INSTITUTO POLITÉCNICO DE BEJA

Escola Superior de Tecnologia e Gestão

Mestrado em Engenharia da Segurança Informática

Batota e Segurança em Jogos *Online*

Recolha e Análise de Comunicações do *Counter-Strike: Global Offensive*

Bruno Manuel Gomes Pereira

2020

INSTITUTO POLITÉCNICO DE BEJA

Escola Superior de Tecnologia e Gestão

Mestrado em Engenharia da Segurança Informática

Batota e Segurança em Jogos *Online*

Recolha e Análise de Comunicações do *Counter-Strike: Global Offensive*

Elaborado por:

Bruno Manuel Gomes Pereira

Orientado por:

Professor Doutor Rui Miguel Soares Silva, IPBeja

2020

Resumo

Batota e Segurança em Jogos Online

Recolha e Análise de Comunicações do Counter-Strike: Global Offensive

Tem havido crescimento em jogos online competitivos com recompensas monetárias, onde a segurança pode ser comprometida para se obter vantagens injustas. Assim, há necessidade de compreender melhor as vulnerabilidades de jogos e consequentes falhas de segurança, que levam à batota. Uma taxonomia simplificada de métodos de batota, baseada em tentativas passadas de outros autores, é fornecida. Para encontrar possíveis falhas de segurança, as comunicações em rede do Counter-Strike: Global Offensive, tanto em ambiente de laboratório como real, foram capturadas e analisadas. Os dados obtidos, que incluíram uma falha nas comunicações do CSGO, foram reportados. Foram propostos possíveis mecanismos de segurança para proteger a integridade do jogo, baseados na informação derivada da captura de comunicações.

Palavras-chave: *Jogos Online, Segurança, CSGO, Comunicações, Rede, Taxonomia, Batota.*

Abstract

Cheating and Security in Online Games

Collection and Analysis of Counter-Strike: Global Offensive's Communications

There has been growth in competitive online games with monetary rewards, where security may be compromised in order to obtain unfair advantages. Therefore, there is a necessity of better understanding games' vulnerabilities and consequential security flaws, which lead to cheating. A simplified taxonomy of cheating methods, based on other authors' previous efforts, is provided. In order to find possible security flaws, Counter-Strike: Global Offensive's network communications, both in a laboratory and real setting, were collected and analysed. The obtained data, which included a security flaw in CSGO's communications, was reported. Possible security mechanisms to protect the game's integrity are proposed, based on the information derived from the collection of communications.

Keywords: *Online Games, Security, CSGO, Communications, Network, Taxonomy, Cheating.*

Agradecimentos

Agradeço ao Professor Rui Silva pela orientação concedida no decorrer desta dissertação, e também pelas oportunidades que me propôs durante todo o percurso do Mestrado.

Um obrigado à minha família pela paciência e apoio demonstrado em toda a minha vida académica.

Um agradecimento aos meus colegas do Mestrado, e em especial ao João Orvalho, parceiro de grupo em muitos projetos e um amigo de confiança.

Índice

Índice de Figuras

Abreviaturas e Siglas

CSGO	*Counter-Strike: Global Offensive*
TCP	*Transmission Control Protocol*
UDP	*User Datagram Protocol*
P2P	*Peer-to-peer*
C/S	Cliente-Servidor
MMORPG	*Massively Multiplayer Online Role-Playing Game*
MMO	*Massively Multiplayer Online Game*
RTS	*Real-time Strategy*
FPS	*First Person Shooter*
NES	*Nintendo Entertainment System*
DoS	*Denial-of-Service*
DDoS	*Distributed Denial-of-Service*
DMA	*Direct Memory Access*
MITM	*Man-in-the-Middle*
ESP	*Extra-Sensory Perception*
ICE	*Information Concealment Engine*
VAC	*Valve Anti-Cheat*
IP	*Internet Protocol*
SimSIC	Simpósio sobre Segurança Informática e Cibercrime
GQUIC	*Google Quick UDP Internet Connections*

Capítulo 1

Introdução

A indústria dos videojogos é uma constante do mundo atual. Atrai mais de 2.5 mil milhões de jogadores por todo o planeta [6], gerou quase 135 mil milhões de dólares em 2018 e espera-se que esse valor chegue aos 180 mil milhões até 2021 [7]. Dentro deste valor encontra-se o mercado de jogos *online* para computadores, com um valor estimado de 33.6 mil milhões de dólares para 2019 [6].

Alguns videojogos inserem-se no que se refere por *esports*, videojogos jogados competitivamente (por profissionais e/ou amadores) coordenados por diferentes ligas e torneios, onde jogadores podem pertencer a equipas ou organizações que são patrocinadas por várias empresas [8]. *Esports* não se restringem apenas a jogos no computador, incluindo também consolas e dispositivos móveis como telemóveis.

As receitas da indústria dos *esports* são estimadas entre 1.1 mil milhões de dólares no pior dos casos, e 3.2 no melhor cenário [6, 7]. Tal como nos desportos tradicionais, existe toda uma infraestrutura de apostas à volta dos eventos, sendo expectável que sejam apostados, no total, 23.5 mil milhões de dólares em 2020 mundialmente, em casas de apostas [6], algo justificado com o número de espetadores, que em 2018 chegou a aproximadamente 380 milhões [9].

Com tanto dinheiro à volta deste mercado, torna-se crucial garantir a integridade dos jogos *online*, por forma a evitar que jogadores recorram a *software* malicioso para tirar partido de vulnerabilidades que lhes permitam obter vantagens injustas, dentro do jogo, sobre jogadores legítimos. Estas situações são mais graves quando se inserem em contexto de *esports*, em torneios com recompensas monetárias, mas mesmo em circunstâncias casuais o uso comum de batotas pode comprometer a reputação dos jogos, provocando perdas financeiras.

O objetivo da presente dissertação passa por compilar um conjunto de informações úteis relativamente aos problemas de segurança associados aos jogos *online*, bem

como a análise de comunicações de um jogo específico - *Counter-Strike: Global Offensive* (CSGO) - na tentativa de encontrar falhas de segurança.

Esta dissertação, além deste primeiro capítulo, está estruturada da seguinte forma: capítulo 2 intitulado "Contextualização", que faz um enquadramento de conceitos associados aos jogos *online* e sua segurança, incluindo arquitetura de comunicações, protocolos de comunicação, a história do *cheating* em contexto cultural, motivações de batoteiros, métodos (formas) de alcançar vantagens imprevistas em jogos *online*, tipos de batota, ou seja, quais as vantagens específicas mais comuns em questão e, por fim, mecanismos de segurança implementados pelos desenvolvedores de jogos; o capítulo 3, "Trabalho Relacionado", onde são expostas as principais fontes de informação que serviram como base para o projeto, é feita uma análise crítica do conteúdo retirado das fontes e é apresentada a hipótese de investigação desta dissertação; no capítulo 4, intitulado "Implementação", é definido um modelo de jogos *online* criado para servir de base de trabalho, são teorizados possíveis pontos vulneráveis que podem ser observados no modelo e discute-se a escolha das comunicações em rede como tema a aprofundar; no capítulo 5, "Captura e Análise de Comunicações do CSGO", abordam-se a recolha de comunicações do CSGO com recurso à ferramenta *Wireshark*, assim como a análise dos pacotes recolhidos e a discussão dos dados obtidos, onde estes são analisados numa perspetiva de segurança, com o intuito de detetar falhas, associar possíveis consequências a essas falhas e propor medidas de segurança; finalmente, o capítulo 6, denominado "Conclusões", destinado a resumir o trabalho realizado, demonstrar conclusões obtidas a partir desse trabalho e propor temas para trabalhos futuros.

Capítulo 2

Contextualização

Neste capítulo irá ser feito o enquadramento de conceitos relacionados com jogos *online*, necessários para compreender e permitir contextualizar as análises efetuadas em capítulos posteriores.

A secção 2.1 irá abordar as arquiteturas de comunicações usadas habitualmente em jogos *online*, algo que auxilia na perceção de quais batotas, ou *cheats*, são possíveis em quais arquiteturas. A secção 2.2 explica as diferenças na utilização de TCP ou UDP nas comunicações de jogos *online*. Em 2.3, pretende-se contextualizar o surgimento da cultura de *cheating* em jogos *online*, com o objetivo de perceber que caminho poderá tomar. Na secção 2.4 explanam-se as possíveis motivações de batoteiros. Em 2.5 apresenta-se uma taxonomia para métodos de batota, numa tentativa de categorizar de forma mais fácil de entender as formas como os batoteiros atingem os seus objetivos. A secção 2.6 apresenta os tipos de batota mais comuns que podem ser alcançados recorrendo aos métodos descritos. Por fim, são apresentados mecanismos de segurança que são ou podem ser postos em prática no mundo dos jogos *online*, em 2.7.

2.1 Arquiteturas de Comunicações

Os jogos *online* podem ter diferentes arquiteturas no que respeita às suas comunicações, o que tem influência nos possíveis problemas de segurança a ter em conta. As mais populares são *peer-to-peer* (P2P) e cliente-servidor (C/S). A figura 2.1 é demonstrativa destas. Em arquiteturas P2P, todos os dados do jogo são atualizados por todos os computadores envolvidos e enviados diretamente para todos os jogadores [1], ou seja, a computação é distribuída por todos os *peers* [10]. Já a arquitetura C/S é mais tradicional no sentido em que existe um servidor central

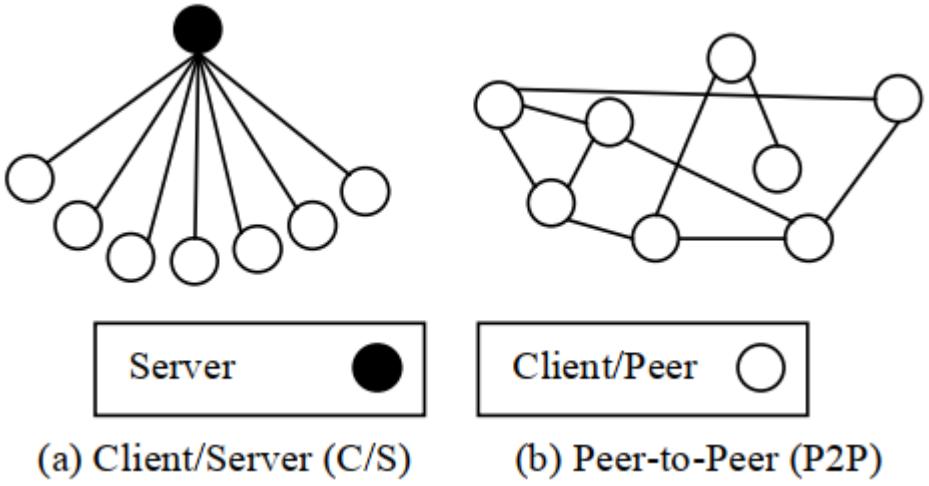

Figura 2.1: Arquiteturas C/S e P2P [1]

autoritário [10], à partida confiável, onde os jogadores se conectam. Os cálculos do jogo são realizados, idealmente, pelo servidor e não existem comunicações diretas entre as máquinas dos jogadores [11].

A arquitetura P2P apresenta algumas vantagens em relação à arquitetura C/S, nomeadamente a inexistência de um único ponto de falha, trocas de informação mais rápidas, e minimização do poder computacional e largura de banda necessários para simular o jogo [12, 1, 13]. Embora estes últimos pontos sejam bons argumentos a favor da escalabilidade de arquiteturas P2P, a verdade é que quanto maior o número de jogadores, maior a probabilidade de erros nas comunicações que influenciem a performance do sistema como um todo [14]. Como tentativa de manter a consistência do jogo, arquiteturas P2P podem usar um protocolo chamado *lockstep*, que permite evitar casos onde um jogador simula ter alta latência para poder ver uma jogada do adversário sem fazer a sua simultaneamente [15]. Neste protocolo, todas as máquinas esperam por uma resposta de todas as outras, e só depois simulam mais uma parte do jogo. Infelizmente, isto significa que basta apenas um jogador apresentar latências altas ou erros de comunicação para o jogo se tornar lento para todos os envolvidos [16].

As arquiteturas P2P são ainda propícias a falta de persistência, ou seja, à impossibilidade de novos jogadores participarem num jogo já a decorrer, além de a taxa de *frames* ter de ser igual para todos os participantes [14]. Já a nível de segurança, como a tomada de decisão é determinada pela máquina de cada jogador, isto poderá levar a que sejam tomadas decisões favoráveis para jogadores específicos [10], pois nenhuma das máquinas envolvidas é confiável. E mais, o facto de dados serem trocados diretamente entre jogadores invoca preocupações no que toca a possíveis

vulnerabilidades do *software* do jogo, que teoricamente poderiam ser exploradas com vista a atacar máquinas de outros jogadores que se encontrem na rede.

As arquiteturas C/S são mais simples e tornam mais fácil manter consistentes os dados do jogo para todos os jogadores [17]. O servidor contém a versão mestre da simulação do jogo e sincroniza todos os clientes, recorrendo a pacotes enviados em intervalos regulares [18]. Têm como desvantagem o facto de existir um único ponto de falha. Para além disto, embora um servidor oficial - controlado pelos desenvolvedores - seja confiável, por vezes é permitido aos membros da comunidade o controlo de servidores particulares. Isto pode causar que, caso exista alguma vulnerabilidade no *software*, surjam ataques aos jogadores que se conectam ao servidor. Um destes casos foi relatado a 11 de março de 2019, no jogo *Counter-Strike 1.6* [19].

Por vezes uma das máquinas dos clientes faz o papel de servidor para todos os outros jogadores, executando uma instância separada do *software*. Nestes casos, algumas das preocupações relativamente à segurança na arquitetura P2P tornam-se novamente relevantes, visto que ocorre novamente a característica de troca direta de dados entre jogadores (entenda-se, entre a máquina servidora - que também hospeda um jogador - e os restantes clientes).

Por forma a minimizar as desvantagens de ambas as arquiteturas, alguns desenvolvedores usam uma arquitetura híbrida, onde geralmente o servidor central do cenário anterior deixa de ser único, passando a ser replicado por várias outras máquinas, o que facilita a escalabilidade, pois a computação é repartida por todas as réplicas e torna-se viável a adição de mais servidores para servir mais jogadores, caso necessário.

2.2 TCP vs UDP

A escolha entre comunicações TCP e UDP recai, muitas vezes, sobre o tipo de jogo a que se aplicam. O TCP é um protocolo orientado à conexão [10, 20, 21], que conecta dois pontos por o que se pode percecionar como um "tubo" imaginário [22], onde um dos pontos envia uma *stream* de *bytes*, e o protocolo garante que o destinatário o recebe pela mesma ordem de envio [10]. Uma das características do TCP é o chamado *flow control*, cujo objetivo é igualar a velocidade de envio e receção de dados, por forma a evitar que pacotes sejam ignorados caso o destinatário esteja a lê-los a uma taxa menor que a taxa de envio [21]. Este protocolo contém verificação de erros, *handshakes* e *acknowledgements*, todas elas características que podem diminuir a performance de uma ligação. Com o TCP, mensagens longas podem ser repartidas por pacotes mais pequenos [21]. Assim, mesmo que uma

máquina tenha recebido um pacote, este poderá não ser processado, pois o protocolo pode decidir esperar por mais pacotes até que a mensagem esteja completa. Perda de pacotes poderá estender este período de tempo [10]. Jogos com uma componente forte de processamento de dados em tempo real não beneficiam da utilização do protocolo TCP por estes motivos. Se o *input* dos jogadores deveria ser processado o mais rapidamente possível, qualquer atraso irá tornar os dados recebidos irrelevantes para a simulação do jogo.

O protocolo UDP, por ser mais básico que o TCP, é a melhor opção para jogos onde baixa latência é a propriedade imperativa a atingir. Ainda assim, os desenvolvedores têm de trabalhar tendo em conta as desvantagens do UDP:

- Não há conceito de conexão (entre dois pontos);
- A ordem de chegada dos pacotes poderá ser diferente da de envio;
- Não existe *flow control*;
- Pacotes perdidos não são reenviados por defeito;
- A forma de verificar a validade dos pacotes (*checksum*) não é fiável.

Assim, o protocolo UDP é mais preferível em jogos do que o TCP, embora isto não queira dizer que seja impossível a utilização do segundo. Nos géneros *Massively Multiplayer Online Role-Playing Game* (MMORPG) e *Real-time Strategy* (RTS), é habitual a incorporação de TCP nas comunicações. Por outro lado, em *First Person Shooters* (FPS), como é o caso do *Counter-Strike: Global Offensive*, recorre-se a comunicações por UDP [23].

2.3 Uma Breve História do Cheating

Em [24], considera-se que a cultura da batota em videojogos começa com o primeiro *easter egg*, codificado num jogo para a consola Atari 2600, "*Aventure*" do ano 1978. Este segredo oferecia aos jogadores algo extra ao jogo, neste caso um ecrã com as palavras "Created by Warren Robinett" (apresentado na figura 2.2), um dos desenvolvedores do jogo. Para aceder ao *easter egg*, os jogadores tinham de encontrar a sala secreta e abri-la com recurso a um pixel no meio de uma parede. Ora, esta pequena piada criada por Robinett não tinha qualquer impacto no jogo, mas nos anos seguintes começou a emergir outro tipo de *easter eggs*.

Uma combinação de *inputs* permitia a jogadores de *Space Invaders* obter uma nova forma de disparar. *Easter eggs* deixaram de ser puramente decorativos, mas

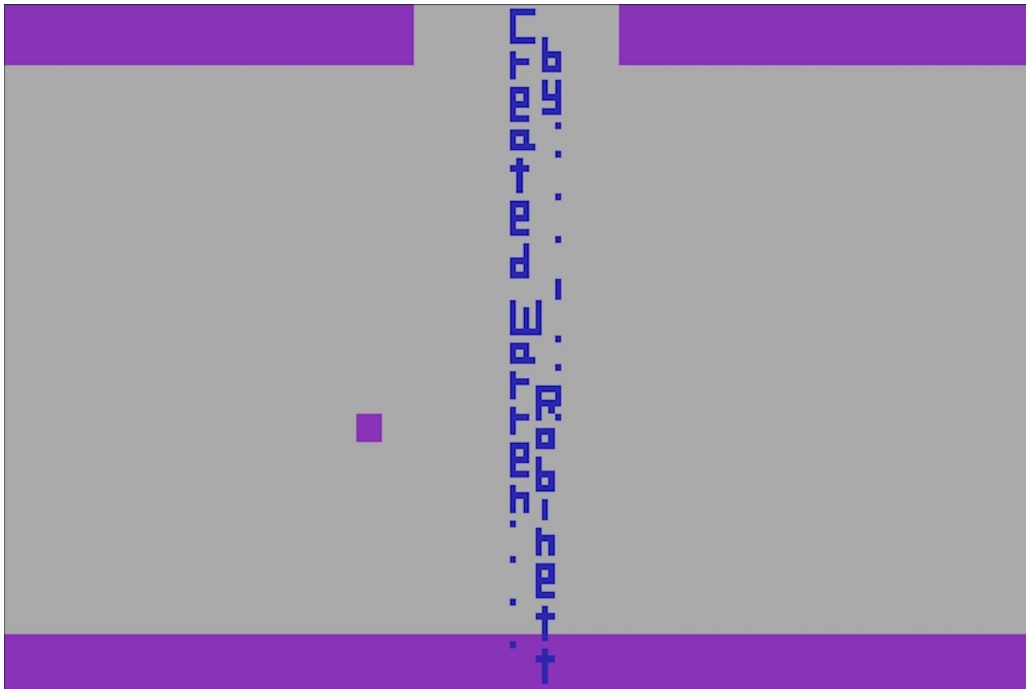

Figura 2.2: Easter egg de Robinett [2]

tornaram-se funcionais, oferecendo aos mais curiosos novas formas de interagir com o jogo. Este tipo de segredos evoluiu na direção dos *cheat codes*, que ainda hoje são usados em jogos para um jogador (*singleplayer*), ou para diminuir a dificuldade ou torná-los mais divertidos [25]. Numa era em que a tecnologia estava ainda bastante longe da atual, e, claro, sem o acesso ubíquo à Internet, existia toda uma indústria à volta de revistas com guias/manuais de jogos, cujo intuito passava também por partilhar informações sobre os *easter eggs* e *cheat codes* aos jogadores por todo o mundo [24].

Com a evolução das tecnologias, no entanto, os jogadores pretendiam ir além das capacidades dos *cheat codes*, e começaram a surgir soluções de *hardware* para atingir estes objetivos. Nos Estados Unidos da América, o primeiro produto popular deste género foi o *Game Genie*, que podia ser usado na consola *Nintendo Entertainment System* (NES) para introduzir diferentes códigos nos seus jogos, com efeitos que iam desde um número de vidas infinito até habilidades que normalmente não seriam possíveis. Isto era alcançado ao alterar alguns *bits* de informação enquanto os jogos eram carregados para a memória. Estas soluções ganharam alguma popularidade e foram evoluindo, passando pelo bem conhecido *GameShark* [26, 27] e até recursos mais sofisticados, como *chips* modificados para a *Playstation* original da *Sony*, que se tornaram a forma principal de fazer batota nas consolas sucessoras (*Playstation 2* e *Xbox*).

Quando a Internet se tornou um ponto importante nos videojogos, o aparecimento dos modos multijogador *online* levou a que a funcionalidade dos *cheat codes* fosse removida nestes. No entanto, para além desta medida raramente existiam outras formas de combate à batota [25]. Isto levou a que muitos jogos sofressem uma praga de batoteiros, levando às vezes a que fosse necessária a utilização de algum tipo de batota para apenas tornar o jogo mais "justo". Eventualmente, devido ao efeito negativo nos seus jogos, os desenvolvedores e editoras introduziram tentativas de tornar os seus jogos legítimos, e desde então o combate a este fenómeno tornou-se um dos esteios dos jogos *online*.

2.4 Motivações dos Batoteiros

Para melhor colocar entraves a batoteiros, é necessário perceber o que os motiva. Esta secção enumera e descreve resumidamente as motivações de quem utiliza batotas em videojogos *online* especificamente, visto ser mais relevante para o tema deste trabalho. Todos os motivos aqui descritos podem ser exacerbados se existir anonimato [24].

- Divertimento - alguns jogadores assumem ter gozo em estragar a experiência de jogadores legítimos. Este tipo de comportamento é caracterizado por alguns como "*grief play*" ou "*griefing*" [24];
- "Justiça¨- tal como referido na secção 2.3, alguns jogadores sentem-se obrigados a utilizar *cheats* por forma a compensar a fraca segurança do jogo, para, desta forma, poderem jogar de igual para igual contra os restantes jogadores [25, 24];
- Vingança - podem surgir, durante um jogo competitivo, situações em que um jogador reage de forma emocionalmente negativa, por exemplo, ao perder, e tenciona obter vingança contra o jogador que venceu, usando consequentemente alguma forma de batota [24, 28];
- Dificuldade - jogos MMORPG são muitas vezes altamente exigentes no que toca ao tempo necessário para evoluir uma personagem. Jogadores com menos tempo livre por vezes decidem usar *cheats* para automatizarem tarefas que lhes permitam chegar ao nível pretendido mais rapidamente [24, 28, 29];
- Competição - talvez a mais específica das motivações aqui expostas. Em determinadas comunidades de batoteiros, tornou-se hábito dispor à prova *cheats*

desenvolvidos para jogos competitivos ao colocá-los uns contra os outros, não só como forma de teste mas também para obter "fama" nessas mesmas comunidades [24], que advém da criação de um *cheat* de melhor qualidade. Este fenómeno tem o nome de "*hack vs hack*". Alguns exemplos disto ocorrem no CSGO e podem ser encontrados na Internet [30, 31].

- Dinheiro - a motivação monetária é possivelmente a mais comum, mas também a mais variada. Pode ser dividida em quatro componentes:

  - Venda de Bens Virtuais - bens virtuais relacionados com videojogos podem ser vendidos em mercados *online*, variando entre locais de troca oficiais, *websites* intermediários ou grupos organizados pela comunidade. O facto de ser necessário muito tempo para evoluir uma conta (principalmente em jogos MMORPG) pode levar jogadores a adquirir contas já evoluídas, mesmo que ilegitimamente [29]; *websites* criados especificamente para a troca e venda de bens virtuais, como o IGXE [32], podem ser usados por batoteiros que obtêm itens de forma ilegítima para ganhar dinheiro, como é o caso de Manfred, que o fez durante 20 anos [33];
  - Venda de *Cheats* - os desenvolvedores de *cheats* têm a possibilidade de os vender ao público, surgindo caso em que estes *softwares* são tratados como serviços mediante uma subscrição por um valor monetário;
  - Jogos de Sorte - *poker*, *blackjack*, *slots* ou roleta são apenas alguns dos jogos disponíveis para jogar *online*, com dinheiro real, totalmente legais. Tendo em conta que há grupos que se dedicam a desenvolver inteligência artificial capaz de vencer jogadores profissionais de *Texas hold'em* [34], torna-se extremamente provável a influência de *bots* em jogos de sorte *online* [28];
  - Torneios *Online* Remunerados - com o crescimento dos *esports* verificou-se também crescimento de torneios *online* com remunerações monetárias para os vencedores, encorajando jogadores a ultrapassar os mecanismos de segurança para obter vantagens injustas usando *cheats*.

2.5 Métodos de Batota

Diferentes tipos de *cheats* podem ser alcançados por diferentes métodos. Esta secção irá focar-se na forma como os batoteiros conseguem obter vantagens injustas,

e abrange a exploração de erros cometidos pelos desenvolvedores, abuso de ferramentas legítimas e até comunicações fora do âmbito dos jogos.

Os métodos apresentados nesta secção resultam da análise de diversas taxonomias (ver tabela 2.1), conjugando conceitos semelhantes numa nomenclatura mais atual, permitindo assim uma melhor apreensão dos diferentes métodos de batota. Apresentam-se de seguida duas considerações que foram tidas em conta na elaboração desta taxonomia:

- Em várias taxonomias, qualquer ataque que, de alguma forma, interfira na esfera do jogo é considerado um método de batota. Um exemplo disto é o uso de engenharia social para roubo de senhas de contas do jogo. Embora seja claramente um problema de segurança que deve ser tido em conta, ataques deste género não irão fazer parte da taxonomia apresentada neste trabalho, pois apenas se considera batota quando existe uma clara vantagem dentro do jogo em si, consequência de um dos métodos. O roubo de contas poderia, teoricamente, dar uma vantagem ao atacante na medida em que possivelmente iria obter uma personagem mais forte que a possuída anteriormente, mas a personagem poderá ter sido evoluída legitimamente pelo seu dono original;
- Métodos e tipos de batotas podem ser conceitos confusos. Neste projeto, considera-se que o método é a forma pela qual se obtém uma vantagem no jogo. O tipo de batota é exatamente qual a vantagem obtida.

Tabela 2.1: Taxonomias de Métodos (e Tipos) de Batota

Pritchard (2000) [5]
Reflex Augmentation
Authoritative Clients
Information Exposure
Compromised Servers
Bugs and Design Loopholes
Environmental Weaknesses
Smed et al. (2002) [11]
Packet and Traffic Tampering
Information Exposure
Design Defects
Jenkins (2004) [13]
Bug/Design Exploits

Network Attacks
Modified Binaries
Modified Packets
Falsified Connections
Yan et al. (2005) [35]
Cheating by Exploiting Misplaced Trust
Cheating by Collusion
Cheating by Abusing the Game Procedure
Cheating Related to Virtual Assets
Cheating by Exploiting Machine Intelligence
Cheating by Modifying Client Infrastructure
Timing Cheating
Cheating by Denying Service to Peer Players
Cheating by Compromising Passwords
Cheating by Exploiting Lack of Secrecy
Cheating by Exploiting Lack of Authentication
Cheating by Exploiting a Bug or Design Loophole
Cheating by Compromising Game Servers
Cheating Related to Internal Misuse
Cheating by Social Engineering
Consalvo (2007) [24]
Taking Advantage of a Glitch: Exploits and Duping Depending
Taking Advantage of People: Social Engineering
Taking Advantage of Code: Hacks, Bots, and Packet Sniffers
Taking Advantage of Third-Party Programs: Mods and Ends
Hoglund et al. (2007) [28]
Building a Bot: Automated Gaming
Using the User Interface: Keys, Clicks, and Colors
Operating a Proxy: Intercepting Packets
Manipulating Memory: Reading and Writing Data
Drawing on the Debugger: Breakpoints
Finding the Future: Predictability and Randomness, or How to Cheat in Online Poker
Webb et al. (2007) [1]
Game Level
Application Level

Protocol Level
Infrastructure Level
Laurens et al. (2007) [3]
Hard-coded Hacks
External Hacks
Environment Hacks
Hook Hacks
Packet Tampering
Lan et al. (2009) [36]
Modify game software or data
A third-party tool but not modifying games
Collude with other players or game operator
Exploit bug or loophole

2.5.1 *Bugs/Exploits*

Um dos métodos que tira partido apenas dos erros de desenvolvimento que se verificam apenas dentro do jogo, não no código [24, 37]. Os batoteiros tiram partido de erros de design do jogo sem manipular o código que o compõe. Claro, estes lapsos não são graves quando encontrados em jogos para um jogador, gerando até situações engraçadas, mas causam desequilíbrios em jogos multijogador. Geralmente, é o suficiente para o jogo não funcionar como devido, e leva a que os jogadores que ponham em prática o abuso destes erros obtenham vantagem sobre outros. Alguns dos mais conhecidos podem ser encontrados em [38].

O abuso de bugs, ainda que por vezes involuntariamente, também pode ser visto na história dos *esports*, como um dos casos mais infames a acontecer em 2014, num torneio de CSGO denominado "*DreamHack Winter 2014*". A ocorrência surgiu no jogo entre as equipas dos *fnatic* e *LDLC*, nos quartos-de-final do torneio. Tirando partido das mecânicas do jogo, é possível empilhar vários jogadores para levar os que ficam no topo da pilha a locais mais altos, geralmente impossíveis de alcançar de outra forma. A isto dá-se o nome de *boosting*, ou efetuar um *boost*. Os jogadores dos *fnatic* executaram um *boost* colocando-se em cima de pixeis invisíveis (referido por *pixelwalking* na comunidade), e com isto o jogador do topo conseguiu obter visão de grande parte do mapa do jogo, permitindo que ele matasse os adversários, vencendo rondas do jogo [39]. A organização do torneio decidiu que o mapa deveria

ser repetido, justificando a decisão com o facto de o jogador do topo do *boost* ser transparente, e até impossível de matar, por alguns ângulos [40]. A equipa dos *fnatic*, face a decisão dos organizadores, desistiu e concedeu a vitória aos *LDLC* [41].

Esta situação demonstra que embora este método seja dos menos intrusivos, ainda assim pode causar enormes danos a qualquer jogo, mas com especial foco nos multijogador, pois os jogadores inocentes sofrem com as ações dos batoteiros. Os *bugs* podem ser evitados com testes frequentes ao *software*, tendo em maior consideração novos conteúdos que sejam adicionados.

2.5.2 Ataques pela Rede

Apenas relevante para jogos *online*. Pode ser uma tentativa de acesso à base de dados que contém os dados do jogador em situações onde os bens virtuais de cada jogador são persistentes, para que os *hackers* possam alterar os dados, manipulando os registos dos inventários [42]. No entanto, pode também ser um ataque menos intrusivo e sofisticado, bem mais comum pela sua menor dificuldade de execução: um ataque de *Denial-of-Service*(DoS) [13] tanto ao servidor como, em alguns casos, aos clientes. Um ataque ao servidor significa que não existe vencedor, enquanto que ataques aos outros jogadores fará com que, no mínimo, as suas conexões ao servidor apresentem latências enormes, efetivamente retirando-os do jogo.

Grandes empresas na indústria dos jogos *online* estão cientes do problema do DoS. Um representante da *Valve Corporation* [43], empresa de jogos multimilionária, disse em público que ataques deste género "vão acontecer", e que os desenvolvedores têm de proteger os "jogadores e servidores de jogo, porque ambos são vulneráveis a ataques" [44]. A própria *Valve* já tomou passos para proteger, pelo menos, dois dos seus jogos deste tipo de ataques [45]. Um destes, o CSGO, foi alvo de bastantes ataques DoS quando sites de apostas começaram a atuar na área. O objetivo era atacar ou o servidor ou os jogadores como tentativa de cancelar uma aposta quando o resultado não era favorável [46]. Ora, nesta situação não existem batoteiros a tentar ganhar vantagens injustas dentro do jogo, mas é demonstrativa de que ataques pela rede são graves o suficiente para alterarem o rumo de um embate.

2.5.3 Modificações no Cliente

Possivelmente o método mais usado, existem uma série de modificações possíveis no lado do cliente que afetam diretamente o *gameplay*.

Análise e modificações da memória ocorrem do lado do cliente e são possibilitadas com o uso de ferramentas como descompiladores, desassembladores e *debuggers* [28]. Quando um jogo é carregado para a memória, um batoteiro pode analisá-la com o âmbito de encontrar variáveis cruciais e alterar ou registar os seus valores [36, 5]. No jogo *Warcraft III* batoteiros usavam este método para verificar informações importantes sobre o adversário [1]. A técnica "*hooking*" é útil para este tipo de situações, na medida em que permite intercetar determinadas funções para depois ler, controlar, ou injetar dados nos locais de memória do jogo [3].

É possível também aceder à memória através de hardware especificamente criado para este fim, o que resulta no que se chama de *Direct Memory Access* (DMA). Um exemplo deste método pode ser encontrado em [47].

A manipulação de controladores é também uma alteração que ocorre ao nível do cliente. Os primeiros *cheats* deste género surgiram no *Counter-Strike*, onde os controladores *OpenGL* e *Direct3D* eram o alvo [5]. Como os gráficos do jogo são renderizados pela placa, é possível manipular o seu controlador para, por exemplo, tornar determinadas paredes transparentes, permitindo ao batoteiro ver através delas, algo especialmente útil em FPS [35, 1].

Outra das modificações exequíveis na máquina do cliente envolve a alteração de ficheiros diretamente ligados ao funcionamento do jogo. Laurens chama a isto de *Hard-coded hacks* pois pode ser tão simples quanto alterar o código presente nalgum dos ficheiros instalados com o jogo, por exemplo, alterar um dos DLLs usados. Esta técnica é, no entanto, muito fácil de detetar [3].

A maior parte destas técnicas podem ser usadas em conjunto. Um exemplo seria alterar os ficheiros do jogo que determinam a cor do inimigo, usar a placa gráfica para realizar *scans* ao ecrã por pixeis com a nova cor do inimigo - escolhida pelo batoteiro - e movimentar o cursor do jogador para uma posição desses pixeis [3].

2.5.4 Modificações no Servidor

Caso um batoteiro obtenha o controlo de um servidor de alguma forma, poderá alterar as suas definições para obter vantagens [35]. Em casos onde os jogadores têm a possibilidade de hospedar servidores, e ainda para mais a possibilidade de os personalizar, é fácil perceber que seria possível inserir alterações vantajosas para o batoteiro. Isto por si só não seria um problema, pois os jogadores poderiam apenas evitar o servidor. No entanto, surgem situações em que os jogadores não têm perceção de que se encontram num servidor onde não estão de igual para igual com o hospedeiro, ou seja, o batoteiro [5].

2.5.5 Análise e Modificação de Pacotes

Tirando partido de ferramentas como o *Wireshark* [48], é possível analisar as comunicações entre servidor e cliente em jogos *online*. Esta ferramenta pode ser usada na máquina do jogador ou numa máquina adjacente que sirva como *proxy* entre o jogador e o servidor. De facto, é a base de um ataque *Man-in-the-Middle* (MITM).

O batoteiro pode inspecionar os pacotes que saem da sua máquina com destino ao servidor, e alterá-los com o intuito de causar no jogo uma resposta mais favorável [3, 35, 47]. Por exemplo, se num jogo FPS o batoteiro falhou um tiro mas gostaria de acertar, poderia alterar o pacote para que em vez de comunicar ao servidor o comando "*miss*", comunicasse o comando "*hit*" (ver figura 2.3). O inverso também poderia acontecer, bastando inspecionar as comunicações que são enviadas pelo servidor. Simplesmente intercetar determinados pacotes também é uma alternativa viável.

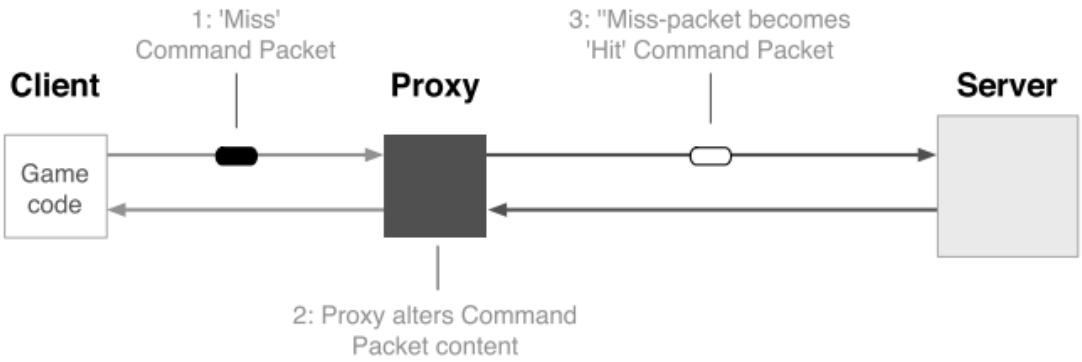

Figura 2.3: Um Exemplo de Modificação de Pacotes [3]

O batoteiro tem a opção de repetir o envio de pacotes quando lhe é favorável. A isto chama-se de *packet replay*. É particularmente útil quando uma ação é limitada por tempo. Esta técnica ignora as limitações e causa a execução da ação quantas vezes o batoteiro desejar [11].

Em jogos onde grande parte do estado do jogo é enviado pela rede (geralmente quando existem demasiados cálculos que, se deixados apenas ao servidor, afetariam a *performance*), não é necessária qualquer modificação de pacotes, pois basta analisá-los e usar algum *software* que crie um modelo do jogo baseado nos dados obtidos.

Por não modificar qualquer código do jogo, estas técnicas são muito difíceis de detetar [3].

2.5.6 Automatização

Um dos métodos mais básicos de fazer batota é a utilização de *software* que permita configurar quando e quantas vezes um *input* é enviado para o servidor automaticamente. Pode ser algo simples como pressionar o botão esquerdo do rato numa determinada posição do ecrã [5]. A isto geralmente se chama de *bot*. Este método é, tecnicamente, exterior ao jogo, visto que não manipula o código do jogo em si, apenas se serve da sua *interface*.

Podem ser também criadas *macros* que são, na sua essência, *bots* mais sofisticados, visto que executam *inputs* automaticamente consoante condições personalizadas relativamente ao jogo em questão [5]. Estes são especialmente populares em MMOs para obter mais facilmente itens valiosos nos jogos. Aqui cria-se um fosso entre jogadores legítimos que gostariam de usar estas macros e os batoteiros, porque os jogadores legítimos não possuem as aptidões necessárias para o desenvolvimento das macros, e estas podem não ser divulgadas publicamente [24].

A automatização de *inputs* é um método muito abrangente, na medida em que pode ser aliada a quase todos os outros métodos. Aliás, um dos exemplos apresentados em 2.5.3 já inclui *inputs* automatizados, quando um pixel de determinada cor é detetado e o cursor se move para as coordenadas correspondentes.

Outra forma em que a automatização pode ser usada é ao se abusar inteligência artificial (AI). Em jogos como o xadrez, o batoteiro pode deixar uma AI fazer as jogadas automaticamente por si. Já foram desenvolvidos vários programas capazes de competir com mestres de xadrez [35], pelo que o batoteiro detém uma vantagem enorme sobre o jogador casual.

2.5.7 Colusão

Este método é um dos mais difíceis de travar por poder contornar software, se desejado. Na colusão, dois ou mais jogadores que deveriam ser adversários, comunicam entre si para abusar ou de algum defeito no *design* do jogo, ou para passar informações que influenciam o jogo em si. A comunicação pode ser realizada por telemóvel, bem longe do alcance das medidas de segurança da maioria dos jogos [25]. Jogos de cartas, como o *poker*, são alvos fáceis para esta categoria de batota [5].

Jogos onde se ganham pontos e/ou itens com vitórias sobre outros jogadores podem geralmente ser abusados com este método [36].

O problema torna-se pior quando a colusão envolve também o administrador do servidor de jogo, pois os jogadores não só obtêm a vitória no jogo, mas possivelmente itens avançados e outros benefícios [36].

A colusão é a única categoria que se verifica tanto em jogos *online* como em desportos reais. Também nos *esports* já surgiram vários escândalos associados a este método [49], que resultaram em consequências graves para os participantes, desde coimas a suspensões vitalícias de torneios organizados por determinadas entidades.

2.6 Tipos de Batota

Existe uma grande variedade de tipos de batotas no mundo dos jogos *online*, que diferem consoante o género do jogo. Assim, esta secção contém uma breve explicação dos tipos de batota mais comuns.

- ***Wallhack*** - tipo de batota onde o batoteiro consegue ver através de paredes de alguma forma. Assim, ele consegue obter informação da localização dos adversários antes que eles o consigam ver, permitindo uma tomada de decisão mais favorável [3, 24]. Os *wallhacks* surgiram em primeiro lugar nos jogos do género FPS, onde naturalmente são muito úteis. Como muitas vezes, por motivos de *performance*, o servidor não pode tratar dos cálculos que decidiriam se um jogador pode ou não ver outro, essa informação é tratada pelo motor de jogo do lado do cliente, pelo que modificações no cliente (exploradas em 2.5.3) são técnicas usadas para alcançar este tipo de batota [28, 50];

- ***Aimbot*** - um batoteiro que use um *aimbot* consegue, na pior das hipóteses, apontar para adversários automaticamente (sendo, novamente, incrivelmente útil para FPS) [3, 24, 50]. *Aimbots* podem ter outras funcionalidades, tais como disparo automático, ativação ao pressionar um botão, ou ângulo personalizável para quando deverá ativar: um *aimbot* que ative em qualquer situação dá muito nas vistas - o adversário pode estar imediatamente atrás do batoteiro - pelo que uma ativação quando a *crosshair* (marca gráfica que simboliza o local para onde o jogador está a apontar) está mais perto do jogador adversário passará mais despercebida [28]. Este tipo de *bot* mistura métodos de automatização com modificações no cliente ou nas comunicações com o servidor [28, 50];

- ***Extra-Sensory Perception* (ESP)** - ESP é um tipo de batota que aproveita a exposição de informação ao cliente que ele não deveria aceder [3]. Pode tomar a forma de pontos num radar ou mapa que demonstrem a posição exata do adversário ou objetos de interesse, ou um *overlay* com informações detalhadas para além da posição [5]. A figura 2.4 demonstra um exemplo deste tipo de batota;

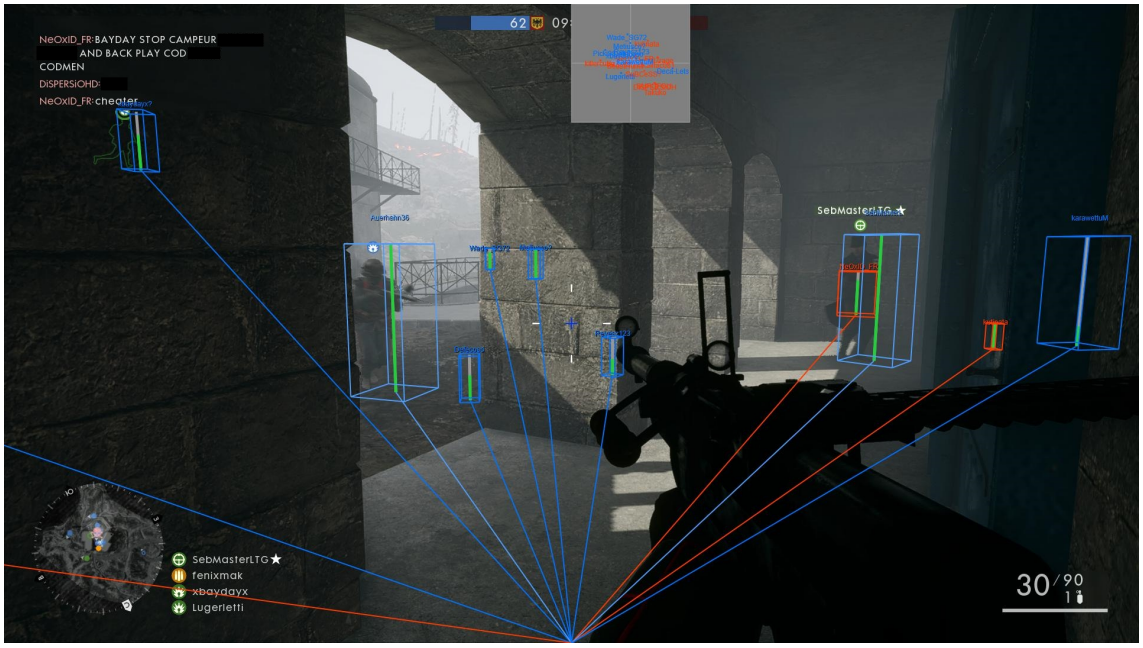

Figura 2.4: ESP no jogo Battlefield 1 [4]

- ***Triggerbot*** - é muitas vezes incluindo em *aimbots* e oferece a funcionalidade de enviar automaticamente o *input* de clique no botão esquerdo do rato. Em jogos FPS, isto significa que quando a *crosshair* se encontra nalgum ponto do corpo do adversário, a arma irá disparar automaticamente [51];
- ***Speedhack*** - como a designação indica, este tipo de batota torna o movimento do jogador extremamente rápido, ignorando os limites impostos pelas regras do jogo. O batoteiro pode, por exemplo, mover-se, explorar e obter itens mais depressa que jogadores honestos [52];
- ***Ghosting*** - como muitos dos jogos *online* possuem um modo de espetador para quando o jogador se encontra temporariamente fora de ação, é possível, através de colusão, passar informações sobre a posição de jogadores da própria equipa a intervenientes da equipa contrária. O modo de espetador permite seguir a ação do jogo através de uma câmara "livre" ou fixa em jogadores específicos [53].

2.7 Mecanismos de Segurança

Vistos os métodos e tipos de batotas disponíveis aos batoteiros, é importante entender quais mecanismos de segurança são implementados em jogos *online* para

prevenir, detetar e dissuadir o uso de *cheats* e manter a integridade do jogo intacta.

2.7.1 No *Design* do Jogo

Muita da proteção do jogo depende do *design* da sua estrutura [5]. Não se deve confiar no cliente e, se possível, os cálculos do jogo devem ser feitos do lado do servidor [50]. Aquando das comunicações com o cliente, deve ser minimizada a quantidade de informação partilhada. No entanto, isto seria a situação ideal e no mundo real torna-se muito complicado seguir esta direção, principalmente devido a questões de *performance* [50]. Assim, existem sempre vetores de ataque disponíveis para batoteiros que têm de ser abordados com alguma criatividade.

Para contrariar *wallhacks*, os servidores de jogo poderiam apenas enviar para o cliente as entidades do jogo que devem ser renderizados no seu campo de visão. Assim, não seria possível ver o jogador adversário através de paredes, pois o jogo do batoteiro não tem forma de obter essa informação [50]. Outra forma de abordar o problema passa por falsificar a posição os modelos do jogadores adversários que estejam fora da visão legítima do batoteiro. Por exemplo, se o motor de jogo calcular que entre o batoteiro e a vítima existe uma parede, o modelo da vítima não é renderizado na sua posição real, mas sim imediatamente atrás da sua câmara, ou seja, fora do seu ângulo de visão. Esta solução apenas se aplica a FPS.

Uma boa parte dos *cheats* desenvolvidos deriva da possibilidade de fazer engenharia reversa do jogo. Quanto a isto, os desenvolvedores podem tomar algumas medidas:

- *Packing* - método de compressão ou encriptação de um executável [54]. Ao importar binário que foi alvo deste método para um desassemblador, é pouco provável que o ficheiro seja legível. *Packers* personalizados são difíceis de contrariar, pelo que poderia ser um forte obstáculo à engenharia reversa [28];

- Ativação de Exceções - técnicas que verificam se o executável está a ser alvo de *debugging*, executando funções inesperadas se for o caso, como por exemplo um cálculo que resulta numa exceção [28];

- *Single-Step Timing* - embora menos fiável que as restantes abordagens, pode ser um obstáculo complementar. Este método consiste em analisar a hora do sistema e detetar se o tempo entre cada leitura é mais longa que o normal (um valor arbitrário). Se isto acontecer, o *software* assume que está a ser lido por um *debugger*, visto que estes atrasam a execução de um programa [28].

Um mecanismo de segurança simples de implementar é a verificação dos ficheiros de jogo de cada cliente, garantindo que todos os jogadores possuem os mesmos ficheiros e que estes são os mesmos que os do servidor [5]. Esta medida é ainda usada atualmente, podendo ser encontrada em jogos da *Valve* com o nome de "*Pure Servers*" [55].

Para evitar *bots* que funcionam externamente ao jogo (referenciados em 2.5.6), desenvolvedores têm a possibilidade de recorrer a *captchas* [28, 53]. No entanto, este método tem a desvantagem de influenciar a jogabilidade de jogadores honestos negativamente, pelo que provedores de jogos *online* são, de forma geral, reticentes no que toca às suas utilizações [42].

A comunicação entre jogador e servidor deverá ser segura por forma a evitar que ataques MITM sejam bem sucedidos. Assim, os pacotes trocados nas comunicações devem ser codificados e/ou encriptados, para que complique a vida a batoteiros que pretendam modificá-los para obter vantagens [5, 11, 28]. A título de exemplo, as comunicações dos jogos da *Valve* são cifrados com a cifra *Information Concealment Engine*, ou ICE [56]. No entanto, a troca de dados em grande parte dos jogos *online* é bastante rápida, pelo que a encriptação deve ser escolhida com precaução e não ser demasiado computacionalmente pesada, pois isto iria afetar a *performance* do jogo e os jogadores iriam sentir os efeitos [5, 28, 42]. Para além disto, não deve ser uma encriptação que dependa de qualquer outro pacote enviado, pois é provável que as comunicações se baseiem em UDP (como explicado em 2.2), ou seja, não há garantias de que todos os pacotes cheguem ao destino. No caso da codificação, esta pode ser feita com base numa *seed* resultante de algum aspeto arbitrário do jogo. Isto é útil pois não havendo troca desta *seed* nas comunicações, obriga os desenvolvedores de *cheats* a encontrá-la de outra forma [5].

2.7.2 Fora do *Design* do Jogo

Fora do *design* do jogo, existem alguns sistemas que podem ser postos em prática para descobrir, dissuadir ou castigar batoteiros. Sistemas de reputação [57] baseados em fatores como o número de horas jogado, número de queixas por parte de outros jogadores e comportamentos observáveis são usados como forma de emparelhar jogadores que têm reputações semelhantes, com a ideia sendo colocar batoteiros a jogar uns contra os outros.

A *Valve* utiliza um sistema de queixas ("*reports*") para o CSGO com o nome de *Overwatch* [58]. Este sistema permite que "membros experientes da comunidade" revejam queixas sobre jogadores específicos para determinar se estes *reports* estão

corretos e castigar os alegados batoteiros. Aliado ao *Overwatch*, encontra-se ativo o *VACnet*, um sistema de *deep learning* que obtém dados a partir do sistema anterior para analisar os comportamentos dos jogadores dentro do jogo, aprender a detetar comportamentos que apenas surgem em batoteiros e castigá-los [59]. Segundo John McDonald, um dos responsáveis pela *VACnet*, o sistema tem entre 80% e 95% de taxa de acerto [60]. A palavra final fica do lado do *Overwatch*, com pessoas reais, o que torna a probabilidade de um falso positivo castigar um jogador inocente muito baixa.

Para além da prevenção e deteção de batoteiros, podem-se aplicar métodos para além da simples suspensão do batoteiro. Se um *cheat* for popular no seu uso, cada vez mais pessoas o irão usar. Seguindo esta lógica, e partindo do princípio de que os desenvolvedores do jogo já o conseguem detetar fiavelmente com os seus mecanismos, atrasar a tomada de medidas poderá levar a que mais batoteiros sejam castigados de uma vez [3].

2.7.3 *Anti-Cheats*

Uma boa parte dos jogos *online* estão protegidos por um *Anti-Cheat*, um *software* que implementa uma série de medidas por forma a melhorar a segurança e manter a integridade do jogo. Os *Anti-Cheats* podem ser propriedade da empresa que desenvolve o jogo - por exemplo o *Valve Anti-Cheat* (VAC) [61, 62] ou o *Anti-Cheat* [63] integrado da Blizzard Entertainment [64] - ou podem ser produtos mantidos por terceiros (*3rd party*) - como o *FACEIT Client Anti-Cheat* [65, 66], *ESEA Client* [67] ou *Easy Anti-Cheat* [68]. Alguns dos mecanismos implementados por *softwares* de *Anti-Cheat* são os seguintes:

- *Scans* à memória - *scans* em tempo real à memória por forma a detetar *hooks* de *cheats* [3, 63] com assinaturas conhecidas [61, 62, 1], ou provas de alterações à memória a partir do exterior do código do jogo [3];
- Autorização - verificação se o ID associado ao jogador não está suspenso devido a transgressões passadas [3, 61];
- *Hashing* de conteúdos - comparar as *hashes* de ficheiros de jogo do cliente com as *hashes* expectáveis [3, 62];
- *Screenshots* - imagens do ecrã do jogo por forma a combater *cheats* visuais [3];
- *Scans* a APIs/Drivers - *scanning* da placa gráfica diretamente antes do jogo ser executado [3].

Capítulo 3

Trabalho Relacionado

A literatura existente no que toca a este tema é limitada. Artigos escritos nos últimos três anos que abordem o tema deste trabalho na totalidade são praticamente inexistentes, pelo que a estratégia passou antes por basear o trabalho em artigos bastante citados na comunidade científica, ainda que as suas datas de publicação sejam relativamente antigas. Nesta secção vão ser apresentadas as fontes que se consideraram mais úteis para a elaboração desta dissertação.

Pritchard publicou, em 2000, um artigo [5] tendo em vista orientar os desenvolvedores de jogos no que toca ao problema da batota em jogos *online*. Segundo o autor, na altura da publicação do artigo, poucas informações estavam disponíveis sobre o tema e os desenvolvedores, por norma, não partilhavam as suas experiências entre si com receio de revelar segredos. Pritchard tem como objetivo retirar o tema das "sombras" e estrutura o seu artigo com uma série de regras para os desenvolvedores. Aborda as motivações de batoteiros, os métodos que lhes permitem fazer batota em vários jogos, formas de defesa para evitar *cheats* e limitações de várias arquiteturas no que toca à batota em jogos *multiplayer*. Apresenta uma *framework* para a classificação de batotas em jogos *online*, explicando em detalhe algumas delas.

Em 2002, Yan *et al.* publicaram um trabalho [69] que se foca nos problemas de segurança em jogos *online*. Na altura, o principal foco dos desenvolvedores era a proteção dos seus jogos contra a pirataria. No entanto, o novo prisma dos jogos *online* alterou os requerimentos de segurança para os jogos. Estes jogos eram desenvolvidos com a tecnologia mais avançada disponível, mas os desenvolvedores eram negligentes relativamente a técnicas de segurança. Assim, os autores investigaram falhas de segurança que aconteceram ou poderiam vir a acontecer, introduziram técnicas de segurança para prevenção de *cheats* e discutem outros problemas de segurança, e ainda partilham uma taxonomia de batota em jogos *online*. Esta taxo-

nomia teve como inspiração a publicada por Pritchard [5], mas era da opinião dos autores que esta não cobria categorias de *cheats* suficientes.

Smed *et al.*, também em 2002, visaram aspetos de *networking* em jogos *multiplayer* no computador no seu artigo [11]. Os autores afirmam que, à altura da publicação, os problemas dos jogos *online* eram raramente focados em artigos científicos, apenas em revistas e conferências por pessoas na indústria do entretenimento. Neste artigo, os autores pretenderam identificar aspetos de *networking* relevantes para os jogos *multiplayer*, entre eles os recursos que limitam este tipo de jogos, conceitos de distribuição que incluem arquiteturas de comunicação e escalabilidade. Pelo facto da segurança *online* se ter tornado uma grande preocupação na indústria do entretenimento, os autores reconhecem os objetivos de proteção de informação sensível e garantia de justiça em jogos *online*. Apresentam três categorias de *cheating* e possíveis formas de prevenção.

No ano 2005, Yan *et al.* publicaram mais um artigo [35] sobre batota em jogos *online*. Nesta altura, o uso de batotas era desenfreado na Internet, mas ainda pouco compreendido. Neste, os autores melhoram a sua classificação sistemática de batota em jogos *online*, tendo em conta os seguintes pontos: as vulnerabilidades subjacentes (o que é alvo de *exploits*), as consequências das vulnerabilidades (que tipo de falha pode ser alcançada) e o responsável pela batota. A taxonomia destina-se a ser usada não só por especialistas em segurança, mas também por desenvolvedores, administradores e jogadores. Neste artigo, conclui-se que as quatro bases da segurança de computadores (confidencialidade, integridade, disponibilidade e autenticidade) não são suficientes para encapsular a batota em jogos, sendo crucial o aspeto de justiça. Trabalhar para garantir a justiça, segundo os autores, fornece uma perspetiva para se compreender o papel de técnicas de segurança no desenvolvimento e operação de jogos *online*.

Um livro [50] publicado em 2006, escrito por Armitage *et al.*, foca-se em conteúdos relativos a *networking* e jogos *online*. A razão para a criação do livro foi a falta de conhecimento de engenheiros de redes relativamente aos jogos que utilizam as suas redes, bem como a ignorância de desenvolvedores de jogos no que toca a como a Internet se comporta. De todos os aspetos abordados no livro, os mais relevantes para a presente dissertação são: os capítulos direcionados à história e evolução dos videojogos, que contêm informações sobre arquiteturas de comunicações em jogos *online*; o papel de protocolos de transporte, que coloca o TCP e UDP frente a frente consoante as suas utilidades em jogos em rede; uma secção sobre modelos de comunicação, *cheats* e mitigação dos mesmos, onde os autores consideram que a batota em jogos *online* é prevalente porque combina competição e anonimato. Nesta

última, apresentam exemplos de batotas do lado do servidor, do cliente e até sob a rede, discutindo técnicas de deteção e de desencorajar este fenómeno.

Consalvo escreveu um livro [24] lançado em 2009 muito completo sobre videojogos. Neste, a autora explana a história cultural de batota em videojogos, examinando como estes comportamentos começaram até o que é considerado batota em termos atuais. Depois, afasta-se um pouco dos jogos e foca-se mais nos jogadores, procurando explicar as motivações e perfil dos batoteiros, bem como as técnicas usadas por estes. Para isto, recorre a provas retiradas a partir de entrevistas, o aparecimento de tipos de batota populares e as reações da indústria dos jogos. O último ponto relevante para o presente artigo tem a ver com as medidas de segurança impostas pelos desenvolvedores de jogos, que surgiram como resposta à popularidade do *cheating* em jogos *online*.

Em 2014 foi publicado o artigo [17] de Chen que é, essencialmente, uma compilação de fontes sobre temas derivados de jogos *online*. Os autores perceberam que não eram claros quais os temas de pesquisa existentes para estudo na esfera dos jogos *online*. Assim, analisaram literatura relacionada e identificaram vinte e cinco categorias. Destas, as categorias "*Game cheating comprehension and prevention*", "*Cheating prevention by design*", "*Fairness preservation in gaming*"e "*Game bots detection and prevention*"são as relevantes para o presente artigo. A primeira categoria apresenta especificamente o impacto da batota em jogos com economias que funcionam dentro do jogo, consoante o valor atribuído pelo mercado a itens virtuais. A economia pode colapsar pois os batoteiros obtém itens de alto valor com pouco esforço. Estes itens geralmente apresentam melhores características que outros mais baratos, ou seja, tornam as personagens controladas pelos jogadores mais poderosas. Isto significa que os jogadores honestos não conseguem acompanhar e ou começam também eles a usar *cheats*, acelerando o declínio da economia do jogo, ou deixam de jogar por completo. Esta categoria possui também fontes para abordagens de prevenção contra *cheats*. Em "*Cheating prevention by design*", os autores expõem fontes que explicam que a prevenção contra batota deve ser realizada logo a partir da fase de *design*; deve ser um requerimento, não apenas uma forma de mitigação. Para a terceira categoria, os autores apresentam referências para a prevenção de batotas, mas também preservação de justiça para todos os jogadores no que toca a componentes técnicos do jogo, como a latência. Por fim, "*Game bots detection and prevention*"menciona a deteção de batotas automatizadas, recorrendo a ferramentas de modelação ou *CAPTCHAs*.

Por fim, em 2017 foi publicada uma dissertação de mestrado em Segurança da Informação [37] de Mikkelsen, onde o autor pretende rever soluções de segurança

da informação usadas em áreas distintas e encontrar abordagens que possam ser transferidas para a esfera dos jogos competitivos *online*. São explorados conceitos semelhantes aos encontrados no presente projeto, tais como as motivações dos batoteiro e as diferentes formas de batota existentes (na visão do autor). O artigo, por ser dos mais recentes usados na bibliografia da presente dissertação, foi usado principalmente como forma de comparar referências bibliográficas, existindo uma igual, [28], e outras semelhantes como fóruns sobre *hacking* em videojogos (por exemplo [30] e [70]) e documentação de *networking* em motores de jogos (ver [14, 71]).

Para além de artigos e livros publicados, também revistas *online* de videojogos, e as já mencionadas documentações para desenvolvedores, bem como fóruns de partilha de *softwares* destinados à prática do *cheating* foram visitados e revelaram-se úteis no desenvolvimento da presente dissertação.

3.1 Análise Crítica

O artigo de Pritchard [5] foi o primeiro grande contributo para um tema envolvido em segredos, numa altura em que os desenvolvedores de jogos tinham receio de partilhar informações quer entre eles, quer com o público, devido à possibilidade de os batoteiros adquirirem mais material para ataques a jogos *online*. Mesmo sendo, à data de escrita desta dissertação, um artigo com dezanove anos (foi publicado no 2000), mantém-se em grande parte atual e útil para quem pretenda entender melhor o mundo do *cheating*. O autor criou a primeira taxonomia relevante para classificações de *cheats* (apresentada na figura 3.1), algo que inspirou a criação de outras (listadas na secção 2.5) que, na opinião dos seus autores, melhor representavam o paradigma das batotas.

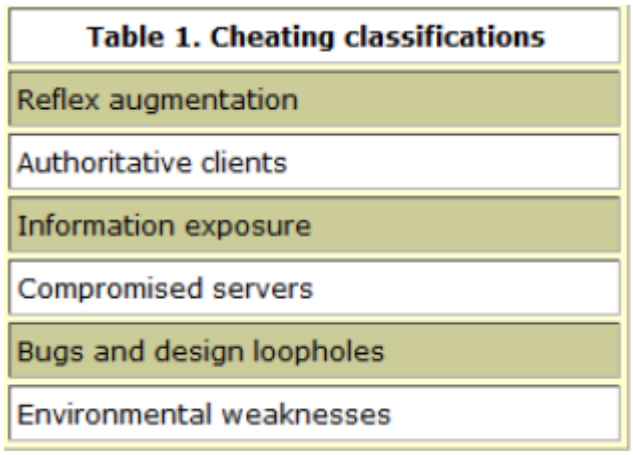

Figura 3.1: Classificação de Cheats por Pritchard [5]

Uma boa taxonomia é importante na medida em que organiza vários conceitos por forma a que possam ser implementados sistemas que os abordem. É uma forma não só de facilitar o entendimento de inúmeros tipos de batota que se enquadram nas classificações, mas também uma ferramenta útil para desenvolvedores de jogos, que podem assim começar a proteger os seus jogos logo a partir da fase de *design*, ou seja, do início do desenvolvimento.

Este artigo teve também importância por entrar em detalhe quanto à forma de como alguns *cheats* funcionavam a nível técnico, algo que deu o mote para outros autores fazerem o mesmo. Pritchard descreve o funcionamento de um *aimbot* que aproveitava as comunicações pela rede, e aborda as ferramentas usadas por desenvolvedores de batotas, como compiladores, desassembladores ou *debuggers*.

O artigo de 2005 de Yan et al. [35] apresenta uma taxonomia extremamente completa no que toca a formas de batota. Todas as categorias encontradas no artigo são baseadas em fontes reais, ou seja, nenhum dos cenários propostos são apenas teóricos. Por ser mais específica que a taxonomia de Pritchard, esta possui maior utilidade em termos de segurança. Apesar disto, a taxonomia parece apresentar um problema de clareza no que toca ao que é ou não batota. Segundo os autores, tirar partido de fraca autenticação para obter senhas é uma forma de batota. Apesar de ter influência num bem virtual relacionado com um jogo, não há necessariamente alterações vantajosas dentro do jogo de forma direta. Na opinião do autor da presente dissertação, considera-se ser demasiado semelhante a roubos de contas noutros contextos, pelo que não seria uma forma de batota ou ataque ao jogo, apenas um ataque genérico. O mesmo se aplica à engenharia social, que o autor classifica também como uma forma de batota. Ainda assim, a taxonomia de Yan et al. serve como uma boa base de trabalho, pois interliga vulnerabilidades, consequências e o autor dos ataques (apenas um batoteiro, um grupo de batoteiros, um administrador, etc.).

Armitage et al. exploram no seu livro [50] conceitos de *networking* relacionados com jogos *online*, sendo, dentro da literatura analisada, a fonte mais completa sobre a relação entre estas áreas. É indispensável para desenvolvedores, visto que oferece uma excelente base de redes sem ser demasiado específico. Consegue fazer uma transição da história relevante de jogos em rede para as configurações básicas da rede em si. O livro aborda temas como a escolha de protocolo de transporte (TCP ou UDP) de comunicações pela rede ou qual a arquitetura de comunicações adequada, informações úteis na escrita da presente dissertação. Este livro contém também uma secção dedicada ao *cheating*, onde se exploram diferentes tipos de batota incluídos numa classificação baseada nas entidades em que a batota atua - servidor, cliente ou comunicações em rede. Estes pontos servem essencialmente para descrever no que

consiste cada uma das batotas identificadas pelos autores.

A captura e análise de pacotes de jogos *online* também são temas abordados no livro, embora sejam apenas para fins estatísticos, por forma a garantir uma *performance* dos servidores adequada para os seus jogadores. Embora fosse útil que estes temas fossem expandidos numa perspetiva de segurança, ainda possuem aplicabilidade no que toca aos métodos usados para analisar os dados e à forma de filtrar as capturas de comunicações. Em suma, o livro de Armitage et al. foi uma boa contribuição no sentido de que aglomera uma grande quantidade de informação, e consegue juntar de uma forma coerente conhecimentos sobre a história dos jogos *online*, configurações das suas redes, batotas que podem ser encontradas nestes tipos de jogos, e recolha e análise capturas de comunicações, tudo com uma abordagem prática, mas científica.

3.2 Hipótese de Investigação

Após a análise de trabalhos relacionados, é justo dizer que existe uma boa base de conhecimento para se conseguir enveredar pelo aprofundamento de qualquer dos métodos de batota existentes atualmente. O que não existe, no entanto, são análises de capturas de comunicações de jogos *online* sob o contexto da segurança, procurando vulnerabilidades que possam ser exploradas por batoteiros.

Apesar de algumas fontes dedicarem partes à análise e modificação de pacotes, estes apenas se debruçam sobre o funcionamento de um ataque deste género na teoria e consequências que advêm desse mesmo ataque, que incluem, claro, os diferentes tipos de batota. Ora, este método de batota alberga alguns dos tipos de batota mais devastadores para a jogabilidade de jogadores honestos. É possível obter as posições de jogadores adversários ou configurar um *aimbot* que ofereça ao batoteiro pontaria e reações sobre-humanas, tornando o batoteiro praticamente invencível. É também pela rede que se pode afetar o servidor diretamente, usando, por exemplo, ataques de DoS, ou tentando tirar partido de vulnerabilidades de *software*.

Com a modificação de pacotes, os tipos de batota podem ser mais sofisticados e os batoteiros podem passar despercebidos em torneios com recompensas monetárias, obtendo claras vantagens contra competidores legítimos. Em vez de um *aimbot* que aponte certeiramente 100% das vezes para o inimigo, o batoteiro poderia usar algo que apenas melhorasse subtilmente a sua própria aptidão como complemento. Em vez de um *wallhack*, o batoteiro optaria por avisos sonoros que seriam executados quando um inimigo estivesse próximo. Estas são apenas algumas situações que podem ocorrer devido a este método de batota.

Por estes motivos, torna-se importante que, mesmo tendo em conta os mecanismos de segurança cujo objetivo é proteger as comunicações de ataques, se capturem e analisem comunicações de jogos *online*, por forma a detetar falhas de segurança que possam comprometer a integridade dos jogos em questão. Portanto, formula-se a seguinte hipótese de investigação: "será possível identificar falhas de segurança, na perspectiva da análise de comunicações em rede, no jogo *online* CSGO, que permitam a exploração de algum método de batota?".

Capítulo 4

Implementação

Este capítulo destina-se à exposição da fundação sobre a qual a recolha e análise de capturas de comunicações assenta. A secção 4.1 aborda a perspetiva de modelo de jogos *online* para a presente dissertação, baseada nos conceitos explicados no capítulo 2. Na secção 4.2, são teorizados pontos potencialmente vulneráveis que podem ser alvos de ataque no modelo de jogos *online* criado. A secção 4.3 aprofunda o tema das comunicações em rede de jogos *online*, com vista a perceber a importância de assegurar a integridade das mesmas, bem como o que implicam falhas de segurança.

4.1 Modelo de Jogos *Online*

Uma primeira fase na abordagem à segurança de jogos *online* requer a criação de um modelo que possa servir como base de trabalho. O primeiro ponto a tratar é a escolha da arquitetura de comunicações que o modelo de jogos *online* irá apresentar. Isto é necessário, pois diferentes arquiteturas comportam-se de formas distintas, como já referido na secção 2.1, o que significa que possuem diferentes possíveis pontos fracos.

A arquitetura C/S é a mais prevalente em jogos *online* [12]. Por este motivo, faz sentido focar nesta arquitetura. Assim, o modelo de jogos *online* imaginado irá ter um servidor central (poderiam ser mais, mas para análise de segurança apenas um basta) que irá manter o estado do jogo para todos os clientes que se conectarem. Isto é relevante porque significa que os jogadores nunca trocam comunicações pela rede uns com os outros de forma direta. Os clientes apenas se ligam pela rede ao servidor.

Outro ponto importante está relacionado com o agrupamento de jogadores em

equipas. Na presente dissertação, considera-se que batota em jogos *online* com vertente competitiva é mais grave do que noutros tipos de videojogos, visto que compromete não só a integridade do jogo, mas também de qualquer torneio organizado sobre esse mesmo jogo. Com a existência de torneios com recompensas monetárias, esta particularidade torna-se ainda mais relevante. Assim, o modelo de jogos *online* terá de ter em conta a possibilidade de competição entre equipas.

Considerando as fontes analisadas para o desenvolvimento da presente dissertação, é razoável admitir que os jogos competitivos com a maior variedade de batotas possíveis são, geralmente, do género FPS. Na verdade, uma boa parte dos avanços nos *cheats* verificados em jogos *online* surgiu neste género. Este tipo de jogos, geralmente, funciona com comunicações transportadas por UDP e não TCP. O modelo de jogos *online* irá, então, ter em conta esse aspeto e as suas comunicações em rede serão por UDP.

A figura 4.1 representa o modelo de jogos *online* tratado. O servidor central está representado por a). b) representa as comunicações em rede trocadas entre o servidor e os clientes, geradas pelo jogo. A letra c) caracteriza as máquinas dos jogadores que jogam individualmente. Por outro lado, d) representa máquinas de jogadores que jogam agrupados em equipas.

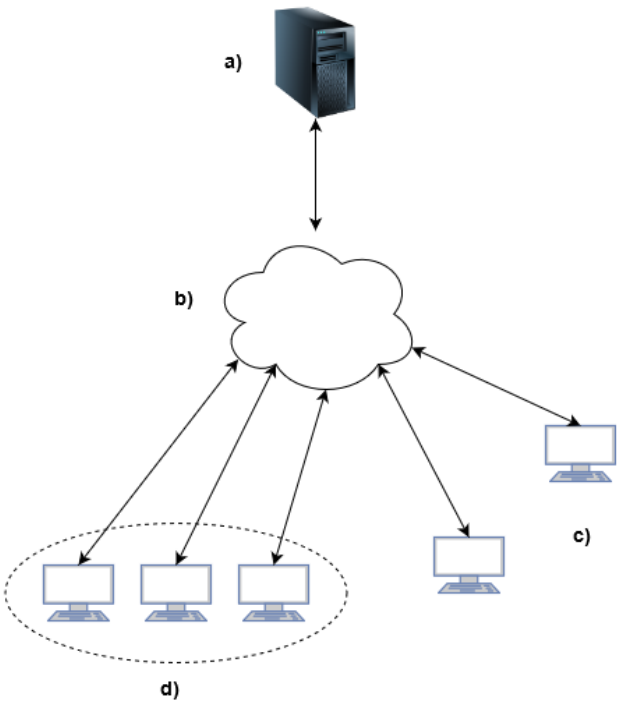

Figura 4.1: Modelo de Jogos *Online*

4.2 Pontos Potencialmente Vulneráveis

A partir do modelo de jogos *online* criado, pode-se agora tentar teorizar possíveis pontos fracos que se verifiquem nas vertentes que constituem o modelo: servidor, clientes e comunicações.

Entendem-se por pontos vulneráveis, pontos que possam ser focos de ataques que violem a integridade do jogo ou a integridade dos componentes envolvidos na estrutura. A figura 4.2 ajuda à compreensão desta formulação.

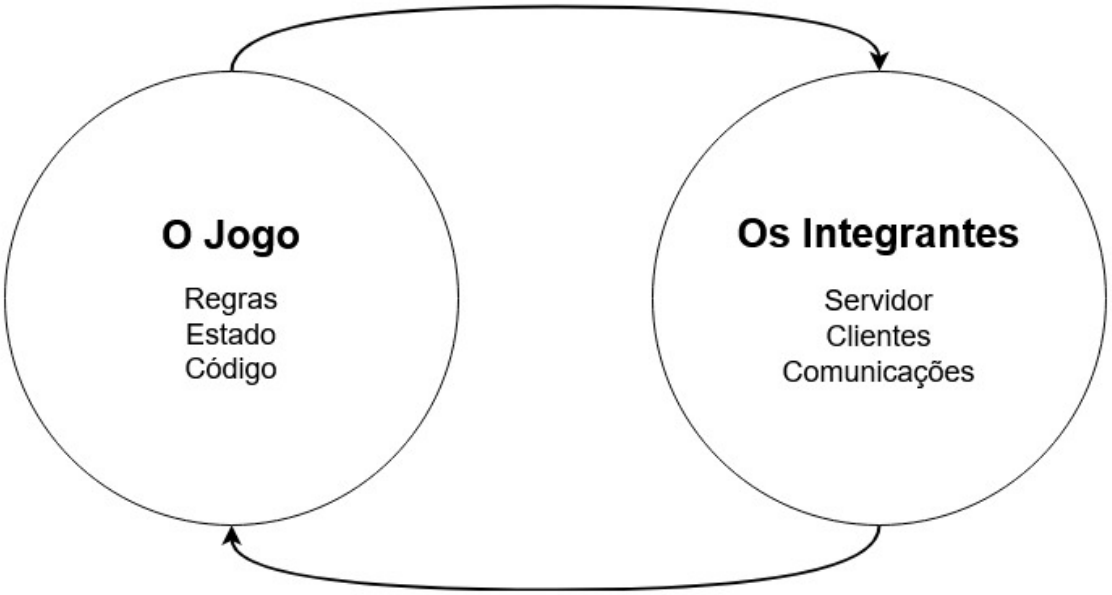

Figura 4.2: A Esfera do Jogo e dos Integrantes

A esfera do jogo pode ser caracterizada por um conjunto de regras, um estado da simulação mantido pelo servidor e o código que controla tudo isto. No fundo, é aquilo com que os jogadores interagem, enviando e recebendo dados a partir dos seus periféricos. Qualquer ponto no modelo que possa ser alvo de um ataque, modificando algum dos constituintes da esfera do jogo para proveito próprio deve ser considerado um ponto potencialmente vulnerável, pois é disto que consiste a chamada batota, ou *cheat*, cujo conceito não existe numa abordagem comum de segurança.

A esfera dos integrantes consiste nas diferentes vertentes do modelo: servidor, clientes e comunicações entre eles que são trocadas pela rede. Estas são a base para que o jogo possa funcionar.

No que toca à relação entre os pontos vulneráveis e as esferas indicadas na imagem, são apenas tidos em conta vertentes onde possam surgir ataques que modifiquem a esfera do jogo, necessariamente passando por um ou mais dos componentes

da esfera dos integrantes, ou ataques aos integrantes que tiram partido de características pertencentes ao jogo.

Em primeiro lugar, o servidor pode ser atacado por forma a tirar partido de uma vulnerabilidade no sentido mais tradicional, ou seja, comparável ao uso de um *exploit* num software. Isto pode ser feito para obter, por exemplo, controlo remoto da máquina onde a instância do servidor do jogo se encontra. Não é uma forma de batota (embora possa evoluir nesse sentido), mas dadas as características da estrutura do modelo, é algo que não se deve ignorar. Qualquer jogador que se ligue ao servidor pelo jogo consegue obter o endereço IP e porta da máquina do servidor, o que significa que o atacante pode ignorar a fase de recolha de informação que normalmente faz parte de um ataque deste género, facilitando a sua ocorrência.

O inverso também é verdade, ou seja, quem controla o servidor tem conhecimento dos IPs e portas dos clientes que se conectam para jogar. Caso o servidor seja controlado por um participante no jogo, este possui as informações necessárias para lançar um ataque "tradicional" a um ou mais clientes [19].

No servidor, a integridade do jogo pode ser posta em causa quando o proprietário do servidor desejar alterar definições por forma a garantir vantagens a jogadores, ou até a si mesmo, quando aplicável. É importante não esquecer que há jogos que permitem que jogadores hospedem os seus próprios servidores.

Os jogadores possuem várias alternativas ao nível do cliente para fazer batota, como já foi explicado na secção 2.5.3, tendo como alvo a memória reservada pelo jogo na máquina, controladores ou os próprios ficheiros que compõem o jogo. Por este motivo, os clientes são pontos vulneráveis no que toca à manutenção da integridade do jogo. Para além das modificações no cliente, os jogadores (batoteiros) podem tirar partido de erros de programação, ou *bugs*, como indicado na secção 2.5.1. No entanto, a prevenção para estes problemas só pode ser feita a partir do *design* do jogo e cabe apenas aos desenvolvedores, pelo que se considera que devem ser ignorados no modelo de jogo apresentado.

Outro ponto a ter em conta é a prática de colusão entre jogadores adversários para a obtenção de valores monetários, através de, por exemplo, manipulação de resultados para vencer apostas. É um problema de segurança ao nível do cliente, mas é incrivelmente difícil de enfrentar. As únicas medidas preventivas exibem-se através de castigos em livros de regras elaborados por organizadores de torneios, algo que não abrange todas as possibilidades. A obtenção de provas *a posteriori* é também complicada, pois requer colaboração com as entidades que gerem locais de apostas. Por estas razões, a colusão é algo que não deverá ser tido em consideração para este modelo de jogos *online*.

Ao nível da rede, surgem novamente preocupações que se enquadram com a segurança tradicional, nomeadamente ataques DDoS que podem ser facilitados com as informações obtidas através da ligação dos jogadores ao servidor. Tanto uns, como o outro são possíveis alvos para este tipo de ataque.

As comunicações transportadas pela rede servem para trocar dados sobre o jogo entre o servidor e os clientes. Estes dados podem incluir valores importantes como as coordenadas de um jogador, pontos de vida ou dano causado, por exemplo. As comunicações são passíveis de serem alteradas, pelo que constituem um ponto potencialmente vulnerável no modelo de jogos *online*.

O método da automatização pode ser aliado a qualquer dos outros métodos de batota, com a exceção da colusão, ou pode ser usado de forma isolada. Assim, pode incidir sobre o cliente, as comunicações, o servidor, ou qualquer combinação dos três componentes.

No seguimento deste exercício teórico, identificam-se então três pontos potencialmente vulneráveis: os clientes; o servidor; e as comunicações em rede entre eles.

4.3 Comunicações

Esta secção tem como objetivo explicar o raciocínio relativamente à escolha das comunicações em rede como o foco do próximo capítulo.

Dos três pontos potencialmente vulneráveis identificados, o cliente é o ponto de ataque mais comum. Isto porque ao nível do cliente existem várias formas como atacar um videojogo e obter vantagens que normalmente não seriam possíveis. Frequentemente são o foco dos desenvolvedores de batotas, pois caso o jogo não possua mecanismos de segurança de qualidade, as vantagens são alcançadas de forma relativamente acessível.

O segundo ponto potencialmente vulnerável é o servidor. Batotas que tiram partido deste componente podem ser muito básicas, como simplesmente alterar configurações através de ferramentas presentes no jogo. Um dos ataques mais comuns ao servidor será o DDoS, não só pelos estragos que causa mas também pela facilidade da sua ocorrência. Este ataque poderá ser detetado ao analisar as comunicações em rede, por ser por elas que se propaga.

Essas mesmas comunicações em rede são o terceiro ponto vulnerável identificado. Estas são alvo de análise e modificação de pacotes por parte de batoteiros, método que apresenta benefícios semelhantes aos ataques verificados no cliente. No entanto, a maior parte dos motores de jogo atualmente usados na indústria encriptam as comunicações em rede ou automaticamente, ou com pouco esforço necessário por

parte dos desenvolvedores que os usem. Por este motivo, os desenvolvedores de batotas poderão sentir-se desencorajados por seguir este caminho. Apesar disto, a encriptação automática das comunicações pode levar a que os desenvolvedores de jogos menosprezem este aspeto para se focarem apenas na defesa dos componentes do cliente. Isto poderá levar a que surjam erros como a encriptação tardia das comunicações, deixando passar alguns pacotes em claro. Tendo em conta que é difícil de detetar batotas que tirem partido de falhas de segurança nas comunicações, torna-se importante ter a certeza de que estas estão o mais protegidas possível.

Assim, o tema da segurança das comunicações em jogos *online* aparenta ser o mais aliciante e útil de explorar, pelos seguintes motivos:

- Ataques que ocorrem através das comunicações, ou incidem sobre elas, geralmente oferecem as mesmas vantagens dentro do jogo aos batoteiros que ataques ao nível do cliente;
- Podem surgir oportunidades para os batoteiros na análise de pacotes, caso os desenvolvedores de jogos menosprezem esta vertente;
- É um tema que pode ser focado mais facilmente, ao invés dos ataques ao nível do cliente, que se estendem por diversos aspetos dos jogos e da máquina do cliente;
- Um dos ataques graves ao servidor mais comuns - o DDoS - pode ser mitigado com melhores defesas ao nível das comunicações;
- É tradicionalmente mais complexo detetar batotas que incidem sobre comunicações.

Por isto, o foco da fase prática da presente dissertação serão as comunicações em rede entre clientes e servidor.

Capítulo 5

Captura e Análise de Comunicações do CSGO

Retirada a conclusão do capítulo 4 de que as comunicações em rede são um alvo apetecível para a execução de ataques, resta agora examinar em diferentes ambientes as capturas de comunicações decorrentes da atividade simulada por um jogo *online* ou que possa ser jogado em rede, em simultâneo com outros jogadores.

O primeiro passo para tal envolveu a escolha do jogo cujas comunicações iriam ser analisadas. Para isto, seria necessário um jogo que fosse de encontro aos seguintes critérios:

- Correspondência às características do modelo de jogos *online* definido na secção4.1:
  - Vertente multijogador em rede;
  - Arquitetura C/S;
  - Capacidade de agrupar jogadores em equipas;
  - Vertente competitiva, onde cada equipa trabalha para um objetivo;
  - Ser do género FPS;
  - Comunicações em rede transportadas por UDP.
- Atribuição de recompensas monetárias, através de um sistema inerente ao jogo ou torneios de *esports*;
- Existência de comunidades voltadas para o ato de fazer batota no jogo;
- Caso possível, existência de sites de apostas credíveis relativamente aos resultados de eventos sobre o jogo.

Tendo em conta estes critérios, o jogo escolhido foi o CSGO. O próximo passo foi a recolha de capturas de comunicações em rede do jogo selecionado. Isto foi realizado em dois ambientes distintos: ambiente controlado, onde foram recolhidas pequenas amostras de dados numa ligação fiável e estável, e ambiente real, sendo a dimensão das amostras muito maior, obtidas no decorrer de um torneio real. As próximas duas secções descrevem os cenários implementados para cada uma destas duas situações, apresentando-se depois uma secção com a análise de comunicações.

5.1 Captura em Ambiente Controlado

O objetivo de uma captura de comunicações em ambiente controlado é eliminar o maior número de variáveis possível que possam interferir com os resultados obtidos. Assim, a captura realizada ocorreu com recurso a uma máquina com o papel de servidor, uma máquina com o papel de um cliente e um *router MikroTik* usado para configurar uma rede local, comum a ambas as máquinas.

Foi usado um *software* desenvolvido por terceiros denominado *csgosl* [72], recomendado na documentação da *Valve* [71], para a configuração do servidor de jogo. A máquina cliente conecta-se ao servidor através de uma combinação de um IP - qualquer, desde que ambas as máquinas se encontrem no mesmo segmento de rede - e um porto, geralmente o 27015, visto ser o padrão para o tráfego de jogos da *Valve* em servidores dedicados [73]. O cliente não necessita de nenhuma configuração em particular, precisando apenas de uma cópia do CSGO com uma versão compatível com a encontrada no servidor.

Para a recolha das comunicações usou-se o *Wireshark*.

5.2 Captura em Ambiente Real

Para analisar as comunicações associadas ao jogo CSGO foi decidido obtê-las num contexto semelhante ao que se encontraria em torneios de *esport* dedicados ao jogo. Assim, a recolha de comunicações ocorreu durante o torneio de CSGO [74] organizado pelo Laboratório UbiNET [75], que faz parte do Instituto Politécnico de Beja, decorrendo nos dias 8 e 9 de maio de 2019, incluído no programa do X Simpósio sobre Segurança Informática e Cibercrime (SimSIC 2019).

No torneio participaram oito equipas de cinco jogadores, perfazendo um total de quarenta jogadores. A maior parte dos participantes utilizou os seus computadores pessoais, com um número limitado de jogadores a ter acesso, por necessidade, a máquinas preparadas pelos organizadores.

O regulamento do torneio foi baseado em regulamentos observados em vários torneios de CSGO a nível nacional, com o intuito de possibilitar a recolha de comunicações em condições semelhantes às apresentadas noutros torneios do género.

A topologia da rede usada no torneio está representada na figura 5.1.

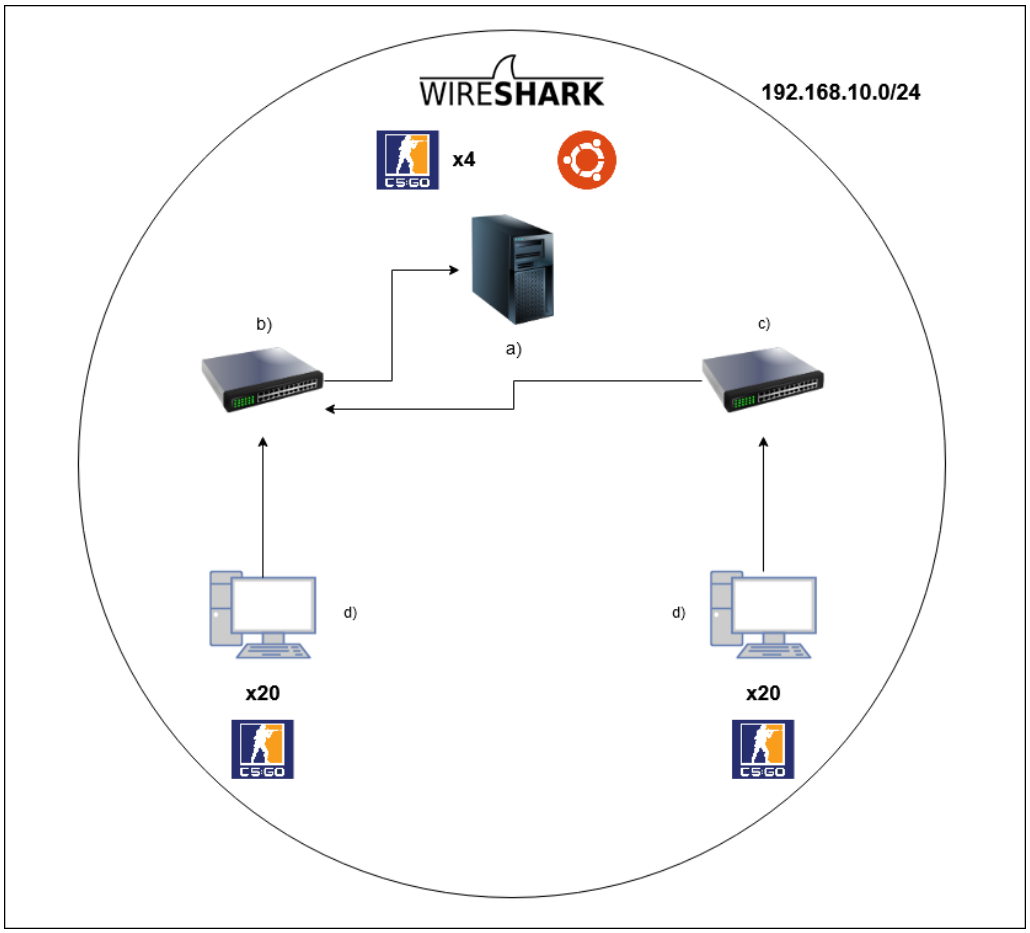

Figura 5.1: Topologia da Rede do Torneio: a) Servidor; b) Switch Principal; c) Switch Secundário; d) Computadores dos Jogadores

Todas as máquinas se encontram no segmento de rede 192.168.10.0/24. O servidor a) hospeda quatro servidores para o jogo CSGO no sistema operativo *Ubuntu* 19.04. Os quatro servidores foram configurados para escutar entre os portos 27015 e 27018, respetivamente, por motivos de organização. O endereço IP da máquina a) é o 192.168.10.254, e todos os servidores de jogo são acedidos por uma combinação desse mesmo IP, um dos quatro portos configurados e uma senha de acesso.

Foi novamente usado o csgosl [72] para a configuração dos servidores de jogo. As senhas de acesso escolhidas para os servidores foram "sim1sica", "simsic2b", "3simsicc"e "simdsic4". A versão mais atualizada do CSGO à data do torneio era

a 1.36.9.4 [76], pelo que foi essa a versão empregue.

Utilizou-se a versão 3.0.1 do *Wireshark* no servidor para capturar as comunicações entre as máquinas dos jogadores e os servidores de jogo. A primeira captura, do dia 8 de maio, possuía 12GB de conteúdo, algo que o servidor teve dificuldades em processar. Assim, as capturas do segundo dia foram divididas em ficheiros com tamanhos entre 150MB e 2GB.

Utilizaram-se dois *switches* - b) e c) - para conectar as máquinas dos jogadores, representados por d), ao servidor. O *switch* b) age como *switch* principal pois recebe a conexão de todos os jogadores.

Decorreram, sempre que possível, quatro jogos em simultâneo, com cada servidor de jogo a prestar serviço a duas equipas, ou seja, dez jogadores.

5.3 Análise das Comunicações

Para poder iniciar a fase de análise das comunicações recolhidas, foi primeiro necessário filtrar os ficheiros obtidos para garantir que apenas os pacotes relevantes restavam. Inicialmente, este processo realizou-se com recurso ao *Wireshark*, mas as limitações do software levaram ao uso de outras ferramentas. De facto, o *Wireshark* apresenta dificuldades ao realizar ações como carregar ou filtrar ficheiros com tamanhos superiores a 100MB [77]. A solução foi usar a ferramenta *tcpdump* como visto em [78], numa máquina virtual com sistema operativo *Ubuntu*. A filtragem foi feita em três partes: em primeiro lugar, manter apenas os pacotes referentes à troca de dados entre os jogadores e o servidor, ou seja, que tinham origem no segmento de rede 192.168.10.0/24 e destino a máquina do servidor nos portos 27015 até 27018 e vice-versa; depois, o resultado em ficheiros apenas referentes a uma conversação (comunicação entre um jogador e o servidor) como demonstrado na listagem 5.1; por fim, e caso necessário, dividir os ficheiros resultantes do segundo passo em ficheiros com tamanhos menores ou iguais a 100MB, por forma a facilitar a análise com o *Wireshark*.

```
1 #!/bin/bash
2
3 FILES=capturas/*.pcapng
4 # Array composto pelos ultimos octetos dos IPs detetados pelo
      Wireshark no ficheiro original
5 array=(193 194)
6
7 for a in "${array[@]}"
8 do
```

```
9  i=0
10 for f in $FILES
11 do
12  mkdir -p capturas/$a
13  tcpdump -r $f -w capturas/$a/"$a"'_'"$i".pcapng 'host
        192.168.10.'"$a"' and 192.168.10.254'
14  i=$((i+1))
15 done
16 done
```

Listagem 5.1: Exemplo de script para filtrar ficheiros .pcapng com tcpdump

O primeiro ponto registado a partir dos ficheiros obtidos é que se confirma que as comunicações no CSGO são feitas via UDP (ver figura 5.2). O *Wireshark* não foi capaz de detetar a forma de encriptação dos pacotes, pois o ICE não se encontra na lista de protocolos suportados.

Protocol	Percent Packets
Frame	100.0
Ethernet	100.0
Internet Protocol Version 4	100.0
User Datagram Protocol	100.0
Data	99.7
Real-time Transport Control Protocol	0.3
Pathport Protocol	0.0
ETSI Distribution & Communication Protocol (for DRM)	0.0
Distributed Computing Environment / Remote Procedure Call (DCE/RPC)	0.0

Figura 5.2: Comunicações do CSGO via UDP

Foram detetados alguns casos em que o protocolo GQUIC (*Google Quick UDP Internet Connections*) surgiu sobre UDP, como se pode ver na figura 5.3.

Protocol	Percent Packets
Frame	100.0
Ethernet	100.0
Internet Protocol Version 4	100.0
User Datagram Protocol	100.0
Real-time Transport Control Protocol	0.0
Pathport Protocol	0.0
GQUIC (Google Quick UDP Internet Connections)	100.0
Malformed Packet	50.2
ETSI Distribution & Communication Protocol (for DRM)	0.0

Figura 5.3: GQUIC nas Comunicações do CSGO

Recorrendo às funções de estatística do *Wireshark*, foi possível obter dados sobre comprimentos dos pacotes trocados durante as comunicações, bem como o número

máximo de pacotes trocados num segundo. A tabela 5.1 é demonstrativa dos primeiros.

	Contagem	**Percentagem**	**Mínimo**	**Máximo**
Comprimento dos Pacotes	22804389	100%	58	1242
0-39	0	0%		
40-79	350	0,0015%		
80-159	4047319	17,75%		
160-319	10629508	46,61%		
320-639	7596203	33,31%		
640-1279	531009	2,33%		
>=1280	0	0%		

Tabela 5.1: Dados sobre os Comprimentos dos Pacotes nas Comunicações do CSGO

Como a tabela indica, não existe qualquer troca de pacotes com as dimensões entre 0 e 39 *bytes*, bem como iguais ou superiores a 1280. Por outro lado, vemos que as comunicações são compostas, na sua maioria, por pacotes com dimensões entre 160 e 639 *bytes*, resultando em cerca de 80% das comunicações. O gráfico da figura 5.4 demonstra que os valores dos comprimentos das extremidades são muito raros de encontrar nas comunicações, comparativamente com os restantes.

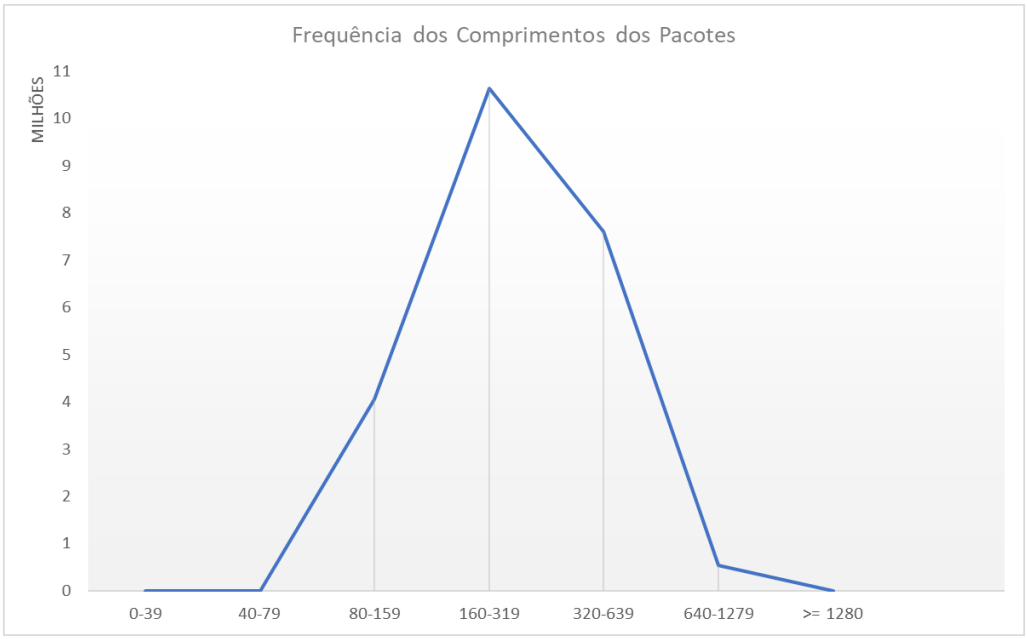

Figura 5.4: Frequência dos Comprimentos dos Pacotes

Já o número máximo de pacotes trocados num segundo, também referido por *Burst Count* no *Wireshark*, foi de 4886.

Por forma a detetar possíveis vulnerabilidades, decidiu-se importante analisar as primeiras trocas de pacotes entre as máquinas dos jogadores e o servidor. Do que foi possível perceber, o conexão ao servidor inicia-se da seguinte forma:

- A máquina do jogador envia um pacote ao servidor com a mensagem "qconnect";
- O servidor responde, enviando um pacote com a mensagem "connect";
- A máquina do jogador envia um pacote que contém a senha do servidor e dados sobre o jogador, como a alcunha e configurações do cliente.

Esta troca de pacotes está representada na figura 5.5.

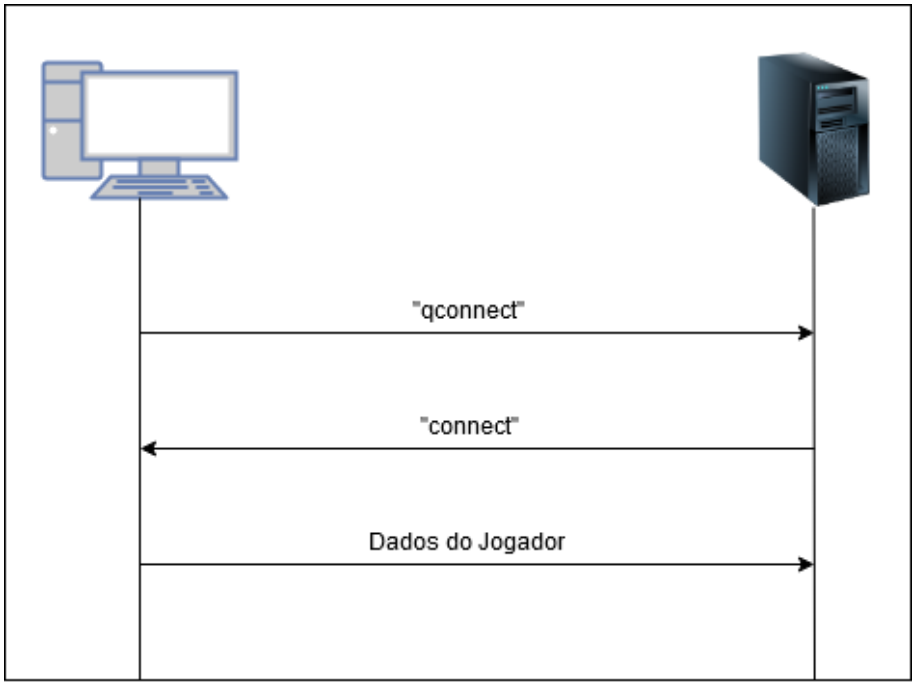

Figura 5.5: Conteúdo dos Pacotes Iniciais

Todos estes pacotes se encontram em *plain text*, ou seja, é possível ver os conteúdos sem a necessidade de decifrar os pacotes. A figura 5.6 apresenta os conteúdos dos primeiros pacotes trocados entre o cliente e o servidor.

Figura 5.6: Conteúdo dos Pacotes Iniciais

Podem-se identificar na figura 5.6 as mensagens "qconnect", "connect", a senha do servidor - "sim1sica" - e informações sobre o jogador - alcunha "KSED" e ritmo de envio (*rate*) de pacotes "786432" [79], por exemplo.

Após a troca destes três pacotes, o servidor responde com um pacote cujo conteúdo em hexadecimal se pode verificar na figura 5.7, e a partir desse ponto as comunicações deixam de ser reconhecíveis. Este comportamento foi registado tanto em ambiente controlado como em ambiente real.

```
f4 30 b9 8d d5 44 10 c3  7b 46 d9 86 08 00 45 00
00 40 25 fa 40 00 40 11  7e 62 c0 a8 0a fe c0 a8
0a 02 69 89 69 88 00 2c  96 8e ff ff ff ff 42 2e
30 30 30 30 30 30 30 30  2e 30 30 30 30 2e 30 30
30 30 30 2e 30 30 30 30  2e 30 30 30 30 2e 00
```

Figura 5.7: Resposta do Servidor aos Pacotes Iniciais

Observou-se também que após este pacote, o primeiro pacote enviado pelo cliente ao servidor contém dados em comum com pacotes de sessões diferentes. Nos testes realizados em ambiente controlado, foi detetada a semelhança salientada na figura 5.8.

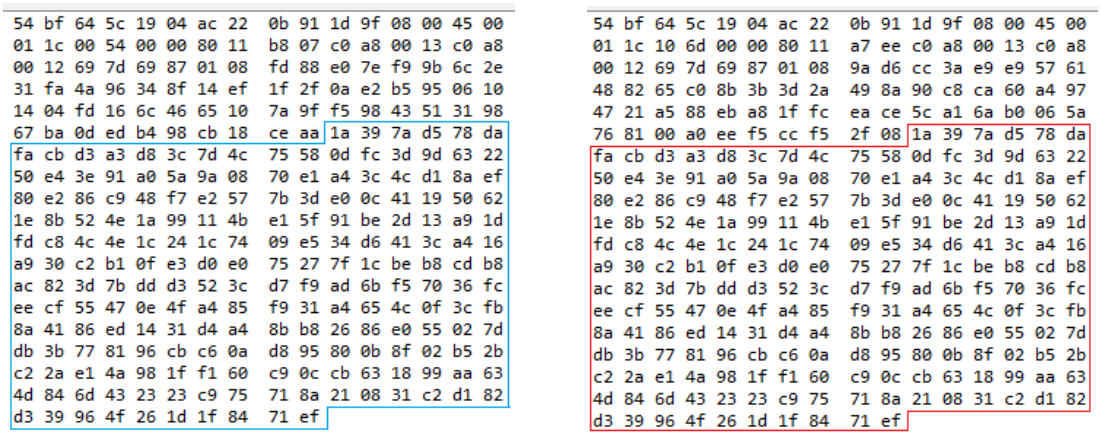

Figura 5.8: Semelhança no Conteúdo dos Pacotes em Ambiente Controlado

Já em ambiente real, obtiveram-se resultados idênticos com todas as comunicações de todos os jogadores, simplesmente com diferença nos dados em comum. Isto é exemplificado na figura 5.9, onde os dados riscados representam os dados que diferem entre os pacotes, dentro da seleção.

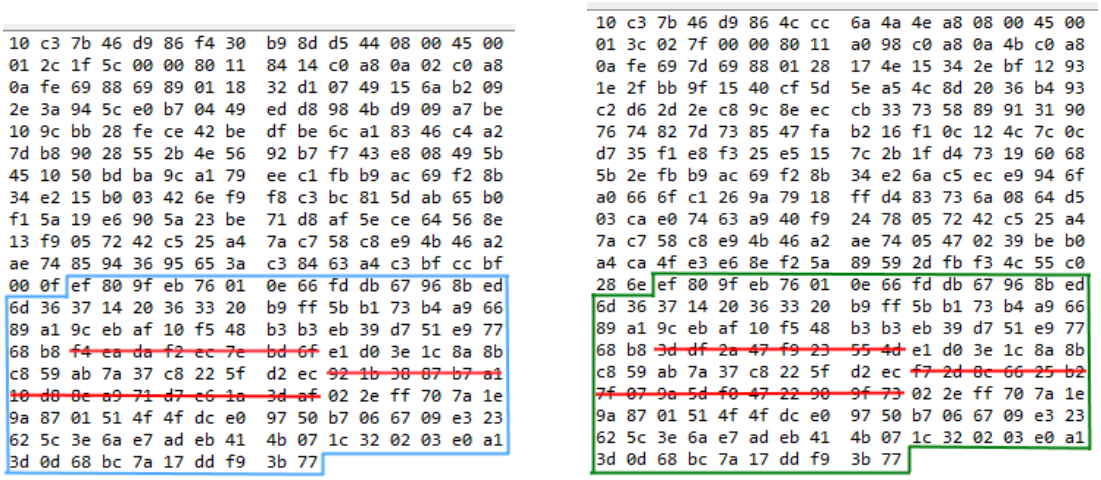

Figura 5.9: Semelhança no Conteúdo dos Pacotes em Ambiente Real

Outro ponto a notar é a repetição de oito *bytes* de dados na parte final de vários pacotes enviados pelo servidor ao cliente. Esta semelhança observa-se em sessões distintas. A figura 5.10 demonstra os *bytes* em ambiente controlado.

```
ac 22 0b 91 1d 9f 54 bf  64 5c 19 04 08 00 45 00
00 4c 03 dd 40 00 40 11  b5 4e c0 a8 00 12 c0 a8
00 13 69 87 69 7d 00 38  e2 e3 6b be 0f 30 69 10
42 17 25 ea c4 75 3f 2b  a5 02 d7 c3 61 47 4f d3
5a cf 45 92 b2 b3 b4 95  c3 a0 0f a4 b4 1f 9f 88
dd c9 00 eb 53 6e e3 39  06 a8
```

```
ac 22 0b 91 1d 9f 54 bf  64 5c 19 04 08 00 45 00
00 4c e2 54 40 00 40 11  d6 d6 c0 a8 00 12 c0 a8
00 13 69 87 69 7d 00 38  8c b9 ed 11 96 da 79 f8
f8 71 61 0c 3c a5 5e 92  18 19 a6 80 f3 3a be da
d9 0f e0 c0 ff 8c 45 26  3b f1 2e 8e 92 a6 be ce
c2 4b 00 eb 53 6e e3 39  06 a8
```

Figura 5.10: Semelhança nos Últimos Oito *Bytes* em Ambiente Controlado

A figura 5.11 exibe esses oito *bytes*, mas desta vez em ambiente real.

```
f4 30 b9 8d d5 44 10 c3  7b 46 d9 86 08 00 45 00
00 4c 26 1e 40 00 40 11  7e 32 c0 a8 0a fe c0 a8
0a 02 69 89 69 88 00 38  96 9a 42 37 54 2c 67 ae
97 69 e2 8b de e5 ff ed  ea 73 14 43 6b d9 2f cf
ae f8 38 b5 5b 95 4b 0c  43 49 a4 8b d4 1a 42 2b
6a e6 dd a5 f0 ac 78 be  f7 b9
```

```
80 fa 5b 25 3c 70 10 c3  7b 46 d9 86 08 00 45 00
00 7c 28 4e 40 00 40 11  7b 70 c0 a8 0a fe c0 a8
0a 64 69 89 69 7d 00 68  97 2c 37 35 18 a0 51 fc
43 78 c3 a0 78 57 be b2  56 6a 29 7c b7 69 4c ce
8e 4a 23 1b b7 43 fe ec  c0 c0 2f 33 8a f2 af 1b
e4 94 a6 15 d8 84 85 e6  cc 04 3d 92 f7 2f 58 dc
44 7c eb 8f 2b a1 7c e3  f2 10 7f 69 ba 33 73 c3
4c 11 b1 03 9f 0d 0b e6  68 1d a4 8b d4 1a 42 2b
6a e6 dd a5 f0 ac 78 be  f7 b9
```

Figura 5.11: Semelhança nos Últimos Oito *Bytes* em Ambiente Real

Por fim, detetou-se um padrão de *bytes* que aparenta ser um *header*. Este *header* ocorre de igual forma em ambos os ambientes de captura de comunicações, na medida em que se inicia com os quatro bytes "fe ff ff ff", seguido de um *byte* que incrementa ao longo da comunicação em que se encontra, até um valor máximo de "fe". Quando se incrementa este mais uma vez, o *byte* a seguir é incrementado uma vez e o ciclo recomeça. A figura 5.12 representa a incrementação do *byte* após do *header*.

```
f4 30 b9 8d d5 44 10 c3  7b 46 d9 86 08 00 45 00
00 b4 23 ea 40 00 40 11  7f fe c0 a8 0a fe c0 a8
0a 02 69 87 69 88 00 a0  97 02 fe ff ff ff 1a 01
00 00 02 01 a4 04 fc 03  61 0e bd a1 ba d8 8e 49
f3 49 19 cd e2 e3 16 e6  80 a3 70 a6 4d 8c 8f e2
45 2b 7e 18 a5 77 b3 e7  a8 1f c9 a7 37 e0 52 66
79 fc 38 21 16 8a 92 42  e5 31 2e 71 f5 df ba 0a
67 9b 4d e8 ed cc a1 eb  7c c3 bc 37 ec 41 c6 85
f3 ee 1c e2 c9 60 98 c8  80 03 da 94 ea e6 e9 4b
67 40 62 16 18 e8 12 a9  29 60 63 f9 fe d5 01 e5
92 36 81 57 13 13 eb c3  df eb 05 4c 8f 93 50 a7
2e 8a 3c ee 37 22 6c 07  71 94 80 e1 cd 08 5f 02
fa ed
```

```
f4 30 b9 8d d5 44 10 c3  7b 46 d9 86 08 00 45 00
00 e4 23 ee 40 00 40 11  7f ca c0 a8 0a fe c0 a8
0a 02 69 87 69 88 00 d0  97 32 fe ff ff ff 1b 01
00 00 02 01 a4 04 6a e2  1e d9 b5 bd a1 3c b8 a3
0b 26 17 40 79 95 54 a1  65 50 e1 ca b2 dd 61 34
4d dc 11 58 4f 15 b5 da  85 a0 61 26 51 28 dd b2
29 c7 a7 fc 12 58 14 7d  c8 57 9d 89 8e 7e a5 29
99 a2 4c dd d4 09 a3 43  34 9f da 0e 25 31 9a 36
92 57 b7 0e 80 58 68 ba  3e 8a 1c 19 75 5e ac 25
b4 fd dd 65 d6 2a 41 09  45 8a 04 16 d0 74 3e 11
48 43 6d f0 5a 4d d0 ed  a2 5d c1 c5 ed 5f e9 1e
4c 39 f6 5e c3 83 bf 22  09 bf eb ab cd 13 e9 f0
29 f9 49 8c f2 a1 e7 76  2f bc 62 f4 9b 0b 12 8e
da 9f fb ae 90 be 8d 7c  02 14 da 1c b4 46 6d c4
a9 69 a9 00 25 cd 34 4e  ca a8 6f 8a 2e 04 9c 3a
38 e2
```

Figura 5.12: *Header* de Quatro *Bytes*

Este comportamento termina de forma presumivelmente arbitrária, ou seja, não se verifica em toda a duração da comunicação e não termina sempre da mesma forma, mas o seu começo é sempre detetado momentos após o pedido de "connect" do cliente ao servidor.

Realizada a recolha e análise de comunicações do CSGO com o *Wireshark*, falta agora relacionar os dados obtidos com uma perspetiva de segurança.

O primeiro ponto a abordar tem a ver com a troca inicial de pacotes que se verifica quando um jogador se liga ao servidor. Acontece que os três primeiros pacotes estão totalmente expostos, na medida em que não são codificados nem cifrados. Um simples ataque *Man-in-the-Middle* é o suficiente para obter o senha do servidor a que o jogador se pretende conectar. Embora este tipo de ataque não se relacione com algum tipo de batota, não se deve ignorar. De acordo com a *Panda Security*, 52% dos utilizadores na Internet reutilizam a mesma senha para diferentes serviços [80]. Tendo este número em conta, esta falha de segurança encontrada nas comunicações do CSGO não só irá permitir o acesso ao servidor de jogo, como poderá alastrar-se para situações exteriores ao jogo. Se o atacante, numa fase de recolha de informação, descobrir contas de outros serviços pertencentes ao dono do servidor, poderá servir-se da vulnerabilidade para, na pior das hipóteses, roubar essas contas.

A razão para o terceiro pacote não ser encriptado não é clara; a *performance* que tanto se quer manter em jogos deste género não é relevante neste caso, pois o utilizador ainda se encontra no menu do jogo quando se liga ao servidor. Além disso, tudo indica que as comunicações são cifradas logo de seguida, enquanto a máquina do utilizador está a carregar os dados do jogo.

Relativamente aos pacotes encriptados, desenvolvedores de *cheats* já descobriram formas de contornar o mecanismo de defesa. Um desenvolvedor com a alcunha *DeepBlueSky* implementou uma solução a que chamou *NetSharkGO* que envolvia aceder à memória do jogo, adquirir a chave de cifra usada para cifrar os pacotes e decifrá-los após a recolha. Este software foi divulgado ao público em janeiro de 2015, num *post* do seu *blog* pessoal, juntamente com um pequeno tutorial em como encontrar a chamada "*ICE key*" para decifrar as comunicações do jogo [81]. Em dezembro do mesmo ano, outro desenvolvedor, possivelmente inspirado pelo trabalho de *DeepBlueSky*, publicou num fórum uma prova de conceito para um *packet sniffer* para CSGO que permitia a batoteiros a utilização de ESPs, mais concretamente no formato de um radar com localizações de jogadores adversários [70].

Em julho de 2018, no mesmo fórum, foi criado um *post* [82] com informações sobre como as comunicações do jogo funcionam, bem como indicações relativamente à *ICE key* baseadas ainda no método divulgado por *DeepBlueSky*, 3 anos antes. Portanto, não está fora das possibilidades que este método funcione até hoje. Ainda assim, existem algumas medidas impostas pela *Valve* para os seus servidores oficiais que os protegem deste método, na medida em que as chaves são dinâmicas para estes casos. *Anti-Cheats* de terceiros também possuem medidas preventivas deste

género.

O principal problema à volta desta técnica é que servidores comunitários, onde os métodos apresentados funcionam, estão à mercê de quem os queira atacar. O VAC não deverá causar quaisquer problemas aos batoteiros que já possuam a *ICE key*, pois o *cheat* irá funcionar depois apenas ao nível das comunicações, sendo incrivelmente difícil de detetar. Outra questão a ter em conta é a de organizadores de torneios pouco experientes, ou simplesmente sem recursos para fazer face a este tipo de ataque.

No que toca a servidores protegidos com *Anti-Cheats*, caso os desenvolvedores destes sejam organizadores de torneios, devem possuir o conhecimento suficiente para, teoricamente, obter a chave e alterar pacotes ou simplesmente eliminá-los das comunicações quando os seus conteúdos não são desejáveis. Com o aparecimento de sites de apostas para torneios de CSGO, esta situação pode ser uma a ter em atenção.

Por fim, no que diz respeito aos comprimentos dos pacotes, os dados obtidos demonstram que, em condições semelhantes aos servidores de jogo configurados para o torneio de CSGO do SimSIC, há comprimentos que simplesmente não são observados nas comunicações. Uma possível medida de prevenção contra ataques pela rede poderia passar por bloquear pacotes enviados pelo cliente com comprimentos entre 0 e 39 bytes e maiores ou iguais a 1280 bytes. Apesar de tudo, devem ser colocadas reservas a esta medida, pois é necessário averiguar se os comprimentos dos pacotes dos clientes podem variar consoante hardware usado, número de jogadores no servidor, entre outros. Segundo Ratti et al. [18], no *Counter-Strike*, uma das versões anteriores ao CSGO, os pacotes dos clientes são independentes desse tipo de fatores. Um estudo deste género teria de ser efectuado para o CSGO. Uma medida preventiva complementar passaria por recolher valores de *Burst Count* numa série de situações, para depois implementar um aviso ao administrador quando o *Burst Count* apresentasse um valor demasiado elevado, logo, suspeito, em relação à média observada.

Capítulo 6

Conclusões

Foi realizado um enquadramento que serviu como base de conhecimento para a análise de segurança das comunicações recolhidas, em contexto de torneio, do jogo CSGO. Neste enquadramento encontra-se uma taxonomia de métodos de batota, criada a partir do relacionamento de taxonomias de várias fontes, com o objetivo de tornar mais fácil o entendimento da matéria.

Na fase de recolha e análise de comunicações, apresentaram-se a metodologia pela qual se utilizou a ferramenta *Wireshark* na recolha e análise de pacotes do jogo CSGO, juntamente com os dados obtidos com o processo.

Discutiram-se os dados numa perspetiva de segurança, procurando falhas e possíveis mecanismos de defesa. Foi detetada uma falha de segurança em que informações, incluindo a senha do servidor, são enviadas em claro, nas comunicações, pela máquina do jogador para o servidor e abordaram-se possíveis consequências. Argumentou-se sobre a provável existência de uma falha de segurança que alia a análise de memória e modificações das comunicações, e o que isso implica para os jogadores e organizadores de torneios. Por fim, foram propostos mecanismos preventivos de segurança com base no comprimento dos pacotes, ainda que com ressalvas.

Trabalhos futuros neste tema devem passar sobre o comportamento das comunicações em rede geradas pelo CSGO, nomeadamente se os comprimentos dos pacotes podem ou não variar sobre diferentes circunstâncias, como número de jogadores em simultâneo no servidor, versão do jogo ou equipamento usado, para apresentar alguns exemplos.

Seria importante também perceber se as semelhanças de dados e/ou padrões entre pacotes de sessões diferentes apresenta alguma relevância no que toca a uma fraqueza no processo de encriptação.

A realização deste projeto permitiu consciencializar que a proteção de jogos

online, nos moldes atuais, é uma tarefa extremamente complicada. Existe um vasto número de vetores de ataque que podem ser explorados por desenvolvedores de *cheats* e/ou batoteiros. Os mecanismos de segurança existentes variam do pouco ou nada eficaz para os muito eficazes, mas não há forma de tornar um jogo *online* 100% seguro. As ferramentas que os desenvolvedores de jogos possuem também estão disponíveis para os desenvolvedores de *cheats*. Para estes últimos, a única limitação no desenvolvimento é o tempo de vida do jogo alvo, e nesse aspeto estão em vantagem em relação aos desenvolvedores de jogos.

Com recursos praticamente ilimitados, a melhor abordagem parece mesmo ser a tomada pela *Valve*, com o seu sistema *VACnet*. Tirando partido de *deep learning*, a grande maioria dos vetores de ataque são neutralizados. O desafio que vem com esta tecnologia é que os batoteiros têm de ser castigados o mais rápido possível. No entanto, se o sistema funcionar com uma amostra limitada de dados de um jogador, corre-se o risco de detetar erroneamente um batoteiro e castigar um jogador inocente. Por outro lado, se o sistema demorar demasiado tempo, as batotas podem-se espalhar livremente pela comunidade até ao momento da deteção, arruinando a experiência de muitos jogadores legítimos. Nem um sistema sofisticado como o *VACnet* tem utilidade, neste momento, sem medidas preventivas fortes a complementá-lo.

Mas a maioria dos desenvolvedores de jogos não possuem recursos ilimitados. Para estes, a realidade é a de que têm de usar os meios mais apropriados tendo em conta os seus custos de implementação. Com a presença de torneios com recompensas monetárias, é bastante viável que ocorram situações onde batoteiros vencem estes torneios e ficam com os prémios.

Bibliografia

[1] S. D. Webb e S. Soh, "Cheating in networked computer games," in *Proceedings of the 2nd international conference on Digital interactive media in entertainment and arts - DIMEA '07*, vol. 45, n. 3. New York, New York, USA: ACM Press, jun 2007, p. 105. [*Online*]. Disponível: https://espace.curtin.edu.au/bitstream/handle/20.500.11937/10134/20769{_}downloaded{_}stream{_}225.pdf?sequence=2{&}isAllowed=yhttp://dl.acm.org/citation.cfm?doid=2480741.2480742http://portal.acm.org/citation.cfm?doid=1306813.1306839 (citado nas págs. ix, 3, 4, 11, 14 e 21)

[2] A. Gonçalves, "Primeiro Easter Egg no gaming - Adventure, de Warren Robinett - legendary," 2015. [*Online*]. Disponível: https://pt.ign.com/legendary/14725/feature/primeiro-easter-egg-no-gaming-adventure-de-warren-robinett (citado nas págs. ix e 7)

[3] P. Laurens, R. F. Paige, P. J. Brooke, e H. Chivers, "A Novel Approach to the Detection of Cheating in Multiplayer Online Games," in *12th IEEE International Conference on Engineering Complex Computer Systems (ICECCS 2007)*. IEEE, 2007, pp. 97–106. [*Online*]. Disponível: http://ieeexplore.ieee.org/document/4276306/https://sci-hub.tw/10.1109/ICECCS.2007.11 (citado nas págs. ix, 12, 14, 15, 17 e 21)

[4] Lorise, "[Release] External ESP Hack," 2016. [*Online*]. Disponível: https://www.unknowncheats.me/forum/battlefield-1-a/193901-external-esp-hack.html (citado nas págs. ix e 18)

[5] M. Pritchard, "How to hurt the hackers: The scoop on internet cheating and how you can combat it," *Gamasutra, July*, vol. 24, 2000. (citado nas págs. ix, 10, 14, 16, 17, 19, 20, 23, 24 e 26)

[6] WePC, "2019 Video Game Industry Statistics, Trends & Data," 2019. [*Online*]. Disponível: https://www.wepc.com/news/video-game-statistics/{#}esport (citado na pág. 1)

[7] C. Gough, "eSports market - Statistics & Facts — Statista," 2019. [*Online*]. Disponível: https://www.statista.com/topics/3121/esports-market/ (citado na pág. 1)

[8] J. Hamari e M. Sjöblom, "What is eSports and why do people watch it?" *Internet Research*, vol. 27, n. 2, pp. 211–232, apr 2017. [*Online*]. Disponível: http://www.emeraldinsight.com/doi/10.1108/IntR-04-2016-0085 (citado na pág. 1)

[9] L. Koch, "Esports Playing in the Big Leagues Now," 2019. [*Online*]. Disponível: https://www.emarketer.com/content/esports-disrupts-digital-sports-streaming (citado na pág. 1)

[10] B. Hook, "Introduction To Multiplayer Game Programming - bookofhook - Trac," 2007. [*Online*]. Disponível: http://trac.bookofhook.com/bookofhook/trac.cgi/wiki/IntroductionToMultiplayerGameProgramming (citado nas págs. 3, 4, 5 e 6)

[11] J. Smed, T. Kaukoranta, e H. Hakonen, "Aspects of networking in multiplayer computer games," *The Electronic Library*, vol. 20, n. 2, pp. 87–97, apr 2002. [*Online*]. Disponível: http://www.uu.net/network/latencyhttps://www.emeraldinsight.com/doi/10.1108/02640470210424392 (citado nas págs. 4, 10, 15, 20 e 24)

[12] A. Yahyavi e B. Kemme, "Peer-to-Peer Architectures for Massively Multiplayer Online Games: A Survey," *ACM Computing Surveys*, vol. 46, n. 1, 2013. [*Online*]. Disponível: http://dx.doi.org/10.1145/2522968.2522977 (citado nas págs. 4 e 31)

[13] K. Jenkins, "Gamasutra - Designing Secure, Flexible, and High Performance Game Network Architectures," 2004. [*Online*]. Disponível: https://www.gamasutra.com/view/feature/130587/designing{_}secure{_}flexible{_}and{_}.php (citado nas págs. 4, 10 e 13)

[14] Epic Games, "Unreal Networking Architecture - Overview." [*Online*]. Disponível: https://api.unrealengine.com/udk/Three/NetworkingOverview.html{#}Overview (citado nas págs. 4 e 26)

[15] M. Kurylovych, "Lockstep protocol," 2018. (citado na pág. 4)

[16] D. Blaisdell, "Game Networking Techniques, Explained with Pong," 2014. [*Online*]. Disponível: http://drewblaisdell.com/writing/game-networking-techniques-explained-with-pong/ (citado na pág. 4)

[17] T. T. Chen, "Online Games: Research Perspective and Framework," *ACM Comput. Entertain*, vol. 11, n. 3, 2014. [*Online*]. Disponível: http://dx.doi.org/10.1145/2582193.2633445 (citado nas págs. 5 e 25)

[18] S. Ratti, B. Hariri, e S. Shirmohammadi, "A Survey of First-Person Shooter Gaming Traffic on the Internet," *IEEE Internet Computing*, vol. 14, n. 5, pp. 60–69, sep 2010. [*Online*]. Disponível: https://www.researchgate.net/publication/220491343http://ieeexplore.ieee.org/document/5445069/ (citado nas págs. 5 e 48)

[19] Doctor Web, "Study of the Belonard Trojan, exploiting zero-day vulnerabilities in Counter-Strike 1.6," Russia, Relatório Técnico, 2019. [*Online*]. Disponível: www.drweb.com (citado nas págs. 5 e 34)

[20] G. Fielder, "UDP vs. TCP — Gaffer On Games," 2008. [*Online*]. Disponível: https://gafferongames.com/post/udp{_}vs{_}tcp/ (citado na pág. 5)

[21] J. Kurose e K. Ross, *Computer Networking: A Top-Down Approach*, 7th ed. Pearson, 2016. [*Online*]. Disponível: https://www.pearson.com/us/higher-education/program/Kurose-Computer-Networking-A-Top-Down-Approach-7th-Edition/PGM1101673.html (citado na pág. 5)

[22] B. A. Forouzan, "TCP/IP Protocol Suite," *McGraw-Hill*, 2008. (citado na pág. 5)

[23] C. Xianhui e B. Ip, "Packet-level traffic analysis of online games from the genre characteristics perspective," *Journal of Network and Computer Applications*, vol. 35, pp. 240–252, 2011. [*Online*]. Disponível: www.elsevier.com/locate/jnca (citado na pág. 6)

[24] M. Consalvo, *Cheating: Gaining Advantage in Videogames*. MIT Press, 2009. [*Online*]. Disponível: http://mitpress.mit.eduhttps://is.muni.cz/el/1421/podzim2014/IM082/um/Consalvo{_}{_}Cheating{_}{_}Gaining{_}Advantage{_}in{_}Videogames.pdf (citado nas págs. 6, 7, 8, 9, 11, 12, 16, 17 e 25)

[25] D. Spohn, "A History of Cheating in Online Games," 2019. [*Online*]. Disponível: https://www.lifewire.com/cheating-in-online-games-1983529 (citado nas págs. 7, 8 e 16)

[26] I. Mussa e J. Messias, "Code breakers e bunny hoppers: transgressão inventiva nas camadas intra e suprajogo," *Sessões do Imaginário*, vol. 22, n. 38, pp. 14–27, oct 2017. [*Online*]. Disponível: http://revistaseletronicas.pucrs.br/ojs/index.php/famecos/article/view/29811 (citado na pág. 7)

[27] Wrestling With Gaming, "The Story Of The GameShark - Gaming's Most Famous Cheating Device! - YouTube," apr 2018. [*Online*]. Disponível: https://www.youtube.com/watch?v=h4vs4X66nas (citado na pág. 7)

[28] G. Hoglund e G. McGraw, *Exploiting Online Games: Cheating Massively Distributed Systems*. Boston, MA: Addison-Wesley Professional, 2007. (citado nas págs. 8, 9, 11, 14, 17, 19, 20 e 26)

[29] Y.-C. Chen, P. S. Chen, J.-J. Hwang, L. Korba, R. Song, e G. Yee, "An analysis of online gaming crime characteristics," vol. 15, n. 3, pp. 1066–2243, 2005. [*Online*]. Disponível: www.emeraldinsight.com/researchregisterwww.emeraldinsight.com/1066-2243.htmCrown (citado nas págs. 8 e 9)

[30] "[CTHOOK.US] Hack vs Hack Servers - MPGH - MultiPlayer Game Hacking & Cheats," 2017. [*Online*]. Disponível: https://www.mpgh.net/forum/showthread.php?t=1322196 (citado nas págs. 9 e 26)

[31] "Hack vs Hack - Reddit," 2019. [*Online*]. Disponível: https://www.reddit.com/r/hvh/ (citado na pág. 9)

[32] "IGXE - Your trusted Fortnite Items, FFXI Gil, Swtor Credits, POE Items, Wow Gold, Fifa Coins Internet Game Exchange." 2019. [*Online*]. Disponível: https://www.igxe.com/ (citado na pág. 9)

[33] L. Franceschi-Bicchierai, "For 20 Years, This Man Has Survived Entirely by Hacking Online Games - VICE," jul 2017. [*Online*]. Disponível: https://www.vice.com/en{_}us/article/59p7qd/this-man-has-survived-by-hacking-mmo-online-games (citado na pág. 9)

[34] M. Moravčík, M. Schmid, N. Burch, V. Lisý, D. Morrill, N. Bard, T. Davis, K. Waugh, M. Johanson, e M. Bowling, "Deepstack: Expert-level artificial intelligence in heads-up no-limit poker," *Science*, vol. 356, n. 6337, pp.

508–513, 2017. [*Online*]. Disponível: https://science.sciencemag.org/content/356/6337/508 (citado na pág. 9)

[35] J. Yan e B. Randell, "A systematic classification of cheating in online games," in *Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games - NetGames '05*. New York, New York, USA: ACM Press, 2005, p. 1. [*Online*]. Disponível: http://portal.acm.org/citation.cfm?doid=1103599.1103606https://sci-hub.tw/10.1145/1103599.1103606 (citado nas págs. 11, 14, 15, 16, 24 e 27)

[36] X. Lan, Y. Zhang, e P. Xu, *An Overview on Game Cheating and Its Countermeasures*, 2009. [*Online*]. Disponível: https://pdfs.semanticscholar.org/c9ad/909c40fbff773f60e9caf4e99ae8f24117ab.pdf (citado nas págs. 12, 14 e 16)

[37] K. K. Mikkelsen, "Information security as a countermeasure against cheating in video games," Tese de doutoramento, Norwegian University of Science and Technology, 2017. [*Online*]. Disponível: https://brage.bibsys.no/xmlui/bitstream/handle/11250/2448953/18113{_}FULLTEXT.pdf?sequence=1 (citado nas págs. 12 e 25)

[38] J. Davis, "8 of the Most Notorious Video Game Exploits of All Time - IGN," 2014. [*Online*]. Disponível: https://www.ign.com/articles/2014/09/29/8-of-the-most-notorious-video-game-exploits-of-all-time (citado na pág. 12)

[39] L. Mira, "LDLC file protest over fnatic boost — HLTV.org," 2014. [*Online*]. Disponível: https://www.hltv.org/news/13722/ldlc-file-protest-over-fnatic-boost (citado na pág. 12)

[40] M. Švejda, "DH: "Overpass will be replayed HLTV.org," 2014. [*Online*]. Disponível: https://www.hltv.org/news/13726/dh-overpass-will-be-replayed (citado na pág. 13)

[41] P. Milovanovic, "Fnatic forfeit LDLC match — HLTV.org," 2014. [*Online*]. Disponível: https://www.hltv.org/news/13731/fnatic-forfeit-ldlc-match (citado na pág. 13)

[42] J. Woo e H. K. Kim, "Survey and research direction on online game security," in *Proceedings of the Workshop at SIGGRAPH Asia on - WASA '12*. New York, New York, USA: ACM Press, 2012, p. 19. [*Online*]. Disponível: http://dl.acm.org/citation.cfm?doid=2425296.2425300https://dabamirror.sci-hub.

se/4360/71acb7f1b04d9c079d1088bdc81c4b81/woo2012.pdf{#}view=FitH (citado nas págs. 13 e 20)

[43] V. Corporation, "About Us - Valve Corporation," 2019. [*Online*]. Disponível: https://www.valvesoftware.com/pt/about (citado na pág. 13)

[44] A. Jones, "DDoS attacks "are going to happen," according to Valve — PCGamesN," 2019. [*Online*]. Disponível: https://www.pcgamesn.com/valve-ddos-attacks (citado na pág. 13)

[45] R. Scott-Jones, "Here's how to protect yourself from DDoS attacks in CS:GO — PCGamesN," 2016. [*Online*]. Disponível: https://www.pcgamesn.com/counter-strike-global-offensive/steam-datagram-relay-sdr-ddos-attacks (citado na pág. 13)

[46] T. Kovanen, "Understanding DDoS attacks — HLTV.org," 2014. [*Online*]. Disponível: https://www.hltv.org/news/13213/understanding-ddos-attacks (citado na pág. 13)

[47] ESEA, "ESEA Hardware Cheats - Update," 2018. [*Online*]. Disponível: https://blog.esea.net/esea-hardware-cheats/ (citado nas págs. 14 e 15)

[48] Wireshark, "Wireshark · Go Deep." 2020. [*Online*]. Disponível: https://www.wireshark.org/ (citado na pág. 15)

[49] C. Godfrey, "'It's incredibly widespread': why eSports has a match-fixing problem — Games — The Guardian," 2018. [*Online*]. Disponível: https://www.theguardian.com/games/2018/jul/31/its-incredibly-widespread-why-esports-has-a-match-fixing-problem (citado na pág. 17)

[50] G. Armitage, M. Claypool, e P. Branch, *Networking and Online Games: Understanding and Engineering Multiplayer Internet Games*. Chichester, UK: John Wiley & Sons, Ltd, apr 2006. [*Online*]. Disponível: http://doi.wiley.com/10.1002/047003047Xhttp://index-of.co.uk/Tutorials/NetworkingAndOnlineGames.pdf (citado nas págs. 17, 19, 24 e 27)

[51] S. Khalifa, "Machine Learning and Anti-Cheating in FPS Games," Tese de doutoramento, University College London, 2016. [*Online*]. Disponível: https://www.researchgate.net/publication/308785899{_}Machine{_}Learning{_}and{_}Anti-Cheating{_}in{_}FPS{_}Games (citado na pág. 18)

[52] Y. S. Fung e J. C. S. Lui, "Hack-proof synchronization protocol for multi-player online games," *Multimedia Tools and Applications*, vol. 41, n. 2, pp. 305–331, jan 2009. [*Online*]. Disponível: http://link.springer.com/10.1007/s11042-008-0230-3 (citado na pág. 18)

[53] R. J. Robles, S.-S. Yeo, Y.-D. Moon, G. Park, e S. Kim, "Online Games and Security Issues," in *2008 Second International Conference on Future Generation Communication and Networking*. IEEE, dec 2008, pp. 145–148. [*Online*]. Disponível: http://ieeexplore.ieee.org/document/4734193/ (citado nas págs. 18 e 20)

[54] MITRE, "Technique: Software Packing - Enterprise — MITRE ATT&CK™," 2019. [*Online*]. Disponível: https://attack.mitre.org/techniques/T1045/ (citado na pág. 19)

[55] V. Corporation, "Pure Servers - Valve Developer Community," 2015. [*Online*]. Disponível: https://developer.valvesoftware.com/wiki/Pure{_}Servers (citado na pág. 20)

[56] Valve Corporation, "ICE - Valve Developer Community," 2016. [*Online*]. Disponível: https://developer.valvesoftware.com/wiki/ICE (citado na pág. 20)

[57] J. Yan, "Security design in online games," in *19th Annual Computer Security Applications Conference, 2003. Proceedings.* IEEE, 2003, pp. 286–295. [*Online*]. Disponível: https://www.acsac.org/2003/papers/114.pdfhttp://ieeexplore.ieee.org/document/1254333/ (citado na pág. 20)

[58] Valve Corporation, "Counter-Strike: Global Offensive » CS:GO Overwatch FAQ," 2019. [*Online*]. Disponível: https://blog.counter-strike.net/index.php/overwatch/ (citado na pág. 20)

[59] J. McDonald, "GDC Vault - Robocalypse Now: Using Deep Learning to Combat Cheating in 'Counter-Strike: Global Offensive'," 2018. (citado na pág. 21)

[60] E. Lahti, "Valve has 1,700 CPUs working non-stop to bust CS:GO cheaters — PC Gamer," 2018. [*Online*]. Disponível: https://www.pcgamer.com/vacnet-csgo/ (citado na pág. 21)

[61] Valve Corporation, "Valve Anti-Cheat - Valve Developer Community," 2017. [*Online*]. Disponível: https://developer.valvesoftware.com/wiki/Valve{_}Anti-Cheat (citado na pág. 21)

[62] Valve Corporation, "Valve Anti-Cheat System (VAC) - Valve Anti-Cheat (VAC) System - Knowledge Base - Steam Support," 2017. [*Online*]. Disponível: https://support.steampowered.com/kb/7849-RADZ-6869/ (citado na pág. 21)

[63] Blizzard Entertainment, "World of Warcraft: Acordo Anti-cheat," 2019. [*Online*]. Disponível: https://us.battle.net/account/legal/anticheating.html (citado na pág. 21)

[64] Blizzard Entertainment, "Blizzard Entertainment," 2019. [*Online*]. Disponível: https://www.blizzard.com/ (citado na pág. 21)

[65] FACEIT, "Anti-Cheat - FACEIT.com," 2019. [*Online*]. Disponível: https://www.faceit.com/en/anti-cheat (citado na pág. 21)

[66] FACEIT, "Anti-Cheat FAQ – FACEIT," 2019. [*Online*]. Disponível: https://support.faceit.com/hc/en-us/articles/115000061290-Anti-Cheat-FAQ (citado na pág. 21)

[67] Turtle Entertainment Online, "ESEA - ESEA Client: How it works," 2015. [*Online*]. Disponível: https://play.esea.net/index.php?s=content{&}d=anticheat (citado na pág. 21)

[68] Epic Games, "Easy Anti-Cheat," 2019. [*Online*]. Disponível: https://www.easy.ac/en-us/ (citado na pág. 21)

[69] J. Jeff Yan e H.-J. Choi, "Security issues in online games," *The Electronic Library*, vol. 20, n. 2, pp. 125–133, 2002. (citado na pág. 23)

[70] Dude719, "[Release] CSGO Packet Sniffer Proof of Concept," 2015. [*Online*]. Disponível: https://www.unknowncheats.me/forum/cs-go-releases/165673-csgo-packet-sniffer-proof-concept.html (citado nas págs. 26 e 47)

[71] Valve Corporation, "Counter-Strike: Global Offensive Dedicated Servers - Valve Developer Community," 2019. [*Online*]. Disponível: https://developer.valvesoftware.com/wiki/Counter-Strike:{_}Global{_}Offensive{_}Dedicated{_}Servers{#}Linux{_}Scripts (citado nas págs. 26 e 38)

[72] Lenosisnickerboa, "GitHub - lenosisnickerboa/csgosl: A CSGO server launcher, one-click install & run," 2019. [*Online*]. Disponível: https://github.com/lenosisnickerboa/csgosl (citado nas págs. 38 e 39)

[73] Valve Corporation, "Required Ports for Steam - Network/Connection Issues - Base de Conhecimento - Steam Support," 2017. [*Online*]. Disponível: https://support.steampowered.com/kb{_}article.php?ref=8571-GLVN-8711{&}l=portuguese (citado na pág. 38)

[74] UbiNET, "SimSIC 2019 - Simpósio de Segurança Informática e Cibercrime - UbiNET - IPBeja," 2019. [*Online*]. Disponível: https://ubinet.ipbeja.pt/?simsic=2019 (citado na pág. 38)

[75] UbiNET, "UbiNET - Segurança Informática e Cibercrime," 2019. [*Online*]. Disponível: https://ubinet.ipbeja.pt/index.php (citado na pág. 38)

[76] Liquipedia, "2019-05-07 Patch - Liquipedia Counter-Strike Wiki," 2019. [*Online*]. Disponível: https://liquipedia.net/counterstrike/2019-05-07{_}Patch (citado na pág. 40)

[77] Wireshark, "Performance - The Wireshark Wiki," 2013. [*Online*]. Disponível: https://wiki.wireshark.org/Performance (citado na pág. 40)

[78] Jasper, "PCAP Split and Merge — Packet-Foo — Network Packet Capture and Analysis," 2018. [*Online*]. Disponível: https://blog.packet-foo.com/2018/07/pcap-split-and-merge/ (citado na pág. 40)

[79] Valve Corporation, "Counter-Strike: Global Offensive » Release Notes for 9/21/2016," 2016. [*Online*]. Disponível: https://blog.counter-strike.net/index.php/2016/09/16032/ (citado na pág. 44)

[80] Panda Security, "52% of users reuse their passwords for different services." 2018. [*Online*]. Disponível: https://www.pandasecurity.com/mediacenter/security/password-reuse/ (citado na pág. 47)

[81] DeepBlueSky, "NetShark: Attack Vectors — #debuglog," 2015. [*Online*]. Disponível: https://debuglog.wordpress.com/2015/01/13/netshark-attack-vectors/ (citado na pág. 47)

[82] Phog, "[Information] Networking Megathread," 2018. [*Online*]. Disponível: https://www.unknowncheats.me/forum/counterstrike-global-offensive/292668-networking-megathread.html (citado na pág. 47)