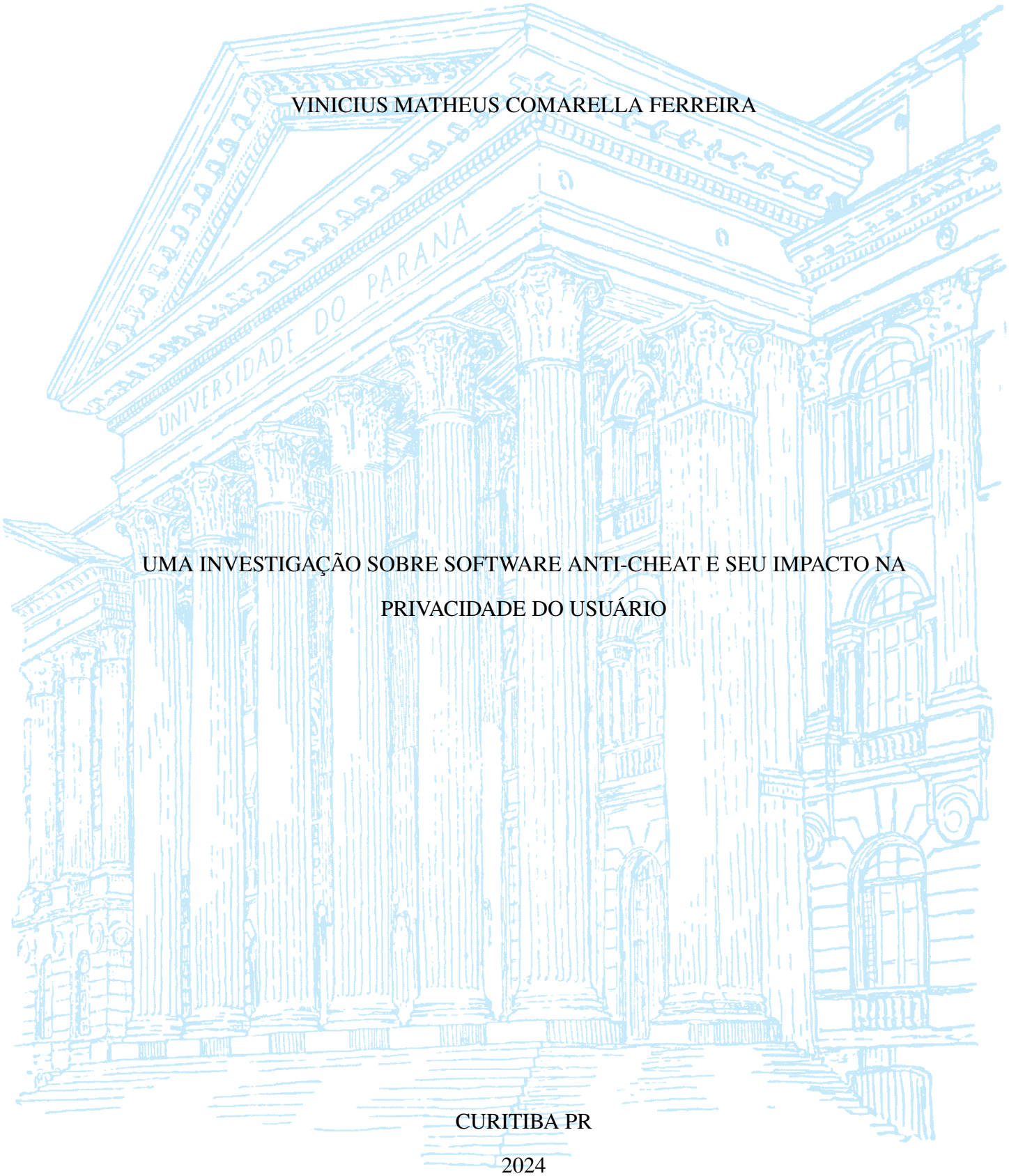
UNIVERSIDADE FEDERAL DO PARANÁ

VINICIUS MATHEUS COMARELLA FERREIRA




# UMA INVESTIGAÇÃO SOBRE SOFTWARE ANTI-CHEAT E SEU IMPACTO NA PRIVACIDADE DO USUÁRIO

CURITIBA PR

2024

VINICIUS MATHEUS COMARELLA FERREIRA

# UMA INVESTIGAÇÃO SOBRE SOFTWARE ANTI-CHEAT E SEU IMPACTO NA PRIVACIDADE DO USUÁRIO

Trabalho apresentado como requisito parcial à conclusão do Curso de Bacharelado em Ciência da Computação, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Carlos Alberto Maziero.

Coorientador: Tiago Heinrich.

CURITIBA PR

2024

# Ficha catalográfica

Substituir o arquivo `0-iniciais/catalografica.pdf` pela ficha catalográfica fornecida pela Biblioteca da UFPR (PDF em formato A4).

**Instruções para obter a ficha catalográfica e fazer o depósito legal da tese/dissertação (contribuição de André Hochuli, abril 2019):**

1. Verificar se está usando a versão mais recente do modelo do PPGInf e atualizar, se for necessário (`https://gitlab.c3sl.ufpr.br/maziero/tese`).
2. conferir o Checklist de formato do Sistema de Bibliotecas da UFPR, em `https://portal.ufpr.br/teses_servicos.html`.
3. Enviar email para "`referencia.bct@gmail.com`" com o arquivo PDF da dissertação/tese, solicitando a respectiva ficha catalográfica.
4. Ao receber a ficha, inseri-la em seu documento (substituir o arquivo `0-iniciais/catalografica.pdf` do diretório do modelo).
5. Emitir a Certidão Negativa (CND) de débito junto a biblioteca (`https://www.portal.ufpr.br/cnd.html`).
6. Avisar a secretaria do PPGInf que você está pronto para o depósito. Eles irão mudar sua titulação no SIGA, o que irá liberar uma opção no SIGA pra você fazer o depósito legal.
7. Acesse o SIGA (`http://www.prppg.ufpr.br/siga`) e preencha com cuidado os dados solicitados para o depósito da tese.
8. Aguarde a confirmação da Biblioteca.
9. Após a aprovação do pedido, informe a secretaria do PPGInf que a dissertação/tese foi depositada pela biblioteca. Será então liberado no SIGA um link para a confirmação dos dados para a emissão do diploma.

# Ficha de aprovação

Substituir o arquivo 0-iniciais/aprovacao.pdf pela ficha de aprovação fornecida pela secretaria do programa, em formato PDF A4.

## AGRADECIMENTOS

Agradeço à minha família, principalmente aos meus pais Anderson e Karina e avós Baltazar e Arlete, por sempre apoiar meus estudos e me incentivar a tomar as decisões corretas na minha vida. Ao meu coorientador Tiago Heinrich por estar presente durante todo o percurso do trabalho, me aconselhando e fornecendo instruções precisas. Ao meu orientador Carlos Alberto Maziero, por ter guiado o desenvolvimento do meu trabalho e também por todos os ensinamentos proporcionados ao decorrer da minha graduação.

Sou muito grato também, à minha namorada e maior companheira Camili que esteve me acompanhando de perto durante todo este processo, me incentivando nos maus momentos e celebrando comigo todas as conquistas. Por fim, agradeço ao Departamento de Informática e à Universidade Federal do Paraná, pela infraestrutura e qualidade de ensino, e aos grandes amigos que a graduação me proporcionou.

## RESUMO

O avanço das técnicas utilizadas nos softwares *anti-cheat*, que são usados para detectar trapaças em jogos eletrônicos, os tornou mais intrusivos nos computadores de seus usuários, possuindo acesso até ao maior nível de permissão nas máquinas, chamado de *kernel-level*. Isso causa uma desconfiança e preocupação dos usuários em relação à privacidade e segurança de seus dados pessoais, não é possível ter certeza de quais dados estão sendo coletados e como estes softwares estão utilizando-os. Este trabalho propõe um estudo prático, capturando e analisando as atividades executadas pelos softwares *anti-cheat*, buscando pontos que afetem a privacidade de seus usuários.



Palavras-chave: Segurança de Computadores, Privacidade e software Anti-Cheat.

## ABSTRACT

The advancement of techniques used in anti-cheat software, which are used to detect cheating in electronic games, has made them more intrusive on their users computers, having access to even the highest level of permission on the machines, called kernel-level. This causes distrust and concern among users regarding the privacy and security of their personal data, as it is not possible to be sure what data is being collected and how these software are using it. This work proposes a practical study, capturing and analyzing the actions made by anti-cheat software, looking for behaviors that affect the privacy of its users.



Keywords: Computer Security, Privacy and Anti-Cheat Software.

## LISTA DE FIGURAS

5.1 Representação do traço de execução do *Battleye*.. . . . . 29
5.2 Representação do traço de execução do *Vanguard*.. . . . . 30

## LISTA DE TABELAS

2.1 Lista de Anti-cheats . . . . . 19

3.1 Trabalhos relacionados . . . . . 24

## LISTA DE ACRÔNIMOS

| | |
|---|---|
| VAC | Valve Anti-Cheat |
| DNS | Domain Name System |
| LGPD | Lei Geral de Proteção de Dados Pessoais |
| GPDR | General Data Protection Regulation |
| GPU | Unidade de Processamento Gráfico |
| CPU | Unidade Central de Processamento |
| ML | Machine Learning |
| DLL | Dynamic Link Libraries |
| WDAC | Windows Defender Application Control |
| EULA | End-User License Agreement |
| RAM | Random Access Memory |
| CDN | Content Delivery Network |

**SUMÁRIO**


**1 INTRODUÇÃO . . . . . 11**
1.1 OBJETIVOS . . . . . 13
1.2 METODOLOGIA . . . . . 13
1.3 ORGANIZAÇÃO TEXTUAL . . . . . 13
**2 FUNDAMENTAÇÃO TEÓRICA . . . . . 14**
2.1 SEGURANÇA & PRIVACIDADE . . . . . 14
2.2 CHEAT SOFTWARE . . . . . 15
2.3 ANTI-CHEAT SOFTWARE . . . . . 17
2.3.1 Server-side Anti-Cheat . . . . . 17
2.3.2 Client-side Anti-Cheat . . . . . 18
2.3.3 Softwares anti-cheat . . . . . 19
2.4 PROBLEMAS E LIMITAÇÕES . . . . . 19
2.5 DISCUSSÃO . . . . . 20
**3 REVISÃO BIBLIOGRÁFICA . . . . . 22**
3.1 PRIVACIDADE EM ANTI-CHEAT . . . . . 22
3.2 CONSIDERAÇÕES . . . . . 23
**4 PROPOSTA . . . . . 25**
4.1 PROBLEMÁTICA . . . . . 25
4.2 ESTRATÉGIA . . . . . 25
4.2.1 Ferramentas de Coleta . . . . . 26
4.2.2 Métodos de avaliação . . . . . 26
**5 AVALIAÇÃO DE RESULTADOS . . . . . 27**
5.1 ESTRATÉGIA DE AVALIAÇÃO . . . . . 27
5.2 ANÁLISE DE DADOS . . . . . 28
5.3 LICENÇAS DE SOFTWARE ANTI-CHEAT . . . . . 32
5.4 DISCUSSÃO . . . . . 35
**6 CONCLUSÃO . . . . . 38**
**REFERÊNCIAS . . . . . 40**

# 1 INTRODUÇÃO

A popularidade de jogos eletrônicos, principalmente multijogadores, trouxe consigo jogadores que possuem o desejo de ganhar a qualquer custo e recorrem a trapaças para obter vantagem de maneira desonesta. Existe uma diversa gama de trapaças em jogos, desde as inofensivas, definidas e implementadas pelos desenvolvedores do jogo, as quais geralmente exigem uma combinação de teclas a serem pressionadas e com isso algum recurso é habilitado para facilitar a jogabilidade, até as mais agressivas, como softwares chamados de *hacks* ou *cheats*, utilizados por usuários mal intencionados com o fim de manipular o comportamento do jogo em sua máquina para benefício próprio, estes softwares são capazes de alterar propriedades fundamentais do jogo fornecendo diversas vantagens aos jogadores, é possível obter benefícios como visão dos inimigos pelas paredes, mira automática, vida infinita, e assim por diante. Os usuários deste tipo de software são comumente chamados de *cheaters*.

A comunidade de jogos eletrônicos repudia os consumidores de *cheats*, já que jogadores honestos constantemente perdem suas partidas jogando sob essa desvantagem, levando-os a se sentir frustrados e impotentes. Todos os jogos que atingem uma base ativa de jogadores inevitavelmente serão alvo de *cheats*, dessa maneira, existe uma pressão nos desenvolvedores dos jogos a evitar essas trapaças, já que a frustração dos jogadores honestos a longo prazo leva a diminuição no número de pessoas jogando, causando prejuízos monetários para as empresas (Maario et al., 2021).

Um agravante que instiga o uso de *cheats* está relacionado a valores monetários, onde moedas virtuais estão vinculadas a valores reais. Jogos que possuem este tipo moedas acabam dando aos jogadores uma motivação a mais para trapacear, obter dinheiro real de maneira fácil através dele. O *duping* é uma maneira desonesta de duplicação de itens, um exemplo seria no caso de transação de moedas, se o remetente se desconectar durante a operação, em jogos sem a devida proteção pode ocorrer do valor ser entregue ao destinatário porém não subtraído do remetente, duplicando a quantidade de moedas (DeLap et al., 2004). O caso de Noah Burn (Greidanus, 2017) é um exemplo onde um jogador explorou a duplicação de itens para ganhos monetários em moedas reais, uma empresa que desenvolve e mantém jogos com esse tipo de moedas deve redobrar a atenção para a prevenção de *cheats*, já que problemas com trapaças envolvendo moedas virtuais já ocorreram antes e causa não só a perda de jogadores, mas também de dinheiro diretamente.

Diferentes estratégias são utilizadas para a detecção de *cheaters* nos jogos. Algumas destas são (i) coleta de denuncias de jogadores dentro do jogo; (ii) uso de software especializado nos servidores do jogo; (iii) uso de software especializado instalado na máquina dos usuários.

A primeira, exige pessoas especializadas constantemente analisando estas denuncias, verificando e confirmando se realmente são válidas, esta é uma tática que pode funcionar, mas é pouco escalável e trabalhosa (Eung et al., 2006), e acaba sendo utilizada apenas como um complemento para outras estratégias.

As outras duas estratégias são as soluções mais comuns, comumente chamadas de *anti-cheat*. Os softwares da segunda estratégia são chamados de *server-side anti-cheats*, eles funcionam juntamente com os servidores dos jogos, apenas monitorando o comportamento recebido do jogador pelo servidor, buscando ações inválidas e inconsistências. Apesar de serem bem protegidos contra alterações por *cheaters*, já que permanecem fora das máquinas dos usuários, estes são bem limitados, pois só possuem acesso à informação enviada pelos jogadores. Sua taxa de detecção tende a ser menor, pois é improvável que este tipo de *anti-cheat* seja capaz

de identificar trapaças que não alteram explicitamente o comportamento do jogador, por exemplo alterações de elementos visuais, como remoção de paredes mantendo a sua colisão. Esta solução tem um custo considerável conforme a quantidade de jogadores, pois depende de recursos dos servidores do jogo para se manter funcionando (Greidanus, 2017).

A terceira estratégia são os softwares chamados *client-side anti-cheat*, estes são instalados diretamente nas máquinas dos jogadores, monitorando e coletando informações de todo o sistema. Esta solução consegue detectar uma gama maior de trapaças, porém ao contrário dos *server-side*, consome recursos dos computadores dos jogadores e acaba sendo vulnerável a alterações maliciosas feitas por *cheaters*, já que está diretamente no computador do usuário, podendo ser analisada e burlada (Ven, 2023).

O *anti-cheat* não é um software simples de ser desenvolvido, exige o conhecimento de todos os tipos de *cheats* e os aspectos técnicos de como são feitos, além de anos de desenvolvimento para que os métodos de detecção sejam elaborados, testados e aprimorados. Existem softwares disponíveis no mercado para que os desenvolvedores de jogos comprem e integrem facilmente a qualquer tipo de jogo, estes fornecem detecção aos mais diversos tipos de *cheats* e várias opções de punições aos trapaceiros (Maario et al., 2021). Um forte nome no mercado atualmente é o BattlEye, um *anti-cheat client-side* que também é integrado aos servidores do jogo para evitar que os *cheaters* banidos continuem jogando, ele é utilizado em diversos jogos populares como DayZ e ARMA 3 (BattlEye, 2024).

Uma disputa ocorre entre os desenvolvedores de *cheats* e *anti-cheats*, enquanto os responsáveis pelos *anti-cheats* buscam maneiras de detectar todas as formas mais sofisticadas de *cheats*, os *cheaters* tentam burlar estas detecções, utilizando software especializado em inserção de código nos jogos, e uma grande rede de fórums engajada em manter os *cheats* indetectáveis (Maario et al., 2021).

Visando aperfeiçoar a detecção de usuários trapaceiros, os *anti-cheats* estão cada vez mais intrusivos, inclusive sendo instalados juntamente ao sistema operacional, com permissão total no dispositivo, onde são executados junto ao sistema mesmo que não exista a execução de um jogo. Estes tornaram suas ações obscuras, para dificultar ao máximo o trabalho de engenharia reversa feita pelos desenvolvedores de *cheats*, evitando a descoberta de pontos fracos que poderiam ser explorados por trapaças (Greidanus, 2017).

Para uma análise mais detalhada, alguns *anti-cheats* adotaram a estratégia de enviar arquivos das máquinas dos jogadores para verificação remota no servidor. Todos os fatores citados causam uma grande preocupação com a privacidade dos usuários. Não é possível saber exatamente quais informações estão sendo coletadas dos usuários, mesmo que seja declarado nos termos de uso que nenhuma informação não relacionada com trapaças será coletada e que nenhuma informação dos jogadores será vendida, como algumas empresas de softwares *anti-cheat* fazem, ainda assim existe a possibilidade que um ataque malicioso comprometa o servidor do *anti-cheat* e consequentemente qualquer informação contida dos usuários seriam vazadas (Ronkainen, 2021).

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 acabavam fechando todos as outras aplicações do seu computador, alguns até pararam de jogar por completo. Em (Pontiroli, 2019) é apontado que alguns *anti-cheats* já impediram jogadores de se conectar a servidores apenas por terem usado a palavra-chave “cheat” em navegadores web, e até o *Valve Anti-Cheat* (VAC) (Corporation, 2024), que é uma solução *anti-cheat* da *Valve*, muito conhecida no mercado, já foi acusado de inspecionar requisições *Domain Name System* (DNS) dos usuários, que são feitas para converter nomes em endereços IP, que são usados para identificação e

endereçamento de dispositivos em redes, isso indica problemas de privacidade, já que é possível identificar sites acessados e aplicações utilizadas pelos usuário.

Estes softwares intrusivos também afetam a estabilidade e segurança dos sistemas operacionais de seus usuários, como estes *softwares* tem permissão total no dispositivo, qualquer vulnerabilidade contida em seu código, se explorada por um atacante, poderá levar ao acesso e controle total do dispositivo, gerando uma enorme preocupação em segurança para os usuários (Basque-Rice, 2023). Já houveram casos de instabilidade causada por esses *anti-cheats*, como no caso do Vanguard criado pela Riot Games, principalmente no início de seu desenvolvimento, incidentes foram reportados, de *softwares* e *drivers* básicos de teclado e mouse sendo caracterizados como trapaça e consequentemente periféricos não funcionando, até *drivers* fundamentais para o funcionamento do computador sendo interrompidos, como de refrigeração dos componentes, causando sobreaquecimento e possível danos (Ven, 2023).

O objetivo deste trabalho de conclusão de curso é uma análise das ações executadas pelos *softwares anti-cheat* no sistema, monitorando os diretórios acessados, suas permissões, comunicações na rede e consultas e alterações nos registros, buscando pontos que afetem a privacidade do usuário.

## 1.1 OBJETIVOS

**Objetivo geral:** Realizar uma análise de softwares *anti-cheat* no sistema, monitorando os diretórios acessados, suas permissões, comunicações na rede e consultas e alterações nos registros, buscando pontos que afetem a privacidade do usuário.

**Objetivos específicos:**

- Investigação no impacto da privacidade do usuário que softwares *anti-cheat* possuem;
- As interações realizadas e recursos acessados por softwares *anti-cheat*;
- Uma discussão do contexto atual que este formato de software impacta a segurança final do usuário.

## 1.2 METODOLOGIA

Este trabalho de conclusão de curso consiste em uma pesquisa aderindo o método de estudo de caso. Primeiramente uma apuração foi realizada para identificar ferramentas e ambientes capazes de monitorar ações relevantes dos *anti-cheats*, como arquivos acessados e permissões adquiridas durante sua execução. Para uma posterior investigação nos *anti-cheats* Vanguard e BattlEye, buscando pontos que prejudicassem a privacidade dos usuários finais.

## 1.3 ORGANIZAÇÃO TEXTUAL

Neste trabalho, há uma divisão em seis partes. O capítulo 2 aborda a fundamentação teórica, expondo os conceitos de segurança e privacidade, *cheats*, *anti-cheats*, seus modelos, problemas e limitações. O capítulo 3 realiza uma revisão bibliográfica da literatura existente sobre privacidade nos *anti-cheats*. No capítulo 4, é apresentada a proposta, descrevendo a problemática, qual a estratégia adotada para a investigação, as ferramentas utilizadas para a coleta dos dados e os métodos de avaliação dos resultados. O capítulo 5 apresenta os dados obtidos na coleta e a avaliação dos resultados. Por fim, o capítulo 6 traz a conclusão, considerações finais deste trabalho e também aponta perspectivas para pesquisas futuras.

# 2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo introduz a fundamentação teórica necessária para a compreensão do problema apresentado no trabalho e dos experimentos e resultados obtidos. A seção 2.1 introduz os conceitos básicos de segurança e privacidade em software; A seção 2.2 descreve o que é um *cheat*, suas variedades e como são feitos; A seção 2.3 apresenta a definição, tipos e técnicas utilizadas pelos softwares *anti-cheat*; A seção 2.4 mostra quais são os problemas e limitações existentes nos *anti-cheats* atualmente e a seção 2.5 discute como estes problemas listados afetam os usuários deste tipo de software.

## 2.1 SEGURANÇA & PRIVACIDADE

Na área de computação e desenvolvimento de software, a segurança contra ameaças intencionais possui algumas propriedades fundamentais: confidencialidade, integridade e disponibilidade, também são muitas vezes utilizadas a autenticidade e irretratabilidade (Maziero, 2020).

- Confidencialidade: se refere a prevenção do vazamento, intencional ou não, dos dados da aplicação. Ou seja, os dados confidenciais, que não devem ser de acesso público, devem estar disponíveis apenas para as partes autorizadas. Nenhuma outra pessoa ou software deve ser capaz de acessar e obter estas informações (Agarwal e Agarwal, 2011).

- Integridade: consiste em garantir que nenhuma modificação nas informações ou dados serão feitas por processos ou pessoas não autorizadas, da mesma forma que garantir que pessoas e processos autorizados não farão modificações não autorizadas, além de também certificar que os dados estão consistentes em todas as partes do software ou sistema (Banerjee e Pandey, 2009).

- Disponibilidade: é a capacidade do software estar disponível para fornecer o serviço correto. Ou seja, um sistema ou componente deve estar pronto e funcional, com a capacidade de executar sua tarefa corretamente sempre que necessário (Khan e Khan, 2012).

- Autenticidade: é o atributo de ser autêntico ou de autoridade estabelecida. Todas as entidades, ações e dados de um sistema são verdadeiros e correspondem às informações do mundo real que elas representam (Maziero, 2020).

- Irretratabilidade: este também é chamado de não repúdio; Deve ser possível saber o responsável por cada ação e dado inserido em um sistema ou aplicação, sem que essa pessoa possa negar ter feito (Buja, 2021).

Estes são os 5 pilares do modelo de garantia de informações definido pelo departamento de defesa dos Estados Unidos em 2002 (Moghaddasi et al., 2016). Em termos gerais, podemos considerar uma aplicação ou software seguro em relação a ataques de usuários maliciosos se ele garante o cumprimento destes princípios.

A privacidade é um conceito antigo. Um artigo considerado como o início da discussão sobre este assunto foi publicado em 1890 por Judges Warren and Brandeis. Nele a privacidade é definida como o direito de ser deixado sozinho: os indivíduos devem ser protegidos de

publicações de qualquer detalhe das suas vidas que consideram confidenciais (Doneda, 2006; Iachello et al., 2007). Hoje, ela é considerada um direito fundamental, presente na Declaração Universal dos Direitos Humanos (Doneda, 2006).

Com o aprimoramento da tecnologia e surgimento da Internet, a privacidade se tornou um recursos fundamental (Machado, 2014). Em computação e desenvolvimento de *software*, é discutida a privacidade da informação. Esta área é definida como a medida de quanto as pessoas decidem quais informações são coletadas sobre elas, ou então como a redução do risco de informações pessoais serem usadas de maneira indevida (Schellens, 2021).

Em (Shin, 2010) é comentado que a privacidade é regularmente confundida com segurança nos *softwares*. Isso acontece por que os problemas de segurança também causam falhas na privacidade, fazendo com que esses dois conceitos andem lado a lado. Uma falha de confidencialidade inevitavelmente levará a exposição de dados pessoais dos usuários, e consequentemente a privacidade desses usuários não será respeitada.

Algumas soluções de integração de privacidade em sistemas de *software* utilizam como base os princípios da privacidade por *design* criados por Ann Cavoukian, que destaca a privacidade e transparência. Todos os componentes e operações devem estar visíveis e transparentes para os usuários. Outro princípio é o respeito pela privacidade do usuário. Os arquitetos e desenvolvedores de *software* devem atender os interesses dos usuários, fornecendo soluções para que apenas as informações que o usuário selecionar sejam armazenadas e/ou processadas (Baldassarre et al., 2020).

Países possuem leis que regulamentam a privacidade no desenvolvimento de *software*. No Brasil existe a Lei Geral de Proteção de Dados Pessoais (LGPD), que é semelhante a *General Data Protection Regulation* (GPDR) que está em vigor na União Europeia. Essas leis definem regras que devem ser seguidas 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 usadas as informações dos clientes, e os usuários devem ser capazes 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. Ele é capaz de fornecer benefícios aos seus usuários de maneira ilegitima, facilitando a jogabilidade e até fornecendo informações privilegiadas e a permissão de execução de ações que deveriam ser restritas (Ronkainen, 2021).

Estes *softwares* podem ser divididos em diversos tipos, como por exemplo aqueles que não afetam o comportamento fundamental do jogo, como *bots* e macros. *Bots* são *cheats* que fazem ações automáticas dentro dos jogos, permitindo que o jogador se ausente de seu computador, e mesmo assim continue fazendo ações. Estes geralmente são usados para adquirir recursos automaticamente durante longos períodos de tempo, sem a necessidade de interações do usuário com o jogando, fazendo com que os jogadores progridam, já que podem se manter "jogando" 24 horas por dia.

Macros permitem definir sequências customizadas de teclas a serem pressionadas. Assim, ao pressionar algum botão definido, toda uma sequência de teclas pode ser pressionada automaticamente, facilitando a jogabilidade em alguns jogos que necessitam de uma grande quantidade de teclas sendo pressionadas simultaneamente, ou que necessitam de uma sequência específica de comandos que deve ser pressionada rapidamente para realizar operações essenciais no jogo (Matthies et al., 2015).

*Softwares* de *cheats* também podem afetar diretamente as regras fundamentais dos jogos, modificando suas leis ou realizando ações essenciais de maneira automática, fornecendo vantagens significativas aos *cheaters*. Estes *cheats* podem realizar uma injeção de código malicioso durante a execução do jogo, alterando as informações em memória para modificar o seu comportamento, ou então capturando dados do jogo durante sua execução para assim realizar ações perfeitas para o usuário (Ronkainen, 2021). Alguns exemplos deste tipo de *cheat* e suas vantagens são:

- *Aimbot*: Popular em *tactical first-person shooter*, fornece ao jogador uma mira automática, eliminando a necessidade de movimentar o mouse até o seu oponente. Os mais avançados possuem opções para serem ligados e desligados rapidamente através de teclas e até sua velocidade e posição da mira são ajustáveis para evitar detecção (Ven, 2023). Este resultado pode ser alcançado através da captura da posição dos inimigos, tanto diretamente pela memória do jogo, como por interceptação dos quadros diretamente da Unidade de Processamento Gráfico (GPU), responsável pelo processamento e renderização das imagens na tela. Com a posição obtida, é enviada a ação perfeita para o jogo, mimetizando uma ação feita pelo mouse (Wendel, 2012);

- *Triggerbot*: *Triggerbots* são usados para atirar automaticamente assim que a mira do jogador está em um oponente, desta forma, são difíceis de serem detectados, já que o jogador esta realizando entradas esperadas pelo jogo, sem uma mira perfeita (Khalifa, 2016). Este tipo pode ser realizado de maneira semelhante aos *aimbots*, mas em vez de mover o mouse, executa a ação de atirar no momento correto;

- *Wallhacks* e *Maphacks*: Nos jogos onde certas áreas do mapa não devem ser exibidas para todos os jogadores. Os *maphacks* fornecem essa informação aos *cheaters*, da mesma maneira, principalmente nos jogos de tiro em primeira pessoa, os *wallhacks* mostram a posição dos inimigos através das paredes, conferindo vantagens (Joshi, 2008). O que possibilita a criação de um *Wallhack* é o fato do servidor do jogo enviar a posição de todos os jogadores, e a decisão de apresentar esta informação na tela ser feita localmente, no jogo do usuário no momento em que é renderizado (Wendel, 2012). Uma alteração na região de memória ou injetando código malicioso permite alterar o comportamento da renderização e mostrar os jogadores através das paredes;

- *No Recoil*: Nos jogos de tiro, as armas possuem um certo recuo, fazendo com que a mira dos jogadores, ao atirar, se mova, tornando difícil o seu controle durante uma rajada. O *no recoil* faz com que este recuo seja anulado automaticamente, facilitando a jogabilidade (Bajaj, 2022). Isso também pode ser realizado através da modificação do código do jogo (Wendel, 2012);

- *Speedhack*: Este tipo de *cheat* faz com que o jogador tenha uma vantagem de velocidade, tornando seu personagem mais rápido, o que pode ser facilmente detectado no lado do servidor (Telimaa, 2021). Esse resultado pode ser atingido alterando a memória do jogo, trocando o valor da variável responsável por contar o tempo, afetando a duração dos quadros e consequentemente das animações e velocidade do jogador (Telimaa, 2021); e

- *Lag Switches*: *Lag Switches* podem ser ativados para parar de enviar ações para o jogo, enquanto ainda recebe as ações dos outros jogadores, ganhando informação de posição e possibilitando ter vantagens em confrontos (Willett e Hogan, 2019). Para obter esse resultado, pode ser feita uma alteração na placa de rede, possibilitando o bloqueio do envio e apenas a recepção de pacotes (Webb e Soh, 2007).

Para desenvolver este tipo de *software* é necessário o conhecimento em engenharia reversa, possibilitando o reconhecimento das vulnerabilidades tanto do jogo, para a descoberta de fragilidades passíveis de exploração para o fornecimento de vantagens, como dos *anti-cheats*, para identificação de pontos fracos na detecção, evitando que seus *cheats* sejam encontrados e os *cheaters* banidos (Webb e Soh, 2007).

*Cheat softwares* são adquiridos pelos usuários principalmente através da compra do produto, tanto de uma solução específica para algum jogo, como de uma assinatura mensal que libera diversos *cheats* para diversos jogos. Existe um mercado deste tipo de *software* na Internet. (Wendel, 2012) mostra uma lista de algum dos sites existentes focados no desenvolvimento e comercialização destas soluções, até menciona um site que faz a avaliação dos melhores fornecedores baseado em preço, facilidade do seu uso, qualidade do serviço ao consumidor, comunidade e taxa de detecção, mostrando que existe uma grande popularidade neste nicho.

## 2.3 ANTI-CHEAT SOFTWARE

Anti-cheat é um tipo de *software* criado especificamente para detectar todos os tipo de *cheats* citados e punir os *cheaters* (Ven, 2023), aplicando desde restrições na jogabilidade até o total banimento do jogador, impossibilitando que este jogue novamente em sua conta (Lan et al., 2009). Todos os *anti-cheats* guardam uma lista possuindo todos os jogadores que já utilizaram *cheats* em seus jogos. Alguns *anti-cheats* são mais agressivos, e quando detectam o uso, banem o jogador em todos os jogos que utilizam o mesmo *anti-cheat* (Wendel, 2012).

Os *anti-cheats* utilizam várias estratégias para tentar impedir os jogadores de trapacear em seus jogos, desde reconhecimento de padrões recebidos pelo servidor, verificando se o jogador não tem picos absurdos de jogabilidade em alguns momentos, ou se ações inconsistentes e não permitidas foram enviadas, até a análise de integridade dos arquivos e escaneamento de memória ao decorrer da execução dos jogos na máquina dos usuários (Maario et al., 2021).

Os *anti-cheats* podem ser divididos em duas grandes categorias, definidas por onde eles farão a sua análise e detecção de trapaças. Os chamados *server-side* são executados apenas no lado do servidor, e os *client-side* são instalados em todas as máquinas dos jogadores, sendo iniciados assim que o jogo é aberto, e alguns até antes da abertura.

### 2.3.1 Server-side Anti-Cheat

Este tipo de *anti-cheat* é feito totalmente pelo lado do servidor, funcionando de maneira que o servidor define as regras do jogo, ou, no mínimo, elas são replicadas também nele. Dessa maneira, cada ação executada pelos jogadores e cada estado de jogo tem sua validade verificada, evitando que modificações e injeções de código nos jogos dos jogadores sejam capazes de alterar uma regra pré-estabelecida (Maario et al., 2021).

(Wendel, 2012) afirma que jogos que possuem a sua lógica definida pelo servidor são praticamente imunes a *cheats*, já que o jogo que está no computador dos usuários apenas envia as informações das teclas pressionadas ao servidor, sendo impossível alterar a lógica pelos arquivos ou memória. Isso é um ponto positivo dos *anti-cheats server-side*. Também seguindo esta ideia de verificação no lado do servidor, existem soluções que se baseiam no estudo dos *logs* de eventos e ações dos jogos coletadas pelo servidor, como cada jogador pode gerar uma grande quantidade de *logs* estas soluções utilizam técnicas de inteligencia artificial, como *machine learning* (ML) (Jonnalagadda et al., 2021).

Estas soluções extraem e analisam informações da jogabilidade de cada jogador, como tempo para que o jogador mate um inimigo na mira, sua precisão, média de mudança da sua

câmera em jogo, entre outras informações importantes para identificar um *cheater*. Estas informações são comparadas e, se reconhecido que um jogador honesto não conseguiria obter os mesmos resultados, então é classificado como um *cheater* (Alayed et al., 2013).

Entretanto, existem alguns pontos negativos a esse tipo de *anti-cheat*. Por sua natureza, verificam todas as ações de todos os jogadores no servidor. Assim, geram um alto custo computacional, necessitando de *hardwares* poderosos nos servidores para que esta solução seja efetiva. Da mesma maneira, por ser limitada ao escopo do servidor, ou seja, das ações que os jogadores enviam, as trapaças sutis, principalmente as visuais, são dificilmente detectáveis. Por exemplo, um *wallhack* não depende do envio de nenhuma informação adulterada ao jogo, apenas modifica o jogo local do *cheater*, mostrando a posição dos jogadores através da parede (Ronkainen, 2021).

### 2.3.2 Client-side Anti-Cheat

*Anti-cheats client-side* são instalados diretamente nas máquinas dos jogadores. Dessa forma, possuem um acesso maior à informação, analisando a memória dos computadores e podendo detectar até os *cheats* mais sutis. Estes também liberam recursos dos servidores, já que por estar diretamente nas máquinas dos clientes utilizam seus recursos, porém também acabam ficando vulneráveis ao estudo, alterações e injeção de código pelos *cheaters* (Ven, 2023).

Este tipo de *anti-cheat* usa diversas técnicas para detectar e evitar trapaças. Uma das dessas maneiras é a proteção dos dados, ou seja, uma tentativa de evitar que os *cheats* consigam identificar e alterar pontos críticos na memória e binários dos jogos. Para isso, utilizam criptografia tanto nos dados em memória, como nos pacotes recebidos pela rede, evitando que os desenvolvedores de *cheats* identifiquem onde os módulos suscetíveis a modificações que forneceriam vantagens estão localizados (Lan et al., 2009).

Uma estratégia para evitar alteração de código é a utilização de algoritmos de verificação de integridade. Assim que os jogadores se conectam ao servidor é verificado se todos os binários de seus jogos correspondem aos da última versão oficial, garantindo que nenhum foi alterado. Um algoritmo comum para isso é o *checksum*. Alguns *anti-cheats* também recorrem a soluções como tirar fotos da tela do usuário para analisar em seus servidores, sendo possível detectar alguns *cheats* visuais (Matthies et al., 2015).

É possível utilizar a detecção de padrões no sistema de arquivos dos jogadores, buscando formas conhecidas de *cheats*, sendo possível detectar formas menos perceptíveis de trapaças. Para isso, é necessário verificar todo o sistema de arquivos dos usuários e manter atualizada uma base de dados com todas as formas de *cheats* mais recentes. Essa técnica possuí um custo alto de processamento e exige um esforço constante de estudo e atualização (Maario et al., 2021).

Em (Feng et al., 2008) é apresentado que uma série de medições no processo do jogo podem ser bastante efetivas para detectar *cheats*. Estão inclusas medições de registradores da *Unidade Central de Processamento* (CPU) e comportamento de execução das chamadas de sistema. Como as injeções e modificações de código geram uma alteração nas instruções e nas chamadas de sistema que ocorrem durante o jogo, é possível detectar *cheats* através da medição frequente. De maneira semelhante, também podem ser executadas medições na memória, para verificar se algum código malicioso foi injetado, verificar a pilha de execução analisando se os endereços de retorno estão dentro do escopo das funções do jogo, além de medições nos sistema de arquivos e processos externos, buscando padrões de *cheats* conhecidos. Essas são invasivas, já que monitoram todos arquivos e pastas do computador dos usuários e verificam o espaço de memória de outras aplicações.

Ultimamente, alguns *anti-cheats* estão trabalhando no nível de *Kernel*, ou seja, no mesmo nível de privilégio que os sistemas operacionais, tendo acesso e permissão total na

máquina de todos os usuários e podendo monitorar desde processos básicos em execução, como navegadores de internet, até *drivers* essenciais para o funcionamento da máquina, como de controle de CPU e refrigeração. Estes possuem mais capacidade de detecção, já que recebem permissão e acesso total a tudo nas máquinas, além de possuírem liberdade total, já que não precisam fazer chamadas para o sistema operacional para adquirir permissões como criar e ler arquivos, acessar dispositivos e interfaces, controlar processos, como as outras aplicações necessitam (Greidanus, 2017).

### 2.3.3 Softwares anti-cheat

Atualmente, existem soluções *anti-cheat* sendo vendidas e empregadas nos jogos, tanto do tipo *server-side*, como *client-side* e até mesclas das duas formas. Uma lista de *anti-cheat* é apresentada na Tabela 2.1 (Pilipovic, 2023; PCGamingWiki, 2023). Estas mesclas se beneficiam das vantagens das duas vertentes, mas precisam lidar com todas as desvantagens.

| **Anti-cheat** | **Desenvolvedora** | **Possui permissão de kernel?** |
|---|---|---|
| Easy Anti-Cheat | Kamu | Sim |
| BattlEye | BattlEye Innovations e.K. | Sim |
| XIGNCODE3 | Wellbia | Sim |
| PunkBuster | Even Balance | Sim |
| Valve Anti-Cheat | Valve | Não |
| nProtect GameGuard | INCA Internet | Sim |
| Ricochet | Activision | Sim |
| Vanguard | Riot Games | Sim |
| FaceIT Anti-Cheat | FACEIT | Sim |
| ESEA Anti-Cheat | ESEA | Sim |

Tabela 2.1: Lista de Anti-cheats

Estes *softwares* possuem características distintas. O *Valve Anti-Cheat* opta pela utilização de *deep learning* para detectar as trapaças, que é uma técnica de inteligência artificial, quando detectadas recorrem a uma avaliação dos jogadores experientes no jogo para julgar se foi correta ou não, ajustando o seu algoritmo. Esta implementação acaba detectando apenas os *cheats* mais aparentes, que extrapolam os limites da capacidade de jogabilidade dos seres humanos (Lehtonen et al., 2020). Existem soluções *anti-cheat* que recorrem a análise de dados e estatística como o *Easy Anti-cheat*, que tem parte de sua execução na nuvem e o resto na máquina dos usuários; Ele também recorre a permissão de *kernel* para sua detecção (Lehtonen et al., 2020). O *BattlEye*, além de trabalhar em modo *kernel*, também envia arquivos para o seu servidor e possui a capacidade de rodar códigos *shell* de maneira remota no computador de seus usuário (Ronkainen, 2021). O Vanguard possui a peculiaridade de ser iniciado junto com o sistema operacional, permanecendo ativo mesmo que nenhum jogo esteja aberto no momento (Telimaa, 2021).

## 2.4 PROBLEMAS E LIMITAÇÕES

Existem *anti-cheats* que para seu funcionamento são executados com permissão de *kernel*. As principais preocupações dos usuários em relação a isso são violação de sua privacidade

e a instabilidade que pode ser causada no sistema e segurança como um todo, já que qualquer vulnerabilidade contida nestes *softwares* podem fornecer o acesso ao *kernel* para um atacante (Basque-Rice, 2023). Um agravante nestas preocupações é que alguns *softwares anti-cheat* são iniciados antes de qualquer jogo ser aberto, como o Vanguard, que é executado assim que o computador do usuário é ligado, e permanece ativo a partir de então, o que é invasivo (Ven, 2023).

Empresas que oferecem soluções de *anti-cheat* utilizam um termo de consentimento que os jogadores devem aceitar assim que seu *software* é instalado. É comum estes termos informarem quais serão as ações executadas e quais informações serão coletadas. Nestes termos, é recorrente o aviso que apenas serão adquiridos dados fundamentais para a detecção de trapaças. O problema é que isto não é o suficiente para garantir a privacidade dos usuários: esta relação depende de uma transparência do *anti-cheat*, o que é difícil, já que não é possível observar facilmente quais ações estão sendo feitas por estes softwares a qualquer momento de execução. Assim, os usuários acabam tendo que confiar apenas no que foi acordado na licença (Greidanus, 2017).

Com a utilização de *anti-cheats*, já foram registrados casos de invasão de privacidade, como o caso do *VAC*, *anti-cheat* da *Valve* utilizado em diversos jogos. Este foi encontrado acessando o histórico de pesquisas dos usuários (Maario et al., 2021). Também já houveram casos de falha segurança, como o *anti-cheat* utilizado no jogo Genshin Impact, mhyprot2.sys. Ele possui uma vulnerabilidade que possibilita atacantes desabilitar softwares de *antivirus*, viabilizando todos os tipos de ataques (Basque-Rice, 2023).

Já houveram diversos casos de instabilidade gerados por softwares *anti-cheats*, principalmente os com permissões de *kernel*. Diversos problemas ocorreram com a introdução do *Vanguard*, *anti-cheat* criado pela *Riot Games*. Como ele possui permissões para encerrar qualquer processo, diversos casos de *drivers* de mouse e teclado já foram considerados trapaças e encerrados, levando a estes dispositivos não funcionarem corretamente. Até mesmo *drivers* de refrigeração já foram afetados, levando ao sobreaquecimento de componentes dos computadores dos usuários (Ven, 2023). Fora que aplicações no nível de *kernel*, se falharem, também fazem todo o sistema falhar, levando a telas de recuperação de erro e reinicializações de todo o sistema. Também, por algumas soluções *anti-cheat* serem instaladas juntamente com o sistema operacional, podem levar a outros problemas, tanto a falhas nas atualizações do sistema, como a outras aplicações não funcionarem corretamente (Maario et al., 2021).

Um caso popular envolvendo *anti-cheats* ocorreu com o *ESEA anti-cheat*. Assim como o *Vanguard*, ele possui a capacidade de se manter ativo mesmo que o jogo não esteja aberto. Dessa maneira, por possuir permissão total no computador dos jogadores a qualquer momento, e ter a natureza de ser silencioso e esconder suas ações, um funcionário conseguiu inserir no meio de suas funcionalidades um *malware* que fazia com que todos os usuários minerassem *bitcoins* sem o seu consentimento (Greidanus, 2017).

## 2.5 DISCUSSÃO

Os problemas apresentados afetam a segurança dos usuários. As instabilidades citadas causam falhas no pilar de disponibilidade, já que fazem com que diversos dispositivos não funcionem e resultam até na total falha do sistema operacional. Também é possível verificar a falha nos pilares de confidencialidade e integridade, pois vulnerabilidades nos *anti-cheats* facilitam ataques invasivos com escalada de permissões, causando vazamento das informações dos usuários e modificações não autorizadas em seu computador. Dentro de todos os problemas citados, se evidencia o de privacidade. Casos de instabilidade são facilmente observados por

todos os usuários, já privacidade não. Para verificar a existência de privacidade nesse tipo de *software*, é necessário analisar profundamente as ações dos *anti-cheats*, verificando se realmente somente informações relevantes à detecção de trapaças são obtidas, e como vários casos de violação já foram expostos, é um ótimo ponto a ser investigado.

# 3 REVISÃO BIBLIOGRÁFICA

No contexto deste trabalho, é possível encontrar na literatura trabalhos que abordam o assunto de softwares de *cheats* e *anti-cheats*, listando dezenas de soluções populares, definindo quais técnicas são utilizadas para o seu desenvolvimento, relatando sua evolução durante os anos e propondo sugestões de novas implementações mais efetivas, tanto em detecções, como em segurança contra alterações. Porém pouco é comentado sobre os problemas de privacidade que estes softwares possuem. A Seção 3.1 apresenta a revisão da literatura que aborda a questão da privacidade nos softwares *anti-cheat*, expondo o objetivo principal de cada trabalho e o que é discutido sobre a privacidade dos usuários. A Seção 3.2 faz uma revisão geral sobre o que foi apresentado nos artigos presentes na revisão bibliográfica.

## 3.1 PRIVACIDADE EM ANTI-CHEAT

Existem trabalhos na literatura que abordam questões relacionadas aos *softwares anti-cheats*. Alguns destes também discutem alguns pontos de privacidade. (Greidanus, 2017) aborda a questão legal da privacidade nos *anti-cheats client-side*. Ele trata este tipo de *software* como um possível *spyware*, que é um tipo de *malware* focado na coleta de informações de pessoas. O texto tem o foco em definir em quais circunstancias os *anti-cheats client-side* são ilegais, analisando a literatura, legislação e a jurisprudência, focando principalmente se as regras definidas no regulamento geral de proteção de dados da União Europeia GDPR são cumpridas. Para fazer esta análise, o trabalho usa especulação e as raras informações existentes na literatura sobre o funcionamento desse tipo de software. Aqui a proposta é descobrir empiricamente as ações dos *anti-cheats* e com isso comparar e julgar os efeitos na privacidade.

(Maario et al., 2021) aborda a questão da invasão de privacidade e problemas de segurança causados pelos *anti-cheats kernel-level*, fornecendo opções alternativas com uma taxa de detecção semelhante ou até mesmo melhor, como espelhamento das regras do jogo no servidor, e análise estatística dos eventos e performance dos jogadores. O trabalho aponta que soluções *anti-cheat* que são instaladas com permissão de *kernel* podem causar problemas até mesmo com atualizações do sistema operacional e, junto com a estratégia de escanear toda a memória do dispositivo, causam preocupação de privacidade em seus usuários.

O trabalho (Ven, 2023) discute os *softwares anti-cheat* de outra maneira, focando na perspectiva dos usuários, investigando os motivos que levam os jogadores a utilizarem *cheats*, e expondo as opiniões da comunidade sobre trapaças e softwares *anti-cheat*. Nele, também é desenvolvida a discussão de privacidade, mostrando que os jogadores se preocupam com a sua privacidade e não desejam suas informações expostas ou vendidas pelos softwares *anti-cheat*.

(Matthies et al., 2015) mostra na prática algumas técnicas de engenharia reversa, utilizando-as para trapacear no famoso jogo "Paciência" que vem instalado junto com o sistema operacional *Windows*. São introduzidas algumas contramedidas utilizadas pelos anti-*cheats*, chegando a citar a situação dos problemas de privacidade e comparando os anti-*cheats* a *spywares* só que instalados voluntariamente. Conclui o texto afirmando que cabe ao julgamento de cada usuário se estes *softwares* violam ou não a sua privacidade. Dentro do mesmo contexto, *cheaters* também podem utilizar a engenharia reversa nos *anti-cheats*. Dessa forma (Feng et al., 2008) sugere a maneiras de realizar detecções de trapaças de maneira furtiva, através do uso de *hardware* resistente a adulterações, como um processador instalado na máquina do usuário, independente do processador principal do computador, dificultando o acesso do *cheater* e assim impedindo

que ele determine o que esta sendo medido e também evitando modificações nos resultados das medições. O autor afirma que esta solução é uma faca de dois gumes, pois apesar de obter vantagens em relação aos *cheaters*, ela impede os usuários de saber se sua privacidade está sendo afetada. Isto também pode ser aplicado às demais soluções *anti-cheats*, já que buscam manter suas ações furtivas para dificultar o trabalho dos desenvolvedores de *cheats*.

(Ronkainen, 2021) explora as diferenças entre soluções preventivas e detectivas de trapaças em jogos *online*. Um dos pontos citados que levam a busca de soluções preventivas é a própria questão da privacidade, os diversos problemas de privacidade e segurança contidos nas soluções *anti-cheat* mais famosas são expostos, como a estratégia de escaneamento da memória que acaba verificando informações de outros processos que estão sendo executados nas máquinas dos usuários, mencionando a capacidade do *BattlEye* de enviar arquivos dos usuários para seus servidores e executar códigos *shell* diretamente nos computadores de maneira remota. É discutido como a prevenção pode ser pensada durante o desenvolvimento dos jogos para evitar softwares tão intrusivos monitorando o sistema do usuário.

O trabalho (Jonnalagadda et al., 2021) propõe uma solução *anti-cheat* baseada na captura e análise do *frame* renderizado pelo jogador, detectando se não existe nenhuma informação a mais na tela do usuário, como trapaças visuais. Neste são apresentadas diversas soluções *anti-cheat* comumente utilizadas e propostas, como os *anti-cheats client-side* que monitoram toda a memória do sistema do usuário, afirmando que estes causam problemas de privacidade, já que com acesso a toda memória do computador, verificam dados de outras aplicações que podem conter dados pessoais. É afirmado que soluções *anti-cheat*, como a proposta no texto, não possuem problemas de privacidade e todo o processamento da imagem contida no *frame* pode ser feita na máquina do usuário.

## 3.2 CONSIDERAÇÕES

Analisando os trabalhos relacionados, é evidente que ao decorrer dos anos a discussão sobre a privacidade dos jogadores estar sendo afetada pelos *softwares anti-cheat* só aumentou. Apesar de todos os trabalhos apresentados abordarem esta questão (como apresentado na Tabela 3.1), os problemas existentes são apenas mencionados, citando alguns casos onde a privacidade foi afetada. Apenas o (Greidanus, 2017) possui privacidade como foco. Não foi possível encontrar nenhuma investigação detalhada das ações executadas por algum *anti-cheat*, verificando se os termos definidos nas suas licenças são respeitados, garantindo que a privacidade dos usuários não está sendo afetada.

| **Trabalho** | **Objetivo** | **Privacidade** |
|---|---|---|
| (Greidanus, 2017) | Definir sob quais circunstancias os *anti-cheats client-side* são ilegais considerando a privacidade | O foco da discussão é se a privacidade é respeitada judicialmente pelos *anti-cheats* |
| (Maario et al., 2021) | Destacar os problemas dos *anti-cheats kernel-level* e propor uma solução não intrusiva | São apontados problemas de privacidade causados pelos *anti-cheats kernel-level* |
| (Ven, 2023) | Investigar a visão dos jogadores sobre *cheats* e *anti-cheats* | É mostrada a visão dos usuários sobre a privacidade nos *anti-cheats* |
| (Matthies et al., 2015) | Avaliar diversas técnicas de engenharia reversa, aplicando-as no jogo Paciência do *Windows* | É discutido os impactos das ações dos *anti-cheats* na privacidade dos usuários |
| (Feng et al., 2008) | Propor uma solução furtiva de detecção de *cheats* utilizando um hardware resistente a adulterações | Menciona que soluções furtivas impedem que o usuário garanta que sua privacidade não está sendo afetada |
| (Ronkainen, 2021) | Explorar as diferenças entre soluções preventivas e detectivas de trapaças em jogos online | São apresentado os problemas de privacidade conhecidos nos *anti-cheats* famosos como o *BattlEye* |
| (Jonnalagadda et al., 2021) | Propor uma solução *anti-cheat* baseada na análise dos *frames* dos jogos | São apresentados os problemas de privacidade contidos nos *anti-cheats* que possuem acesso a toda memória dos computadores dos usuários |

Tabela 3.1: Trabalhos relacionados

# 4 PROPOSTA

O capítulo 4 apresenta a proposta da pesquisa. A Seção 4.1 abrange as problemáticas na área. A Seção 4.2 aborda a estratégia proposta, incluindo as ferramentas escolhidas para a coleta de dados e os métodos de avaliação utilizados.

## 4.1 PROBLEMÁTICA

Na literatura, é possível encontrar trabalhos que apresentam e discutem os problemas de privacidade existentes nos *softwares anti-cheat*, como em (Greidanus, 2017) que discute os aspectos judiciários destes problemas. Porém, para definir como um *anti-cheat* funciona e suas ações realizadas nas máquinas do usuário, estes trabalhos se baseiam apenas no que é alegado nas licenças destes *softwares* ou no senso comum, já que este tipo de *software* tende a esconder o que realmente está monitorando para dificultar a tarefa dos criadores de *cheats*. Não existem trabalhos que verificam de maneira prática, avaliando o comportamento dos *softwares anti-cheat* em execução, averiguando se estão de fato respeitando a privacidade de seus usuários.

Existem casos recentes envolvendo privacidade que preocupam os usuários deste tipo de *software*, como do *anti-cheat* Vanguard, que é executado desde a inicialização do sistema e continua atuando mesmo que nenhum jogo esteja aberto, levantando a dúvida do que ele está fazendo neste momento (Ven, 2023), e do ESEA, onde um funcionário inseriu um minerador de cripto moedas juntamente com o *software* de *anti-cheat* e utilizava o computador de diversos usuários para minerar sem permissão (Basque-Rice, 2023). Essas são provas que esta questão de privacidade deve ser averiguada por uma avaliação do *software anti-cheat*.

O objetivo deste trabalho é investigar se os *softwares anti-cheats* estão respeitando a privacidade de seus usuários, focando na análise das ações realizadas por eles durante sua execução, verificando se estão de acordo com o que foi declarado na sua licença e garantindo que nada não informado acontece.

## 4.2 ESTRATÉGIA

A estratégia deste trabalho consiste em uma análise detalhada da execução e das ações realizadas pelos *softwares anti-cheat* existentes no mercado, buscando pontos que possam afetar a privacidade dos usuários. Para isso, algumas ferramentas serão utilizadas para capturar dados relevantes para a avaliação das ações executadas por estes *softwares*, como alteração em registros, permissões concedidas aos processos, pastas acessadas, *DLLs* utilizadas e dados transmitidos na Internet.

Como objeto de estudo serão utilizados os *anti-cheats BattlEye* e *Vanguard*, tanto por sua relevância e popularidade no cenário de jogos atual, como também por suas características específicas. O Vanguard é executado automaticamente durante o *boot* da máquina e se mantém em execução mesmo sem o jogo estar aberto, além de possuir diversos relatos de problemas com interrupção de processos não relacionados a *cheats*. Já o BattlEye foi escolhido por ser *kernel-level* assim como o Vanguard, porém não se mantém ativo desde a inicialização do sistema operacional, e também por ser utilizado em diversos jogos.

O processo de análise proposto terá as seguintes etapas principais:

1. Coleta de dados: Durante a execução dos *anti-cheats* serão coletadas informações relevantes sobre as ações realizadas por estes *softwares*;

2. Avaliação dos dados: Os dados coletados serão analisados, buscando pontos relevantes para a discussão de privacidade;

3. Discussão: Com os dados relevantes coletados e avaliados, será examinado o impacto de *softwares anti-cheat* na privacidade de seus usuários.

### 4.2.1 Ferramentas de Coleta

Nesta subseção, é descrita a abordagem utilizada para coletar os dados relevantes. Para essa finalidade, foram empregadas as seguintes ferramentas:

**Process Explorer:** é uma ferramenta do site *sysinternals*, que fornece uma gama de recursos avançados para o monitoramento e diagnóstico do sistema (Microsoft, 2023). Esta ferramenta é utilizada para identificar, dado um processo, quais arquivos e diretórios ele possui aberto, com permissões de leitura ou escrita, como também quais *Dynamic Link Libraries* (DLLs) foram carregadas por ele (Russinovich, 2023a).

**Process monitor:** faz parte das ferramentas do *sysinternals*, seu propósito é fornecer informações em tempo real sobre o sistema de arquivos, registro e atividade de processos e *threads* em execução (Russinovich, 2023b).

**Wireshark:** é um analisador de protocolos de rede, capaz de monitorar o tráfego e capturar pacotes de rede para análise, fornecendo filtros de diversos tipos (Wireshark, 2023). Com ele é possível observar os pacotes que todas as aplicações rodando estão enviando e recebendo.

### 4.2.2 Métodos de avaliação

Com os dados obtidos pelas ferramentas é possível avaliar se a privacidade dos usuários de *softwares anti-cheats* está sendo afetada de alguma maneira. O *Process Explorer* permite coletar quais diretórios foram acessados pelos *anti-cheats* durante sua execução, verificando se algo incoerente está sendo observado. Também é possível verificar as *DLLs* que foram carregadas por eles, investigando as funções delas será avaliado se alguma ação invasiva está sendo realizada, prejudicando a privacidade dos usuários.

Com o *Process Monitor* é possível observar as alterações no registro do *Windows*, averiguando se os *anti-cheats* fazem alguma alteração ou consulta desnecessária, que fuja de sua função e que afete a privacidade.

O *Wireshark* será utilizado para capturar e avaliar o tráfego de rede gerado por estes *softwares*, verificando se existe a possibilidade de alguma informação pessoal dos usuários estarem sendo enviadas para os servidores dos *anti-cheats*.

Os resultados de todas estas ferramentas serão comparados com a licença de seus respectivos *anti-cheats*, já que é ela que define quais dados dos usuários estão sendo utilizados e como eles são tratados.

# 5 AVALIAÇÃO DE RESULTADOS

Neste capítulo é apresentada toda a coleta e avaliação dos dados obtidos. A Seção 5.1 define a estratégia utilizada para coletar dados e avaliação do impacto de *softwares anti-cheat* na privacidade de seus usuários. A Seção 5.2 apresenta a análise dos dados coletados. A Seção 5.3 apresenta os termos de licenças de uso e políticas de privacidade destes *softwares*. Por fim, na seção 5.4 é feita uma discussão do impacto de *softwares anti-cheat* para o usuário, considerando as suas políticas de privacidade e observações da análise.

## 5.1 ESTRATÉGIA DE AVALIAÇÃO

Visando investigar *softwares anti-cheat*, verificando se a privacidade de seus usuários está sendo respeitada, as ferramentas definidas para a captura dos dados de execução dos *softwares* devem fornecer informações que descrevam as operações realizadas pela aplicação durante seu processo de execução. Estas ferramentas também devem permitir o armazenamento de dados para futuras avaliações. No processo de avaliação proposta, será considerando tanto operações realizadas no sistema operacional como comunicações para outros serviços que não sejam locais (comunicação pela rede).

Para alcançar estes objetivos utilizamos um conjunto de três ferramentas. O *Process Explorer* foi utilizado para conseguir uma lista de quais *DLLs* cada *anti-cheat* está utilizando durante sua execução, como *DLLs*, que são bibliotecas que fornecem um conjunto de funcionalidades específicas. Assim, é possível ter uma base de que tipo de ações estão sendo executadas por estes *softwares*. Com essa ferramenta também é possível extrair uma lista de arquivos e chaves de registros do sistema que estão sendo utilizados durante a captura, o que permite analisar o momento capturado, verificando a existência de informações sensíveis do usuário nesses dados examinados.

Para melhor compreender as operações realizadas pelo *software anti-cheat* também utilizamos o *Process Monitor* visando coletar todas as chaves do registro de sistema que foram lidas ou alteradas, todas as *DLLs* carregadas, e os pacotes de rede enviados para a Internet. Este conjunto de dados permite a construção de um traço de execução, que descreve padrões de ações executadas por estes *softwares* desde o inicio da sua execução. Através deste conjunto de dados é possível comparar as semelhanças e diferenças entre os dados acessados, buscando a presença de informações sensíveis do usuário e o comportamento de cada *anti-cheat* em sua execução, verificando se alguma ação realizada pode afetar a privacidade dos seus usuários.

Por fim, também considerando todas as interações do *anti-cheat* com a Internet. Coletamos os dados com o *Wireshark*, observando o volume e destino das comunicações, investigando quais dados são enviados na rede e para quais servidores estão indo, verificando se informações sensíveis dos usuários não são enviadas para servidores próprios dos *anti-cheats* ou então do jogo.

Para a coleta dos dados, foi utilizado um computador com o Windows 10 Pro versão 22H2 Build 19045.4046, Process Explorer versão 17.05, Process monitor versão 3.96, Wireshark versão 4.2.3-0-ga15d7331476c. As respectivas versões dos *softwares anti-cheat* investigado também levam em consideração a versão do jogo. Para a execução do *Battleye* foram utilizados os jogos *Unturned* versão 3.24.1.0 e Tibia versão 13.34.14623. Como o *Vanguard* é apenas utilizado no *Valorant*, foi utilizado o *Valorant* versão 8.02. Ambos *anti-cheats* não informam sua versão.

Ao todo, foram executadas 3 capturas com cada ferramenta para cada jogo apresentado, de maneira separada para evitar conflitos entre elas. Este número se mostrou satisfatório já que não foram observadas diferenças significativas em cada captura.

A captura de dados para o *Vanguard* apresenta maiores desafios, já que sempre que o *Process Monitor* está aberto o jogo apresenta uma tela de erro, retornando ao normal somente após o sistema ser reiniciado. Após diversas tentativas e mudanças de configuração do sistema, como ativar e desativar o *Secure Boot* e TPM 2.0, foi possível capturar algumas execuções completas do *Vanguard* durante partidas de *Valorant*. Já o mesmo processo para o *Battleye* não apresenta o mesmo comportamento.

## 5.2 ANÁLISE DE DADOS

A principal diferença entre o *Battleye* e o *Vanguard* acaba sendo observada na forma como o *anti-cheat* é encontrado no sistema. O *Battleye* é executado junto com o jogo. Desta forma, ele acaba sendo iniciado com a aplicação e posteriormente encerrado quando o jogo é encerrado. Já o *Vanguard* utiliza um *Driver* que é iniciado junto com o sistema operacional. Ambos modelos de execução podem ser observado pelo *Process Explorer*, já que os *anti-cheats* iniciam um processo ao abrir o jogo que ele está vinculado, este sendo o alvo da análise.

Caso este processo da aplicação *anti-cheat* seja finalizado, o jogo é encerrado, não permitindo a execução sem o *anti-cheat* executando. Para realizar a coleta de dados foi realizado a (i) abertura da aplicação responsável pela captura de dados; e (ii) abertura do jogo, que inicia a execução do *software anti-cheat*. Este processo foi repetido para cada uma das ferramentas e *anti-cheat* investigados.

Com o *Process Explorer* foi possível obter a lista de *DLLs* que a aplicação interage durante seu processo de execução[1], arquivos abertos e registros do sistema. Os *softwares anti-cheat* não aparentam ofuscar os recursos acessados durante sua execução, sendo de fácil identificação na aplicação. Identificamos que 42% das *DLLs* observadas são comuns para ambos os *softwares*, com as respectivas aplicações acessando recursos como funções de baixo nível do *Kernel* e informações sobre o sistema, como hardware, dispositivos e eventos gerados através das *DLLs ntdll* e *ntmarta*. Ambos também utilizam *DLLs* para manipulação da *shell* do *Windows*, como *SHCore, shell32, shlwapi*, que possibilitam executar qualquer comando do sistema e com isso obter diversas informações, desde sistema de arquivos até de processos e *drivers* em execução.

Através do *Process Monitor* foi realizado a construção dos respectivos traços de execução da aplicação. Onde identificamos as principais operações/ações realizadas pelo *software anti-cheat*. Este traço de execução visa representar todo o processo desde o inicio da execução até seu encerramento. Os padrões de execução também permitem a comparação entre diferentes *softwares anti-cheat* para a compreensão de semelhanças.

A Figura 5.1 apresenta o traço de execução observado para o *Battleye*. O quadrado início indica a abertura do jogo e o começo da execução do *anti-cheat*, seguindo o fluxo de execução é possível acompanhar quais ações foram executadas em sequência até o fim da execução. A sequência de execução observada no *Battleye* começa com o carregamento das *DLLs* do sistema, após sendo realizado a leitura do registro do *Safer* que é uma *API* do *Windows* que controla a política de execução das aplicações no sistema. A chave *CodeIdentifiers* indica o local onde um arquivo de *log* contendo todas as aplicações que foram executadas no sistema, qual processo iniciou e qual regra foi utilizada para avaliar se ela poderia ser executada ou não (Microsoft, 2024).

[1]Os dados coletados estão disponíveis em `https://github.com/Viniciusmcf/dadosTcc`.

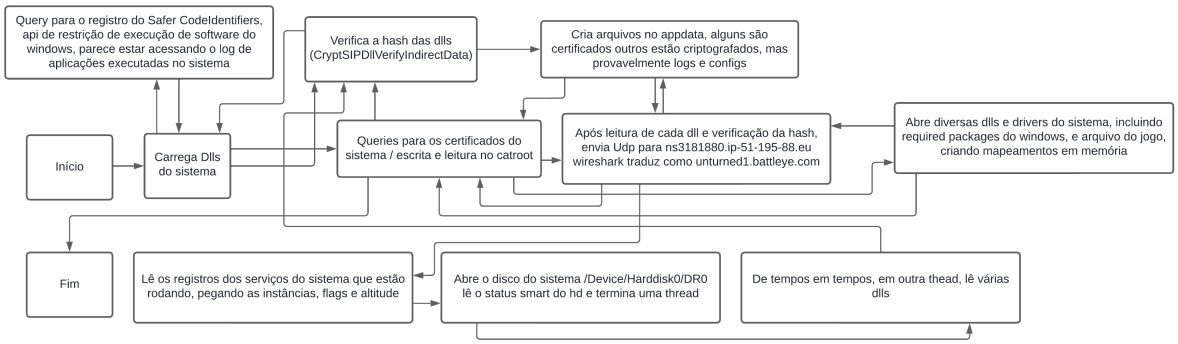



Figura 5.1: Representação do traço de execução do *Battleye*.

Após este processo, novas *DLLs* são carregadas e são feitas verificações das *hashes* com a função *CryptSIPDllVerifyIndirectData*. Aparentemente verificando a integridade das *DLLs*, em conjunto também são feitas *queries* para os registros de certificados do sistema e escritas/leituras no diretório *CatRoot*, que contém certificados do sistema, o que indica verificação de autenticidade dos arquivos utilizados pelo *anti-cheat*. Durante esse processo, alguns arquivos são criados no diretório *AppData*, onde foi possível visualizar o conteúdo de alguns deles, que consistia de certificados salvos, outros estavam criptografados, mas pela localização se tratam de *logs* de execução.

Após estes eventos, um padrão parece se repetir durante a execução: a abertura de alguma *DLL* ou *Driver*, seguido da verificação da *hash* utilizando a função *CryptSIPDllVerifyIndirectData*, então é feito o acesso aos certificados do sistema e enviado um pacote *UDP* para o servidor do *Battleye*. Após esse ciclo, são lidos os registros dos serviços que estão executando no sistema, que acaba incluindo as ferramentas utilizadas para a coleta de dados, como o *Process Monitor*. Assim, o ciclo principal é finalizado, fazendo a abertura do disco do sistema para leitura e escrita, capturando as informações *SMART* desse dispositivo e finalizando a *thread*. A partir deste momento, o *anti-cheat* faz verificações periódicas em intervalos irregulares de alguns segundos, repetindo o ciclo descrito anteriormente de abertura de alguma *DLL* ou *Driver*, seguido da verificação da *hash* utilizando a função *CryptSIPDllVerifyIndirectData* e acesso aos certificados do sistema, no fim observamos que a aplicação envia um pacote *UDP* para o servidor do *Battleye*.

Com este traço de execução, foi observado que o *Battleye* faz acesso recorrente a *drivers* e certificados do sistema, recursos que, se comprometidos, podem danificar ou até corromper todo o sistema do usuário.

A Figura 5.2 apresenta o traço de execução observado para o *Vanguard*, no mesmo formato do traço do *Battleye*. A sequência de execução observada no *Vanguard* começa com o carregamento das *DLLs* do sistema, então é feita a leitura do registro do *Safer*.

Após esta leitura, novas *DLLs* são carregadas e são feitas leituras no diretório *CatRoot*. Durante toda a execução, alguns arquivos de *logs* são criados no diretório do *anti-cheat*, não foi possível visualizar o conteúdo pois estavam criptografados, mas estavam no diretório */Logs* e possuíam a extensão *.log*. Mais *DLLs* são carregadas e *queries* são feitas para diversos registros do sistema, incluindo *computer name* e *display*, que podem respectivamente informar o nome do computador do usuário e informações do monitor; informações que, como apresentado em (Kaur et al., 2017), podem ser usadas para identificação dos usuários, já que existem algoritmos especializados na criação de uma identificação única de usuários (*fingerprint*) que funcionam com poucas informações, incluindo a resolução do monitor.

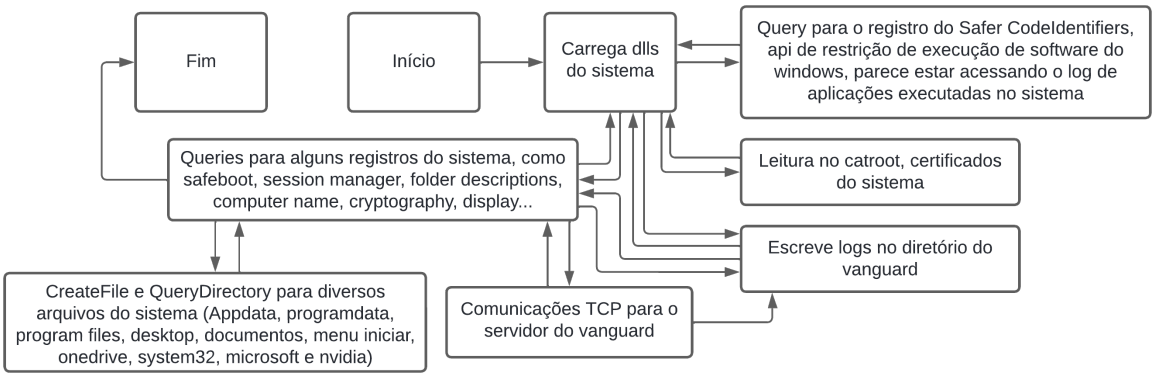



Figura 5.2: Representação do traço de execução do *Vanguard*.

Então, o *anti-cheat* lê diversos diretórios de arquivos do sistema, incluindo *Desktop, onedrive*, documentos e pastas que existem principalmente para arquivos pessoais. Ao fim das verificações, diversos pacotes *TCP* são enviados para o servidor do *Vanguard*, mais *logs* são escritos e o processo de *queries* para certificados e leitura de arquivos se mantém durante a execução.

Com este traço de execução, foi observado que o *Vanguard* faz acesso recorrente a arquivos dos usuários, incluindo pastas pessoais que podem conter dados sensíveis, e envia informações criptografadas para seu servidor ao final das verificações dos diretórios, causando a preocupação de que informações pessoais e sensíveis dos usuários estejam inclusas nestas enviadas para o seu servidor.

Através da investigação dos traços de execução, conseguimos identificar fatores chaves no processo de execução de aplicações *anti-cheat*. Como por exemplo, ambos os softwares iniciam sua execução carregando as *DLLs* que serão utilizadas e fazendo uma *query* para o registro *CodeIdentifiers* do *Safer* que é uma API de restrição de execução de softwares do *Windows*. Esta *query* parece configurar o acesso a um *log*, permitindo que os *anti-cheats* observem todas as aplicações que foram executadas no sistema anteriormente e também as que forem abertas durante a execução. Foi observado que ambos verificam certificados do sistema, tanto no diretório *catroot*, quanto nos registros e escrevem *logs* em seus diretórios, enviando pacotes de maneira recorrente a servidores próprios.

Mais de 80% da execução se trata da verificação dos arquivos do sistema. O *Battleye* parece focar apenas nas *DLLs*, dedicando 19.3% das suas ações para o sistema de arquivos e 80% nos registros verificando as *hashes*. Estas *DLLs* incluem até mesmo definições do *Windows Defender*, o que não parece ter muita utilidade para um anti-cheat. A cada verificação ele envia um pacote UDP para seu servidor. Já o *Vanguard*, além das *DLLs* também faz uma varredura em diversos arquivos do sistema, incluindo diretórios pessoais do usuário, como Desktop, Documentos, OneDrive, e pastas do sistema operacional, como system32, programfiles, Appdata, utilizando 64% das suas ações no sistema de arquivos e 19% nos registros verificando as *hashes*.

Em uma postagem da *Riot Games* (Riot, 2024a), é afirmado que assim como em outros *anti-cheats* o *Vanguard* executa uma varredura de assinaturas, buscando dados em memória que sejam iguais aos utilizados por softwares de trapaças. O que indica que o comportamento observado no *Battleye* de análise de *DLLs*, pode se tratar também de uma varredura de assinaturas. No fluxo de execução observado, o maior potencial de prejudicar a privacidade dos usuários são

as verificações feitas no sistema de arquivos e os pacotes enviados aos servidores próprios dos *anti-cheats*, já que informações sensíveis dos usuários podem ser obtidas e enviadas pela rede.

Após a varredura executada pelo *Vanguard*, foram observadas diversas comunicações com o seu servidor utilizando TCP, o que é semelhante ao comportamento do *Battleye*, que enviou pacotes após a análise de cada *DLL*, porém os pacotes eram UDP e os conteúdos estavam criptografados.

Com o *Wireshark* foram avaliados os dados enviados através da rede por cada *anti-cheat*, observando principalmente o volume e destino das comunicações. No caso do *Vanguard*, a maioria desses pacotes foram enviados para o endereço *na.vg.ac.pvp.net.cdn.cloudflare.net (2606:4700:4400::ac40:9258)*, que é uma rede de entrega de conteúdo (CDN) na *Cloudflare*. Toda a comunicação foi feita utilizando *TCP*, somando 16.2 MB durante 2 minutos de execução. Já o *Battleye*, no começo de sua execução, faz uma autenticação utilizando *TCP* para o servidor *battleye.b-cdn.net (2400:52e0:1690::1157:1)*, que é uma CDN na *Bunny*, somando 16.9 MB em 1 segundo. Após isso como observado no *Process Monitor* o *anti-cheat* enviou para *unturned1.battleye.com (51.195.88.133)*, que é um servidor hospedado na *OVH*, um total de 257 pacotes *UDP* em 10 segundos e então enviou mais 6 pacotes para o mesmo servidor em intervalos irregulares de tempo durante 4 minutos de execução somando 33 KB.

- **Registro**: Ambos os *anti-cheats* acessaram os registros do *Safer Code Identifiers* para obter os *logs* das aplicações executadas no sistema. Também acessaram registros que contém informações do sistema como *ComputerName* e *Display*, que podem respectivamente informar o nome do computador do usuário e informações do monitor, informações que podem ser usadas para identificação dos usuários. A principal diferença nos acessos aos registros dos *anti-cheats*, é que o *Battleye* utiliza com mais frequência as funções de criptografia como a *CryptSIPDllVerifyIndirectData* e acessa diversos registros de certificados.

- **DLL**: Dentre as *DLLs* utilizadas pelo *Battleye* que o *Vanguard* não utiliza, se destacam *cfgmgr32* responsável por interagir com os dispositivos de *hardware* do sistema, como discos, mouses, teclados, cameras, etc; *sfc*, responsável pelo verificador de arquivos do sistema do Windows, pode ser utilizada para verificar a integridade do sistema e restaurar arquivos corrompidos ou danificados; e a *mpr*, utilizada para interação com dispositivos e pastas compartilhadas na rede. Dentre as *DLLs* utilizadas pelo *Vanguard* que o *Battleye* não utiliza, se destacam *FWPUCLNT*, que permite a filtragem e inspeção de pacotes de rede em baixo nível, permitindo 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 permite a conexão remota a computadores e a execução de aplicativos em sessões de usuário remoto. Dentre as *DLLs* que ambos os *anti-cheats* utilizam, se destacam *ntdll* e *ntmarta* que são 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*, que está relacionada ao *Windows Defender Application Control (WDAC)*, é utilizada para restringir quais aplicativos e *scripts* podem ser executados no sistema; e *windows.storage* responsável pelo armazenamento do sistema e manipulação de arquivos e pastas. Ambos também utilizam *DLLs* para manipulação da *shell* do *Windows*, como *SHCore, shell32, shlwapi*, que possibilitam executar diversos comandos e obter diversas informações do sistema.

- **Arquivos acessados**: O *Battleye* durante toda sua execução, só acessou *DLLs*, arquivos do jogo e *Drivers* do sistema. Já o *Vanguard*, além das *DLLs*, *Drivers* e arquivos do jogo, acessou diversos diretórios do sistema e arquivos de usuário. Dentre eles estão *Windows, Program Files, ProgramData, Desktop, Documents, Microsoft, OneDrive, Nvidia, AppData, Start Menu e Windows Defender*, incluindo subdiretórios dos diretórios listados e alguns arquivos que estavam dentro deles.

## 5.3 LICENÇAS DE SOFTWARE ANTI-CHEAT

Examinando a política de privacidade do *Battleye*, é possível encontrar a explicação da empresa para o processamento de dados pessoais dos usuários. A *BattlEye* afirma processar informações pessoais para fornecer seus serviços de detecção do uso de *cheats*, com o objetivo de garantir um ambiente de jogo justo (BattlEye, 2024). Ele é definido como o principal destinatário dos dados, porém também podem compartilhar com o editor do jogo, ou quaisquer parceiros ou afiliados que forem explicitamente mencionados. Essa mesma política define que os dados pessoais processados e coletados pelo *anti-cheat* são:

- Endereço IP;
- Identificadores do jogo (Apelido do jogo, id da conta, etc.);
- Informações e identificadores do hardware do dispositivo (Números seriais);
- Informações do sistema operacional;
- Informações dos arquivos em memória do jogo e do sistema operacional;
- Informações sobre os processos executando, *drivers* e outros códigos executáveis;
- Nomes de arquivos inclusos nas outras informações listadas, que podem conter o nome de usuário do sistema operacional.

A empresa afirma que segue uma política de minimização de dados, guardando-os apenas quando necessário. Ou seja, só quando são encontradas informações que possam indicar o uso de *cheats*, podendo ser armazenados por toda a duração do serviço da empresa para o jogo. A *Battleye* declara que não armazena nenhuma informação sobre a maioria dos usuários, esta maioria sendo os usuários que não possuem indícios de uso de *cheats* (BattlEye, 2024).

É informado que, apesar do *Battleye* precisar de acesso total ao sistema para detectar todos os tipos de *cheats*, nenhuma informação pessoal como dados, documentos, detalhes de cartão de crédito, senhas ou similares são buscadas nem transmitidas. É apresentada a capacidade de transmissão de código executável, identificado como suspeito, aos servidores para análise posterior, porém a empresa afirma que usuários normais, que não executam *softwares* suspeitos, nunca devem ter informações transmitidas, e que, fora código suspeito, nenhum outro conteúdo da memória é enviado para os servidores do *Battleye*. Por fim, o *Battleye* alega que todos os dados coletados relativos aos usuários são armazenados em servidores seguros, o que normalmente acontece somente quando detectam o uso de *cheats*.

Avaliando o *End-User License Agreement* (EULA) do *Battleye* é possível encontrar mais informações sobre os dados coletados e processados, é apresentado que o *anti-cheat* pode verificar toda a memória de acesso aleatório (RAM) dos usuários e quaisquer arquivos e pastas relacionados ao jogo e ao sistema, usando algoritmos de identificação de *cheats*, podendo relatar os resultados para servidores próprios ou outros computadores conectados na rede, armazenando

estas informações somente com o propósito de prevenir e detectar o uso de *cheats*. Está descrito no *EULA* que o *Battleye* pode ainda relatar e armazenar o endereço IP dos usuários, nome e identificador da conta do jogo, apelido no jogo e informações relacionadas ao sistema e ao hardware, como ids de dispositivos e números de série de hardware, não relatando nenhuma informação não relacionada a identificação de *cheats* e de identificação pessoal, exceto as mencionadas na licença.

Examinando os termos de serviço da Riot Games, empresa responsável pelo *Vanguard*, é possível encontrar uma seção sobre monitoramento e *anti-cheat* onde é declarado que podem monitorar ativamente o uso dos Serviços da Riot, tanto nos próprios servidores quanto nos computadores ou dispositivos dos usuários, para uma ampla variedade de finalidades diferentes, incluindo prevenção de trapaças e redução de comportamento tóxico de jogadores, e para prevenção de trapaças podem exigir a instalação de *software anti-cheat* que pode ser executado em segundo plano no dispositivo dos usuários (Riot, 2024b).

A Riot Games possui uma nota de privacidade que informa sobre sua política *anti-cheat* e esclarece quais dados são coletados por seus serviços e como são processados. Sobre *anti-cheat* a empresa apresenta que procura oferecer aos jogadores uma experiência de jogo justa, divertida e competitiva, tendo definido nos termos de serviço que programas de terceiros como *mods*, *hacks*, *cheats*, *scripts* e *bots* são proibidos. É declarada a possibilidade de uso de tecnologias *anti-cheat* e de prevenção contra fraudes, como *software anti-cheat* executado no dispositivo dos usuários, que podem tomar decisões automatizadas, como suspensões temporárias ou permanentes e limitações ao conteúdo do jogo (Games, 2024).

A empresa afirma que, resumidamente, dependendo dos serviços utilizados pelos usuários, os tipos de dados a seguir podem ser coletados e divulgados para fins comerciais:

- Identificadores, incluindo nome, nome no jogo, nome de usuário, endereço de email, endereço de cobrança, número de telefone, endereços IP, IDs exclusivos de dispositivos e identificadores online;
- Características de classificação protegidas pela legislação estadual ou federal, incluindo data de nascimento, idade, sexo e informações de saúde relacionadas à acessibilidade;
- Informações comerciais, incluindo interação com os Serviços da Riot, informações de suporte ao cliente, histórico de compras, forma de pagamento e outras informações de pagamento;
- Informações de atividade na Internet ou em outras redes eletrônicas, incluindo o uso dos Serviços da Riot (carimbos de data e hora, cliques, rolagem, tempos de navegação, mapas de calor de navegação, pesquisas, páginas de referência/saída e atividades e interações nos clientes de jogo, incluindo registros de bate-papo), computador ou dispositivo (capacidades de processamento, fabricante e modelo, idioma e outras configurações regionais, resolução de tela e configurações semelhantes), conexão com os Serviços da Riot (tipo e versão do navegador, nome e versão do sistema operacional, ISP e suas configurações de preferência), como desempenho dos Serviços da Riot (erros de carregamento e tempos de resposta), plataforma de terceiros e dados de uso, além de informações de publicidade e análise;
- Dados de geolocalização, incluindo país, e uma localização aproximada derivada do endereço IP;

- Informações de áudio, eletrônicas, visuais, térmicas, olfativas ou similares, incluindo bate-papo de áudio e texto, jogabilidade gravada e informações relacionadas (ordem de construção e ações no jogo);
- Outras informações, incluindo dados de pesquisas (preferências, interesses, informações demográficas gerais, hobbies, jogos favoritos) e informações de inscrição em concursos;
- Inferências extraídas das categorias de informações não sensíveis acima, incluindo preferências do consumidor. Não são inferidas características utilizando informações pessoais confidenciais.

Para realizar esta coleta de dados, a Riot afirma que seus serviços usam *cookies*[2], *web beacons*[3] e tecnologias semelhantes para coletar, armazenar e ler arquivos automaticamente nos dispositivos dos usuários. Também usam seus servidores em conexão com essas tecnologias para coletar informações sobre as formas como os usuários interagem com os seu serviços, armazenando-as em *logs* contendo IDs de dispositivos, endereços IP e especificações de hardware ou software, além de detalhes de como cada usuário utiliza os serviços.

Sobre como as informações coletadas são utilizadas, é apresentado na nota que utilizam para a operação do negócio, declaram ser necessárias para melhorar os produtos, serviços, e a comunicação e anúncios aos usuários. A empresa relata que não compartilha as informações de contato com terceiros sem o conhecimento dos usuários, com exceções apenas por questões legais, porém declara que compartilha informações que não sejam de contato (como nome nos jogos, estatísticas e outras informações agregadas ou anônimas) inclusive publicamente por meio da *API* da Riot Games.

A *Riot Games* está no processo de inserção do seu *anti-cheat Vanguard* em seu outro jogo muito popular, o *League of Legends*. Com isso, fizeram uma publicação em seu site tirando duvidas comuns e explicando partes do funcionamento do *Vanguard*, incluindo informações como dados específicos que são coletados pelo *anti-cheat* (Riot, 2024a).

Nesta publicação, é afirmado que o *Vanguard* coleta só o que é necessário para proteger os jogos, executando uma varredura nos arquivos do sistema, buscando por bytes na memória que correspondam à um *cheat* conhecido, mas apenas enviam de volta ao seu servidor uma resposta binária, informando se existe ou não essa correspondência, e caso exista qual foi o número do processo identificado. Porém, também alegam que algumas detecções necessitam de uma captura instantânea para análise posterior, e estas podem conter informações pessoais identificáveis, como nome de usuário em um caminho de arquivo. A empresa declara que só utilizam as informações destas capturas para identificar *cheats* e que estes dados ficam armazenados apenas 14 dias.

Na mesma publicação, é informado que utilizam Inteligencia Artificial no lado do servidor para ajudar na detecção de *cheats*. Porém, como já mencionado anteriormente no Capítulo 2, soluções apenas *Server Side* não funcionam bem para detectar trapaças visuais, tornando necessário uma solução também *Client Side*, como o *Vanguard*. É afirmado que apesar de possuir o *Driver* no nível de *Kernel*, ele não é utilizado para coletar mais informações. A empresa alega que toda informação necessária pode e é capturada em nível de usuário, e este *Driver* apenas serve como uma garantia que o ambiente que o jogo está rodando é seguro, por esse mesmo motivo ele é iniciado junto com o sistema operacional: para verificar que nada indevido ocorreu entre a inicialização do sistema e a abertura do jogo.

[2]Método utilizado para salvar informações de personalização, sessão e identificação de cada usuário em sites.

[3]Técnica utilizada para identificar, de maneira invisível, se um usuário acessou algum conteúdo específico, rastreando seu comportamento.

Apesar das informações nas licenças da Riot serem mais abrangentes, já que não se referem apenas ao seu *anti-cheat Vanguard*, mas sim de todos os diversos serviços prestados pela empresa, ambas as empresas deixam explícito que o papel de seus *anti-cheats* é de detectar as trapaças que estejam sendo utilizados nos jogos, para assim manter um ambiente justo e competitivo. Foi apresentado que tanto o *Vanguard*, como o *Battleye* procuram coletar apenas o que for necessário para detectar os *cheats* e apenas enviam para seus servidores amostras de dados que foram identificados como suspeitos. É possível observar semelhanças nos dados coletados por eles, ambos declaram que coletam as seguintes informações de seus usuários:

- Endereço IP;
- Identificadores do jogo;
- Informações e identificadores do hardware do dispositivo;
- Informações do sistema operacional.

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 estes *softwares* possuem acesso a diversas informações do sistema dos seus usuários, como números seriais de componentes de hardware, versões do sistema, lista de processos e serviços que estão sendo executados, endereço IP, identificadores dos jogos, acesso aos arquivos do sistema, incluindo pastas do usuário, e também aos códigos executáveis em memória. Especialmente com o traço de execução obtido com auxilio do *Process Monitor*, foi possível evidenciar esses comportamentos durante a execução dos *anti-cheats*.

Como os *anti-cheats* possuem grande liberdade no sistema, possuindo acesso completo ao sistema de arquivos e a memória dos computadores dos usuários, e como observado pelo *Process Monitor* na análise de dados, pastas pessoais dos usuários são listadas e verificadas por estes *softwares*, o que apesar de estar especificado nas licenças, não permite a total privacidade de seus usuários, já que como apresentado na Seção 2.1 a privacidade da informação é definida como a medida de quanto as pessoas decidem quais informações são coletadas sobre elas (Schellens, 2021), como os usuários de *anti-cheats* não podem definir exatamente o que pode ser verificado em seus computadores, estes *softwares* afetam a privacidade de seus usuários.

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 e 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 no computador de um usuário, monitorando as atividades e reportando a 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 e 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 e Goundar, 2020);
- Cavalos de troia: Software aparentemente benigno e de interesse do usuário, mas que contém funções maliciosas ocultas, capaz de 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. Assim, como não definem explicitamente o que consideram como código suspeito, apenas afirmam que é o que se assemelha aos *cheats*, o que abre a possibilidade de considerar qualquer coisa suspeita. Como mencionado na Seção 1, o *Vanguard* já detectou erroneamente *drivers* básicos de teclado e mouse como *cheats*. 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: como foi atestado com o *Wireshark*, diversos pacotes foram enviados e recebidos de servidores próprios dos *anti-cheats*.

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*. Com o acesso total aos dados da máquina, tanto em memória como em disco, informações sensíveis como e-mails, nomes, senhas podem aparecer no meio de código considerado suspeito, este que, como descrito na Seção 5.3, pode ser enviado para os servidores próprios dos *anti-cheats* para análise posterior, assim se assemelhando muito a um *Key Logger*.

Como mencionado anteriormente, os *anti-cheats* possuem acesso a diversas informações dos usuários, tornando-se potenciais alvos para entidades maliciosas. Se um atacante descobrir qualquer brecha neste tipo de *software* é possível conseguir acesso facilitado aos dados do usuário, do sistema e até uma escalada de permissões, podendo ter acesso remoto arbitrário a máquina do usuário e manipulação total do funcionamento do sistema. Outro alvo de ataques são as informações armazenadas em servidores próprios dos *anti-cheats*. Como comentado, é possível que dados sensíveis dos usuários estejam armazenados nestes servidores, desde nomes até e-mails e senhas. Apesar de todos os sites da Internet estarem sujeitos a ataques e vazamentos deste tipo, as informações que os *anti-cheats* possivelmente mantém dos usuários são particularmente comprometedoras, já que podem ser qualquer arquivo de seus computadores, até mesmo dados bancários que foram anotados em um arquivo de texto.

Nas licenças também é apresentado que técnicas como *cookies* e *web beacons* são utilizadas, estratégias comumente utilizadas para identificação e rastreamento de usuários, isso em conjunto com os dados como IP e informações específicas do hardware que são coletadas dos usuários podem ser utilizadas em conjunto para montar uma identificação única (*fingerprint*) muito precisa de cada pessoa. Assim, como apresentado em (Kaur et al., 2017), poucos dados do navegador já são o suficiente para criar uma *fingerprint*. Isso se torna mais um agravante à segurança dos usuários, se uma identificação tão especifica quanto essa for vazada em um ataque malicioso, a privacidade dos usuários seria afetada ao ponto de permitir que terceiros rastreiem os usuários de maneira precisa ao longo do tempo, mesmo havendo mudanças nos dispositivos ou nas redes utilizadas. Além disso, estes dados podem ser usados para direcionar ataques personalizados, como *phishing* ou *malware* que exploram vulnerabilidades específicas do sistema operacional, navegador ou outros componentes identificados.

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 CONCLUSÃO

A popularidade dos jogos eletrônicos, principalmente os multijogadores, tornou comum a existência de jogadores utilizando *softwares* que alteram a jogabilidade, manipulando o comportamento do jogo, a fim de obter vantagens de maneira desonesta, chamados de *cheats*. Em contrapartida, os desenvolvedores de jogos recorrem a maneiras cada vez mais sofisticadas de métodos de detecção deste tipo de *software*, a solução mais adotada é o *software* chamado *anti-cheat*, instalado diretamente na máquina de todos os usuários durante a instalação do jogo. Esse *software* permanece monitorando o sistema do usuário durante a execução do jogo, buscando por indícios da utilização de *cheats* e ao detectar o uso, também podem aplicar restrições aos usuários, como o banimento.

Os *cheats* são constantemente desenvolvidos e aprimorados, tentando burlar os métodos de detecção e permanecer indetectados nas máquinas dos usuários. Para acompanhar esta evolução os *anti-cheats* estão recorrendo a soluções invasivas, como o uso de *drivers* com permissão de *kernel*, instalados juntamente com o sistema operacional e com permissão total nas máquinas do usuários. Também tornaram suas ações obscuras para dificultar ao máximo o trabalho de engenharia reversa feita pelos desenvolvedores de *cheats* e possuem capacidade de enviar arquivos das máquinas dos jogadores para verificação remota no servidor. Assim, os jogadores se tornaram receosos e preocupados com sua privacidade, pois não é possível saber exatamente quais informações estão sendo coletadas e transmitidas na rede, mesmo que exista nos termos de uso e de privacidade, cláusulas indicando que nenhuma informação não relacionada com *cheats* será coletada e nenhuma informação será vendida, não é possível confirmar que realmente nada sensível está sendo capturado pelo *anti-cheat*. Ainda, existe a possibilidade que um ataque malicioso comprometa o servidor do *anti-cheat* e assim qualquer informação do usuário obtida para análise seria vazada.

Como tentativa de verificar se a privacidade dos usuários de *anti-cheats* é respeitada, foi realizado um estudo, utilizando ferramentas para verificar as ações executadas por estes *softwares* durante a sua execução. Foram escolhidos os *anti-cheats Vanguard* e *Battleye* para serem analisados, foi utilizado o *Process Explorer* para conseguir uma lista de quais *DLLs* cada um utilizou durante sua execução, como *DLLs* são bibliotecas que fornecem um conjunto de funcionalidades específicas. Assim, é possível ter uma base de que tipo de ações estão sendo executadas por estes *softwares*. O *Process Monitor* foi empregado para coletar informações mais específicas em tempo real dos *anti-cheats*, como alterações nos registros do sistema, acesso e mudanças feitas no sistema de arquivos e comunicações na rede, para uma investigação mais detalhada das comunicações realizadas na rede também foi utilizado o *Wireshark*. Como forma de validação dos dados encontrados, foram analisadas as licenças destes *anti-cheats* buscando informações sobre como a privacidade dos usuários é tratada, verificando se o que está descrito é o mesmo que foi observado na execução real dos *softwares*.

Com as informações obtidas pela análise foi possível montar um traço de execução dos *anti-cheats*, ordenando cronologicamente as ações executadas por estes *softwares*, dentre as ações observadas, se evidenciam acesso recorrente a recursos como certificados do sistema e *drivers*, que se comprometidos podem danificar ou até corromper todo o sistema do usuário, comunicação frequente com servidores próprios pela rede e principalmente acesso a diversas pastas do sistema, incluindo arquivos pessoais dos usuários como área de trabalho, documentos e *Onedrive*. A análise das licenças mostrou que existe 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.

Dessa forma, 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.

# REFERÊNCIAS


Agarwal, A. e Agarwal, A. (2011). The security risks associated with cloud computing. *International Journal of Computer Applications in Engineering Sciences*, 1(Special Issue on):257–259.

Alayed, H., Frangoudes, F. e Neuman, C. (2013). Behavioral-based cheating detection in online first person shooters using machine learning techniques. Em *2013 IEEE conference on computational inteligence in games (CIG)*, páginas 1–8. IEEE.

Bajaj, S. (2022). *Detect Cheater in Online Gaming using AI*. Tese de doutorado, Dublin, National College of Ireland, Dublin, Irlanda.

Baldassarre, M. T., Barletta, V. S., Caivano, D. e Scalera, M. (2020). Integrating security and privacy in software development. *Software Quality Journal*, 28:987–1018.

Banerjee, C. e Pandey, S. (2009). Software security rules, sdlc perspective. *arXiv preprint arXiv:0911.0494*.

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

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

Buja, A. G. (2021). Cyber security featuresfor national e-learning policy. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, 12(5):1729–1735.

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

Corporation, V. (2024). Valve anti-cheat (vac) system. `https://tinyurl.com/hpthkw9e`.

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

Dias Canedo, E., Toffano Seidel Calazans, A., Toffano Seidel Masson, E., Teixeira Costa, P. H. e 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. e Yan, J. (2006). Detecting cheaters for multiplayer games: Theory, design and implementation. Em *Consumer Communications and Networking Conference*, páginas 1178–1182.

Feng, W.-c., Kaiser, E. e Schluessler, T. (2008). Stealth measurements for cheat detection in on-line games. Em *Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games*, páginas 15–20.

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

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

Greidanus, R. (2017). *Client-side Anti-cheat in Online Games: Legal Implications from a Privacy and Data Protection Perspective*. Tese de doutorado, 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*, páginas 1–137.

Jonnalagadda, A., Frosio, I., Schneider, S., McGuire, M. e Kim, J. (2021). Robust vision-based cheat detection in competitive gaming. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 4(1):1–18.

Joshi, R. (2008). Cheating and virtual crimes in massively multiplayer online games. *Royal University of London*.

Kaur, N., Azam, S., Kannoorpatti, K., Yeo, K. C. e Shanmugam, B. (2017). Browser fingerprinting as user tracking technology. Em *2017 11th International Conference on Intelligent Systems and Control (ISCO)*, páginas 103–111. IEEE.

Khalifa, S. (2016). *Machine Learning and Anti Cheating in FPS Games*. Tese de doutorado, University College London, Londres, Inglaterra.

Khan, S. A. e Khan, R. A. (2012). A framework to quantify security: complexity perspective. *International Journal of Information and Education Technology*, 2(5):439.

Lan, X., Zhang, Y. e Xu, P. (2009). An overview on game cheating and its counter-measures. Em *2nd International Symposium on Computer Science and Computational Technology*, página 195, Huangshan, China.

Lehtonen, S. et al. (2020). Comparative study of anti-cheat methods in video games. *University of Helsinki, Faculty of Science*.

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

Matthies, C., Pirl, L., Azodi, A. e Meinel, C. (2015). Beat your mom at solitaire — a review of reverse engineering techniques and countermeasures. Em *2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, páginas 1094–1097, Potsdam, Alemanha.

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

Microsoft (2023). Sysinternals. `https://learn.microsoft.com/en-us/sysinternals/`.

Microsoft (2024). Determine allow-deny list and application inventory for software restriction policies. `https://tinyurl.com/4ebf59hs`.

Moghaddasi, H., Sajjadi, S. e Kamkarhaghighi, M. (2016). Reasons in support of data security and data security management as two independent concepts: a new model. *The open medical informatics journal*, 10:4.

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). Riot games terms of service. `https://tinyurl.com/36c9p94d`.

Ronkainen, W. (2021). *Prevention vs Detection in Online Game Cheating*. Tese de doutorado, University of Oulu, Oulu, Finlândia.

Russinovich, M. (2023a). Process explorer v17.05. `https://tinyurl.com/4rd378r8`.

Russinovich, M. (2023b). Process monitor v3.96. `https://tinyurl.com/5n6bkcha`.

Schellens, K. (2021). Addressing privacy in software architecture. Dissertação de Mestrado, 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*, páginas 428–438.

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

Telimaa, E. (2021). Hacking detection in unreal engine 4. *Karelia University of Applied Sciences*.

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

Webb, S. D. e Soh, S. (2007). Cheating in networked computer games: A review. Em *Proceedings of the 2nd International Conference on Digital Interactive Media in Entertainment and Arts*, página 105–112, New York, NY, USA. Association for Computing Machinery.

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

Willett, S. e Hogan, M. (2019). Cheating the network: how gamers play the infrastructure. *Canadian Journal of Communication*, 44(3):439–453.

Wireshark (2023). Wireshark. `https://www.wireshark.org/`.