soluções de *anti-cheat* utilizam um termo de consentimento que os jogadores devem aceitar antes de instalar o *software*. Esses termos geralmente informam quais ações serão executadas e quais informações serão coletadas, além de avisar que apenas dados fundamentais para a detecção de trapaças serão analisados. Porém, o escopo dessas necessidades não é bem estabelecido, sendo nebuloso ou ocultado detalhes do que é considerada uma análise fundamental [Greidanus 2017].

Com o uso de *anti-cheats*, já foram registrados casos de invasão de privacidade, como o caso do VAC, *anti-cheat* da Valve. Tal solução foi detectada acessando diretamente o histórico de pesquisas dos usuários [Maario et al. 2021]. Também, já houve casos de falha segurança, como o *anti-cheat* utilizado no jogo Genshin Impact, mhyprot2.sys, que possui uma vulnerabilidade que possibilita atacantes desabilitar *softwares* de antivírus [Basque-Rice 2023].

Um caso popular envolvendo *anti-cheats* ocorreu com o ESEA *anti-cheat*. Assim como o Vanguard, o ESEA *anti-cheat* se mantém ativo mesmo que um jogo monitorado não esteja em execução. Dessa forma, por possuir permissões privilegiadas no computador dos jogadores, e ter a natureza de ser silencioso e esconder suas ações, um funcionário de sua desenvolvedora inseriu *malware* de mineração de *bitcoins*, que executou oculto e sem a anuência dos usuários [Greidanus 2017].

4. Proposta

Este trabalho visa realizar uma análise detalhada da execução e das ações realizadas por *softwares anti-cheat* disponíveis no mercado, buscando e evidenciando pontos que possam afetar a privacidade dos usuários. Para isso, foram utilizadas algumas ferramentas, descritas a seguir, para capturar dados relevantes para a avaliação das ações executadas por esses *softwares*, como alterações em registros, permissões concedidas aos processos, diretórios acessados, *dynamic link libraries (DLLs)* utilizadas e dados transmitidos na Internet.

Como objeto de estudo serão utilizados os *anti-cheats* BattlEye [BattlEye 2024] e Vanguard [Riot 2024b], tanto por sua relevância e popularidade no cenário atual de jogos digitais quanto por suas características técnicas: (i) o Vanguard é executado automaticamente durante o *boot* do *host* e permanece ativo mesmo sem o jogo monitorado estar em execução, além de haver diversos relatos de problemas com a interrupção de processos

não relacionados a *cheats*; (ii) o BattlEye foi escolhido por ser executado em *kernel-level* (assim como o Vanguard), por sua grande popularidade, e, diferente do Vanguard, por não se tornar ativo desde a inicialização do sistema operacional.

Para observar as interações dos *softwares* com os sistemas subjacentes, foram utilizadas três ferramentas: (i) *Process Explorer*, que permite coletar os diretórios acessados pelos *anti-cheats*, verificar incoerências, *DLLs* carregadas e funções com comportamento invasivo; (ii) *Process Monitor*, que permite monitorar as alterações no registro do sistema, averiguando se os *anti-cheats* executam alterações ou consultas desnecessárias, desviando de sua função e afetando a privacidade; e (iii) *Wireshark*, utilizado para capturar e avaliar o tráfego de rede gerado por esses *softwares*. Por fim, analisam-se também as licenças das soluções *anti-cheat* consideradas, onde se espera encontrar informações sobre os comportamentos observados empiricamente.

5. Avaliação

Esta seção apresenta e discute em detalhes o processo de coleta e os dados obtidos sob um aspecto técnico, operacional e burocrático.

5.1. Coleta de dados

Para a coleta dos dados, foi empregado um computador com o Windows 10 Pro versão 22H2 Build 19045.4046, *Process Explorer* versão 17.05, *Process Monitor* versão 3.96 e Wireshark versão 4.2.3-0-ga15d7331476c. As versões dos *softwares anti-cheat* são vinculadas às versões dos jogos monitorados. Para a execução do *BattlEye*, foram utilizados os jogos *Unturned* versão 3.24.1.0 e *Tibia* versão 13.34.14623; já para o *Vanguard*, o *Valorant* versão 8.02 foi considerado. Foram realizadas três capturas com cada ferramenta, separadamente, para evitar conflitos entre elas, para cada jogo considerado. Esse número se mostrou satisfatório, já que não existiram diferenças significativas entre as capturas.

A captura de dados do *Vanguard* apresentou alguns desafios, pois sempre que o *Process Monitor* estava em execução, o jogo exibia uma tela de erro, solicitando a reinicialização do sistema. Após diversas tentativas e mudanças de configuração, como ativar e desativar o *Secure Boot* e TPM 2.0, foi possível capturar algumas execuções completas do *Vanguard* durante partidas de *Valorant*. Essa situação, no entanto, não foi enfrentada com o *BattlEye*, que permitiu a execução plena do *Process Monitor*.

A principal diferença entre o *BattlEye* e o *Vanguard* é observada na forma como o *anti-cheat* é integrado ao sistema. O *BattlEye* é executado junto ao jogo, sendo iniciado e encerrado independentemente do ciclo de vida do sistema. Já o *Vanguard* emprega um *driver* iniciado com o sistema *Windows*. Ambos modelos de execução podem ser observados pelo *Process Explorer*, já que os *anti-cheats* iniciam um processo ao executar o jogo ao qual estão vinculados, sendo esse processo o alvo da análise. Assim, para realizar a coleta de dados, o processo padrão consistiu (i) na abertura da aplicação responsável pela captura de dados e (ii) na abertura do jogo, que inicia a execução do *software anti-cheat*.

5.2. Análise de dados

Com o *Process Explorer*, foi possível obter a lista de *DLLs* com as quais a aplicação interage durante seu processo de execução, bem como os arquivos abertos e os registros

do sistema. Os *softwares anti-cheat* não parecem dedicar esforços para ocultar os recursos acessados durante sua execução. Identificou-se que 42% das *DLLs* observadas são comuns a ambos os *softwares*, com as respectivas aplicações acessando recursos como funções ao nível de *kernel*, além de informações sobre o sistema, como hardware, dispositivos e eventos gerados, através das *DLLs ntdll* e *ntmarta*. As soluções também utilizam *DLLs* para a manipulação da *shell* do *Windows*, como *SHCore*, *shell32* e *shlwapi*, que possibilitam executar qualquer comando do sistema e, assim, obter diversas informações, desde o sistema de arquivos até processos e *drivers* em execução.

Através do *Process Monitor*, foi construído um esquema de traços de execução da aplicação, identificando as principais operações realizadas pelos *anti-cheats*. Esse traço de execução visa representar todo o processo, desde o início até o encerramento da execução, evidenciando os padrões que também permitem a comparação entre diferentes *softwares anti-cheat*, para a compreensão de suas semelhanças e diferenças.

A Figura 1 exibe o traço de execução identificado para o *BattlEye*. A sequência de execução observada no *BattlEye* começa com o carregamento das *DLLs* do sistema. Em seguida, há a leitura do registro do *Safer* (uma *API* do *Windows* que controla a política de execução das aplicações no sistema) e da chave *CodeIdentifiers*, indicando o local onde reside um arquivo de *log* contendo todas as aplicações executadas no sistema, qual processo iniciou e a regra usada para avaliar a permissão de execução [Microsoft 2024].

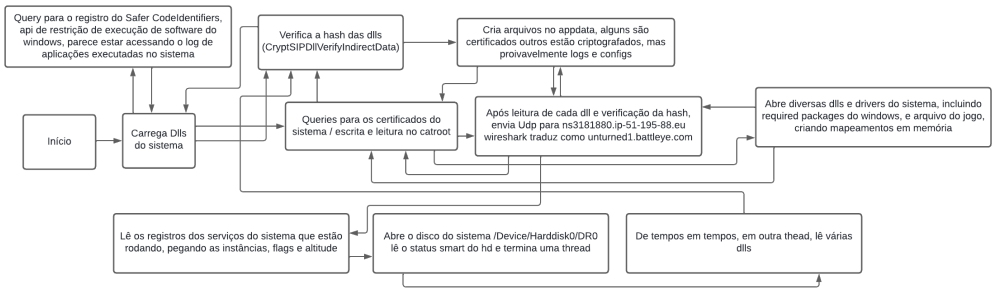

Figura 1. Representação do traço de execução do *BattlEye*.

O *anti-cheat* continua com o carregamento de *DLLs*, verificando a integridade das mesmas via *hashes* e a função *CryptSIPDllVerifyIndirectData*. Ao mesmo tempo, são feitas *queries* para os registros de certificados do sistema e escritas/leituras no diretório *CatRoot*, que contém certificados do sistema, indicando uma verificação de autenticidade dos arquivos usados pelo *anti-cheat*. Durante esse processo, alguns arquivos são criados no diretório *AppData*. A análise desses arquivos revelou que consistiam em certificados ou arquivos criptografados (potencialmente *logs* de execução).

Após os eventos relatados, um padrão parece se repetir durante a execução do *BattlEye*, consistindo na abertura de uma *DLL* ou *Driver*, seguida pela verificação do seu *hash* utilizando a função *CryptSIPDllVerifyIndirectData*, de um acesso aos certificados do sistema e do envio de um pacote *UDP* para o servidor do *BattlEye*. Após tal ciclo, são lidos os registros dos serviços que estão executando no sistema (incluindo as aplicações usadas para coleta de dados). Finalmente, o ciclo principal é concluído, fazendo a abertura do disco do sistema para leitura e escrita e capturando as informações *SMART* do dispositivo. Dessa forma, o *anti-cheat* passa a fazer verificações periódicas em intervalos

irregulares de alguns segundos, repetindo o ciclo descrito anteriormente.

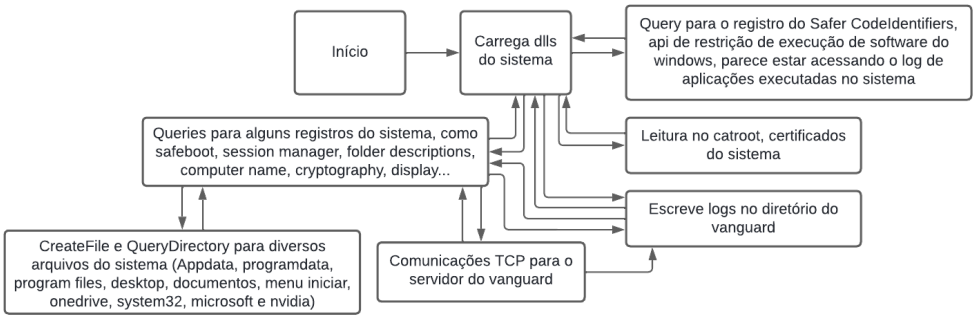

Figura 2. Representação do traço de execução do *Vanguard*.

A Figura 2 mostra o traço de execução observado para o *Vanguard*. A sequência de execução inicia com o carregamento das *DLLs* do sistema, seguido da leitura do registro do *Safer*. Em seguida, o *anti-cheat* carrega *DLLs* e são realizadas leituras no diretório *CatRoot*. Um segundo conjunto de *DLLs* é carregado e consultas são feitas para diversos registros do sistema, incluindo o *computer name* e o *display*, que podem respectivamente fornecer o nome do computador do usuário e informações do monitor. Após, o *anti-cheat* lê diversos diretórios de arquivos do sistema, incluindo *Desktop* e *onedrive*, documentos e diretórios dedicados majoritariamente para armazenamento de arquivos pessoais. Ao fim das verificações, diversos pacotes TCP são enviados para o servidor do *Vanguard* em conjunto com a escrita de novos *logs*. Durante o restante do ciclo de vida, o processo de envio de *queries* para certificados e leitura de arquivos mantém-se como descrito anteriormente.

Através da investigação dos traços de execução, foi possível identificar fatores-chave no processo de execução dos *anti-cheats*. Por exemplo, ambos os *softwares* iniciam sua execução carregando as *DLLs* necessárias e fazendo uma *query* para o registro *CodeIdentifiers* do *Safer*, que é uma API de restrição de execução de *softwares* do *Windows*. As características dessa *query* remetem a configuração do acesso a um arquivo de *log*, permitindo que os *anti-cheats* observem todas as aplicações executadas no sistema.

Foi observado também que as soluções analisadas verificam certificados do sistema, tanto no diretório *catroot* quanto nos registros, escrevendo *logs* em seus diretórios e enviando pacotes de maneira recorrente a servidores próprios. A maioria da execução envolve a verificação dos arquivos do sistema; o *BattlEye* foca apenas nas *DLLs*, realizando verificação de *hash*. Já o *Vanguard*, além das *DLLs*, também faz uma varredura em diversos arquivos do sistema, incluindo diretórios pessoais do usuário.

No fluxo de execução, destaca-se que a privacidade do usuário pode ser impactada nos momentos em que a aplicação troca mensagens com o servidor, já que existe a possibilidade de transmissão de informações sensíveis dos usuários, decorrente do acesso que o *anti-cheat* possui ao sistema. Ambos os *anti-cheats* verificam as *DLLs* do sistema. Em especial, o *BattlEye* realiza essa verificação inúmeras vezes ao longo da sua execução e, após cada verificação, envia um pacote UDP para seu servidor. Essas *DLLs* incluem até mesmo configurações do *Windows Defender*.

O *Vanguard*, além das *DLLs*, também realiza uma varredura em diversos arqui-

vos do sistema, desde diretórios do usuário, como *Desktop*, *Documentos* e *OneDrive*, até diretórios do sistema operacional, como *system32*, *programfiles* e *Appdata*. Em um anúncio oficial da *Riot Games* [Riot 2024a], destaca-se a afirmação de que, assim como outros *anti-cheats*, o *Vanguard* executa uma varredura de assinaturas, buscando dados em memória que sejam iguais aos utilizados por *softwares* de trapaça. Após a varredura executada pelo *Vanguard*, foram observadas diversas comunicações com seu servidor utilizando TCP, semelhante ao comportamento do *BattlEye* após a análise das *DLLs*.

Apesar das mensagens enviadas pela rede estarem criptografadas, foi possível observar que ambos os *anti-cheats* se comunicam com certa frequência com seus respectivos servidores. No caso do *Vanguard*, as mensagens são enviadas para uma CDN na *Cloudflare*, totalizando 16,2 MB de dados comunicados durante 2 minutos de execução. Analogamente, para o *BattlEye*, observamos um processo inicial de comunicação utilizando TCP quando a aplicação é executada, com uma CDN do *BattlEye*, totalizando 16,9 MB. Após essa operação, o *anti-cheat* envia mensagens para um servidor do *BattlEye*, que aparenta ser específico para o jogo sendo executado, gerando um total de 33 KB de dados.

Dentre as *DLLs* usadas pelo *BattlEye*, mas não utilizadas pelo *Vanguard*, destacam-se: *cfgmgr32*, responsável por interagir com os dispositivos de hardware do sistema, como discos, mouses, teclados, câmeras, etc.; *sfc*, responsável pelo verificador de arquivos do sistema do Windows e pode ser utilizada para checar a integridade do sistema e restaurar arquivos corrompidos ou danificados; e *mpr*, dedicada à interação com dispositivos e diretórios compartilhados na rede. Dentre as *DLLs* empregadas pelo *Vanguard*, mas não pelo *BattlEye*, destacam-se: *FWPUCLNT*, que permite a filtragem e inspeção de pacotes de rede em baixo nível, habilitando até o bloqueio de pacotes; *wintrust*, utilizada para verificação e validação de arquivos e dados, validando e verificando assinaturas e certificados; e *wtsapi32*, utilizada para interagir com o *Terminal Services*, uma tecnologia da *Microsoft* que possibilita a conexão remota a computadores e a execução de aplicativos em sessões de usuário remoto.

Já considerando as *DLLs* acessadas por ambos os *anti-cheats*, destacam-se: *ntdll* e *ntmarta*, utilizadas para acessar funções de baixo nível do *kernel* e obter informações sobre o sistema, como hardware, dispositivos e eventos gerados; *rpcrt4*, responsável pelo Protocolo de Chamada de Procedimento Remoto, que permite a comunicação entre processos em computadores diferentes na mesma rede ou até mesmo entre processos em computadores diferentes pela Internet; *wldp*, relacionada ao *Windows Defender Application Control (WDAC)*, sendo usada para restringir os programas e *scripts* que podem ser executados no sistema; e *windows.storage*, responsável pelo armazenamento do sistema e manipulação de arquivos e diretórios. Além disso, ambos os *anti-cheats* também utilizam *DLLs* para manipulação da *shell* do *Windows*, como *SHCore*, *shell32* e *shlwapi*, que possibilitam a execução de diversos comandos e a obtenção de várias informações do sistema.

5.3. Licenças de *Software Anti-Cheat*

Examinando a política de privacidade do *BattlEye*, encontramos informações relacionadas ao processamento de dados pessoais dos usuários. O *BattlEye* declara processar dados pessoais para fornecer seus serviços de prevenção e detecção do uso de *software* fraudulento, visando garantir um ambiente de jogo justo [BattlEye 2024]. É estabelecido que

o principal destinatário dos dados é o próprio *BattlEye*, porém essas informações também podem ser compartilhadas com o editor do jogo monitorado, ou quaisquer parceiros, ou afiliados. Essa mesma política define que os principais dados pessoais processados e coletados pelo *anti-cheat* são: (i) endereço IP; (ii) identificadores do jogo como apelido no jogo e identificador da conta; (iii) informações do *hardware* do dispositivo; (iv) informações do sistema operacional; (v) informações dos arquivos em memória do jogo; (vi) informações sobre os processos, *drivers* e outros códigos executáveis; e (vii) nomes de arquivos que podem conter o nome de usuário.

O *BattlEye* afirma só coletar dados quando necessário, armazenado apenas quando são encontradas evidências de uso de *cheats*, podendo ser retidas por toda a duração do serviço da empresa para o jogo. A empresa também esclarece que, embora precise de acesso total ao sistema para detectar todos os tipos de *cheats*, não busca, transmite ou vende informações pessoais. No entanto, códigos executáveis suspeitos podem ser transmitidos aos servidores para análise posterior quando detectados.

Avaliando o Contrato de Licença do Usuário Final (EULA) do *BattlEye*, é possível encontrar informações adicionais sobre os dados coletados e processados. Nele, é estabelecido que o *anti-cheat* tem permissão para verificar toda a memória das máquinas dos usuários, assim como quaisquer arquivos e diretórios relacionados ao jogo e ao sistema, utilizando algoritmos para identificação de programas fraudulentos. Além disso, a solução pode reportar os resultados desses algoritmos para outros computadores conectados e/ou para o próprio servidor, armazenando essas informações com o único propósito de prevenir e detectar o uso de programas fraudulentos.

Já analisando os termos de serviço da Riot Games, responsável pelo *Vanguard*, é possível encontrar uma seção sobre monitoramento e *anti-cheat*, onde é declarado que a solução pode monitorar ativamente o uso dos serviços da Riot, tanto nos próprios servidores quanto nos computadores ou dispositivos dos usuários, para uma variedade de finalidades, incluindo prevenção de trapaças e redução de comportamento tóxico de jogadores. Ademais, é declarado que, para prevenir trapaças, a empresa pode exigir a instalação de *software anti-cheat*, que pode ser executado em segundo plano nos dispositivos dos usuários.

Ainda, em sua nota de privacidade, a Riot Games informa que, em sua política *anti-cheat*, programas de terceiros que atuam sobre serviços Riot, como *mods*, *hacks*, *cheats*, *scripts* e *bots*, são proibidos. Assim, em caso de detecção de atividade maliciosa, o *anti-cheat* pode tomar decisões automatizadas, como suspensões temporárias ou permanentes de conta, restrições de comunicação, remoção de conteúdo ou acesso limitado ao conteúdo do jogo [Games 2024]. Nesse contexto, a coleta dos seguintes tipos de dados dos usuários é prevista:

- Identificadores que consistem de informações como jogo, nome do usuário, email, telefone, e endereço IP;
- Características que podem ser utilizadas para a classificação, como data de nascimento e gênero;
- Informações comerciais relacionadas a serviços da Riot como histórico de compras e informações de pagamento;
- Atividade na Internet ou em redes eletrônicas, as informações coletadas variam de interações com os serviços da Riot como plataforma de terceiros.

- Dados de geolocalização derivada do endereço IP;
- Informações de áudio e visuais como bate-papo.

Para realizar a tarefa de monitoramento, a Riot Games informa que seus serviços utilizam por exemplo *cookies* para coletar, armazenar e ler arquivos nos dispositivos. Essas tecnologias são também empregadas em seus servidores para coletar informações sobre a interação dos usuários com os serviços da Riot, armazenando-as em arquivos de *log*.

Em uma publicação [Riot 2024a], é afirmado que o *Vanguard* coleta dados estritamente necessários para proteger os jogos, executando uma varredura nos arquivos do sistema e buscando por bytes na memória que correspondam a um *cheat* conhecido, enviando ao seu servidor apenas uma resposta binária, informando se existe ou não uma correspondência e qual programa ou processo a disparou. No entanto, a empresa também alega que algumas detecções necessitam de uma captura instantânea para análise posterior, e estas podem conter informações pessoais identificáveis, como nome de usuário em caminhos de arquivos. Por fim, a empresa declara que utilizam as informações dessas capturas somente para identificar *cheats* e que os dados ficam armazenados por 14 dias.

Apesar das informações nas licenças da Riot Games serem mais abrangentes que as do *BattlEye*, cobrindo não apenas o *anti-cheat Vanguard*, mas todos os diversos serviços prestados, ambas as empresas deixam explícito que o papel de seus *anti-cheats* é detectar trapaças utilizadas nos jogos, garantindo um ambiente justo e competitivo.

Com as informações apresentadas nas licenças dos *anti-cheats* e nas publicações da *Riot Games*, é possível observar uma linha delicada entre a privacidade do usuário e as funcionalidades exercidas pelos *anti-cheats*. A privacidade é respeitada de forma limitada, uma vez que as empresas reconhecem que dados sensíveis dos usuários podem ser capturados, mas apenas em situações suspeitas de fraude. **No entanto, não há uma definição clara e final do que constitui um comportamento suspeito**.

5.4. Discussão

Devido à falta de clareza das operações realizadas por um *software anti-cheat*, e levando em consideração as informações obtidas nas licenças e pela análise dos *anti-cheats* em execução, é possível afirmar que este tipo de aplicação tem acesso ao sistema de arquivos, memória e diretórios pessoais dos usuários; indícios suficientes para afirmar que apresentam potencial ameaça à privacidade do usuário.

O nível de acesso garantido aos *softwares anti-cheat* os coloca como um importante ponto nevrálgico para a segurança do sistema, uma vez que essas soluções podem ser atacadas e/ou comprometidas por entidades maliciosas e falhas na execução. Nesses casos, atacantes podem obter acesso e controle abrangente ao sistema devido aos privilégios concedidos aos *anti-cheats*.

Outro ponto de atenção refere-se às características que definem um *software* como invasivo à privacidade do usuário. Uma definição apropriada consiste em considerar como invasiva toda e qualquer aplicação que coleta e transmite informações que podem ser utilizadas para a identificação de um indivíduo [Boldt and Carlsson 2006], se assemelhando muito ao padrão de operação das soluções *anti-cheat* analisadas neste artigo.

Esse tipo de padrão de operação apresenta similaridades com os padrões percebidos em *spywares*. *Spyware* pode ser definido como um programa instalado e executado clandestinamente em um sistema computacional, monitorando as atividades dos usuários e reportando-as para terceiros. Essencialmente, é um *software* que exerce controle sobre um computador sem o consentimento do usuário. Entre as categorias de *spywares*, destacam-se os *adwares*, *keyloggers* e cavalos de Troia [Stafford and Urbaczewski 2004]:

- *Adware*: *software* que monitora a navegação na Internet, gerando *pop-ups* com anúncios [Gao et al. 2019]. Apesar de intrusivos, *adwares* alteram tipicamente a experiência de navegação do usuário, sem causar maiores problemas operacionais ao sistema infectado;
- *Key Loggers*: *software* que monitora os usuários de um sistema para capturar informações sensíveis, como e-mails, senhas, dados financeiros, entre outros [Bhardwaj and Goundar 2020];
- Cavalos de Troia: *software* aparentemente benigno e de interesse do usuário, mas que contém funções maliciosas ocultas, podendo monitorar e roubar dados dos usuários de um sistema computacional [Chen et al. 2012].

Os *softwares anti-cheat*, apesar de dependerem do consentimento do usuário para a sua instalação e operação, não descrevem em detalhes quais recursos são monitorados nem os algoritmos usados na análise desses recursos. Além disso, após a análise dos traços de execução dos *anti-cheats*, foi possível aferir o nível de acesso que este tipo de aplicação possui, bem como suas características intrusivas que podem impactar diretamente a segurança e a privacidade do usuário. É fato que eles possuem a capacidade de leitura e amplo acesso aos arquivos do sistema dos usuários, além de total liberdade para se comunicar com qualquer servidor na Internet, recebendo e enviando dados arbitrariamente.

Considerando a definição de *spyware* como um *software* que monitora as atividades dos usuários em seus computadores e as reporta para terceiros sem o consentimento dos usuários, os *anti-cheats*, para usuários comuns e leigos que simplesmente jogam um jogo que utiliza uma dessas soluções sem se atentarem às políticas de privacidade, podem apresentar fortes características operacionais de um *spyware*.

Como mencionado anteriormente, os *anti-cheats* têm acesso a diversas informações dos usuários, tornando-se potenciais alvos para entidades maliciosas. No entanto, é igualmente importante destacar que os servidores que armazenam as informações coletadas pelos *anti-cheats* também são alvos significativos para ataques. Essa situação pode resultar no vazamento de dados sensíveis, como nomes, e-mails e senhas de usuários. Embora todos os servidores conectados na Internet estejam sujeitos a ataques e vazamentos, as informações mantidas pelos *anti-cheats* são particularmente comprometedoras, podendo manter qualquer arquivo dos computadores dos usuários, incluindo dados bancários anotados em um simples arquivo de texto.

De maneira geral, os *softwares anti-cheat* são considerados essenciais para preservar a equidade nos jogos, mas conforme indicam os dados coletados, também podem representar um alto custo em termos de privacidade para os usuários. Uma possível solução para esse cenário envolve estabelecer políticas mais transparentes por parte das desenvolvedoras quanto às permissões e limitações de suas soluções. Além disso, considerar a adoção de estratégias menos intrusivas, como realizar a análise das regras de jogo

de forma centralizada em servidores, pode contribuir para manter tanto a privacidade dos jogadores, quanto a justiça nos ambientes de jogo.

6. Trabalhos Relacionados

Ao melhor de nosso conhecimento, ainda não existe um trabalho de avaliação e discussão técnica aprofundada sobre *anti-cheats* na literatura recente. No entanto, existem trabalhos que abordam algumas questões relacionadas aos *softwares anti-cheats*, discutindo, em outras searas, questões de privacidade.

Em [Greidanus 2017], a questão legal da privacidade nos *anti-cheats client-side* é abordada, tratando esse tipo de *software* como um *spyware* em potencial. O trabalho objetiva definir em quais circunstâncias os *anti-cheats client-side* seriam ilegais, analisando a literatura, legislação e jurisprudências, considerando principalmente o Regulamento Geral de Proteção de Dados da União Europeia. Para realizar a análise, o trabalho usa informações especulativas e as informações limitadas existentes na literatura sobre o funcionamento dos *anti-cheats*.

Em [Maario et al. 2021], a questão da invasão de privacidade e os problemas de segurança causados pelos *anti-cheats kernel-level* são considerados, fornecendo opções alternativas de sistemas de detecção, como o espelhamento das regras do jogo no servidor e a análise estatística dos eventos e desempenho dos jogadores. O trabalho aponta que soluções *anti-cheat* instaladas com permissão de *kernel* podem causar problemas até mesmo com atualizações do sistema operacional e, com a estratégia de escanear toda a memória do dispositivo, levantam inúmeras suspeitas de invasão de privacidade dos seus usuários.

Em [Ronkainen 2021], são exploradas as diferenças entre soluções preventivas e detectivas de trapaças em jogos *online*. O estudo expõe diversos problemas de privacidade e segurança contidos nas soluções *anti-cheat*, como a estratégia de escaneamento da memória que verifica informações de outros processos em execução nas máquinas dos usuários. Também é mencionada a capacidade do *BattlEye* de enviar arquivos dos usuários para seus servidores e executar códigos *shell* diretamente nos computadores de maneira remota. O trabalho conclui com a discussão de que uma alternativa preventiva poderia ser projetada e integrada durante o desenvolvimento dos jogos, evitando a necessidade de soluções terceiras e intrusivas de monitoramento e detecção de *cheats*.

7. Conclusão

A popularização do mercado de jogos contribuiu para o surgimento e disseminação de *softwares cheats*, amplamente utilizados por usuários mal-intencionados para obter vantagens indevidas. Em resposta a isso, os *softwares anti-cheat* se tornaram uma escolha comum entre as empresas desenvolvedoras de jogos para detectar e prevenir o uso de *cheats*, garantindo a justiça e a equidade nas condições de jogo.

No entanto, o impacto dos *softwares anti-cheat* na privacidade dos usuários ainda não é totalmente compreendido. Portanto, este trabalho apresenta um estudo focado na análise dos processos de execução de *softwares anti-cheat*, especificamente o *BattlEye* e o *Vanguard*. O estudo revelou que os *anti-cheats*, conforme escopo investigado, apresentam potenciais problemas para a privacidade e segurança dos usuários, devido ao alto nível de acesso e à falta de transparência nas operações realizadas por eles.

Como trabalhos futuros, pretende-se ampliar o escopo de *anti-cheats* analisados para verificar se os comportamentos identificados nas opções estudadas se mantêm nas demais disponíveis. Além disso, há o objetivo de determinar o nível de semelhança operacional dos *anti-cheats* com *softwares spywares* convencionais. Por fim, planeja-se comparar a eficácia e eficiência de estratégias menos intrusivas para combater *cheats* em comparação com os *softwares anti-cheat* tradicionais.

Agradecimentos

Este trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES), UDESC e FAPESC. Os autores também agradecem o Programa de Pós-Graduação em Informática da UFPR e a UTFPR *Campus* Dois Vizinhos.

Referências

Basque-Rice, I. (2023). Cheaters could prosper: An analysis of the security of video game anti-cheat. Honours Project Proposal, School of Design and Informatics, Abertay University. `https://tinyurl.com/mpf9km5z`.

BattlEye (2024). Battleye - the anti-cheat gold standard. `https://www.battleye.com`.

Bhardwaj, A. and Goundar, S. (2020). Keyloggers: silent cyber security weapons. *Network Security*, 2020(2):14–19.

Boldt, M. and Carlsson, B. (2006). Privacy-invasive software and preventive mechanisms. In *International Conference on Systems and Networks Communications*, pages 21–21.

Chen, Z., Roussopoulos, M., Liang, Z., Zhang, Y., Chen, Z., and Delis, A. (2012). Malware characteristics and threats on the internet ecosystem. *Journal of Systems and Software*, 85(7):1650–1672.

DeLap, M., Knutsson, B., Lu, H., Sokolsky, O., Sammapun, U., Lee, I., and Tsarouchis, C. (2004). Is runtime verification applicable to cheat detection? In *Workshop on Network and System Support for Games*, pages 134–138.

Dias Canedo, E., Toffano Seidel Calazans, A., Toffano Seidel Masson, E., Teixeira Costa, P. H., and Lima, F. (2020). Perceptions of ICT practitioners regarding software privacy. *Entropy*, 22(4):429.

Doneda, D. (2006). *Da Privacidade à Proteção de Dados Pessoais*. Revista dos Tribunais, São Paulo, SP, Brasil.

Eung, S., Lui, J. C., Liu, J., and Yan, J. (2006). Detecting cheaters for multiplayer games: Theory, design and implementation. In *Consumer Communications and Networking Conference*, pages 1178–1182.

Games, R. (2024). Riot games privacy notice. `https://tinyurl.com/3r22d3sp`.

Gao, J., Li, L., Kong, P., Bissyandé, T. F., and Klein, J. (2019). Should you consider adware as malware in your study? In *International Conference on Software Analysis, Evolution and Reengineering*, pages 604–608.

Goh, E., Al-Tabbaa, O., and Khan, Z. (2023). Unravelling the complexity of the video game industry: An integrative framework and future research directions. *Telematics and Informatics Reports*, page 100100.

Greidanus, R. (2017). *Client-side Anti-cheat in Online Games: Legal Implications from a Privacy and Data Protection Perspective*. PhD thesis, Tilburg University, Tilburgo, Países Baixos.

Iachello, G., Hong, J., et al. (2007). End-user privacy in human-computer interaction. *Foundations and Trends® in Human–Computer Interaction*, pages 1–137.

Lan, X., Zhang, Y., and Xu, P. (2009). An overview on game cheating and its countermeasures. In *2nd International Symposium on Computer Science and Computational Technology*, page 195, Huangshan, China.

Maario, A., Shukla, V. K., Ambikapathy, A., and Sharma, P. (2021). Redefining the risks of kernel-level anti-cheat in online gaming. In *International Conference on Signal Processing and Integrated Networks*, pages 676–680.

Machado, J. d. M. S. (2014). A expansão do conceito de privacidade e a evolução na tecnologia de informação com o surgimento dos bancos de dados. *Revista da AJURIS*, 41(134).

Maziero, C. (2020). *Sistemas Operacionais: Conceitos e Mecanismos*. Universidade Federal do Paraná, Curitiba, PR, Brasil.

Microsoft (2024). Determine allow-deny list and application inventory for software restriction policies. `https://tinyurl.com/5x4eujft`.

PCGamingWiki (2023). List of games with anti-cheat technology. `https://tinyurl.com/3cwc2ays`.

Pilipovic, S. (2023). Every game with kernel–level anti–cheat software. `https://tinyurl.com/23f2nbn2`.

Pontiroli, S. (2019). The cake is a lie! uncovering the secret world of malware-like cheats in video games. *Virus Bulletin*.

Riot, G. (2024a). /DEV: Vanguard X LoL. `https://tinyurl.com/mrkw7nry`.

Riot, G. (2024b). Vanguard anti-cheat. `https://tinyurl.com/yc3tuvcf`.

Ronkainen, W. (2021). *Prevention vs Detection in Online Game Cheating*. PhD thesis, University of Oulu, Oulu, Finlândia.

Schellens, K. (2021). *Addressing Privacy in Software Architecture*. PhD thesis, Utrecht University, Utreque, Países Baixos.

Shin, D.-H. (2010). The effects of trust, security and privacy in social networking: A security-based approach to understand the pattern of adoption. *Interacting with Computers*, pages 428–438.

Stafford, T. F. and Urbaczewski, A. (2004). Spyware: The ghost in the machine. *The Communications of the Association for Information Systems*, 14(1):49.

Ven, B. (2023). *Cheating and anti-cheat system action impacts on user experience*. PhD thesis, Radbound University, Nimega, Países Baixos.

Wendel, E. (2012). *Cheating in Online Games A Case Study of Bots and Bot-Detection in Browser-Based Multiplayer Games*. PhD thesis, Norwegian University of Science and Technology, Trondheim, Noruega.