ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Alma Mater Studiorum - Università di Bologna

DIPARTIMENTO DI INFORMATICA: SCIENZA E INGEGNERIA
Corso di laurea in Informatica

Tesi di laurea

La diffusione dei cheating bot nei videogiochi e possibili soluzioni di contrasto

Relatore:
Prof. Michele Colajanni

Candidato:
Lorenzo Castelli
Matricola 0000880149

Anno Accademico 2021-2022

A Nonno Antonio "Tonino"

Sommario

Il fine di questo elaborato riguarda lo studio di soluzioni per il contrasto di giocatori baranti controllati da algoritmi presenti nel videogioco online *Team Fortress 2*. Dopo una breve introduzione alla storia degli sparatutto online, si descriverà il funzionamento di tutti i componenti che sviluppano l'ambiente di gioco, oltre a definire termini e sistemi vitali per la comprensione dell'elaborato ed una breve introduzione a Team Fortress 2. Si procederà alla discussione del cheat e dei software e/o *environment* sfruttati dagli attacanti in partita, andando a cercare di spiegare il meccanismo e l'origine di questi elementi, nonché introdurre il concetto dei bot baranti implementati usando il programma open source cathook. Una volta spiegata la minaccia si andrà a spiegare la difesa da parte del gioco e degli sviluppatori attraverso il software di anticheat *Valve Anti-Cheat* (VAC) presente sul gioco, definendo le terminologie e alcune caratteristiche comuni rispetto agli altri, per poi introdurre le nuove tecnologie di contrasto sviluppati per Counter Strike: Global Offensive, ovvero Overwatch, Trust Factor e l'anticheat con *deep learning* VACNET. Infine, dopo aver definito più approfonditamente il funzionamento degli algoritmi baranti, verranno suggerite delle possibili soluzioni implementabili e del motivo per cui non riescono a risolvere completamente il problema. Concluderemo spiegando cosa stanno facendo i sviluppatori, per poi descrivere come effettivamente il problema possiede come l'unica soluzione di evitare di giocare nei server ufficiali di gioco, mantenendo comunque gli algoritmi liberi nei server ufficiali.

Indice

Ringraziamenti

Ringraziamento al mio relatore Michele Colajanni per avermi assistito alla stesura della tesi e per i generosi consigli.

Ringrazio di cuore ai miei amici che lungo i periodi accademici più bui sono sempre stati parte della luce che mi ha spinto a proseguire il mio cammino professionale.

Ringraziamento speciale per la mia famiglia che si è sempre impegnata a supportarmi e incoraggiarmi a continuare e completare il mio cammino universitario.

E infine caloroso ringraziamento a Nonno Antonio, incoraggiatore principale e fonte di ispirazione per la quale ho scelto di intraprendere lo studio dell'informatica, seguendone il mio desiderio di diventare un game developer.

Introduzione

Nel vasto panorama dell'intrattenimento moderno, il videogioco ha ottenuto una posizione di interesse generale esponenziale. Anni di evoluzione del genere hanno portato implementazioni di differenti tecnologie per mantenere l'interesse dell'utente. Il videogioco possiede differenti tipi di genere; uno di questi è uno dei più comuni e redditizi: il *First Person Shooter* (FPS, in italianoSparatutto in Prima Persona). Col passare degli anni, gli FPS si sono evoluti al punto che si possono giocare sia in solitario che in compagnia con altri utenti, online e non. L'introduzione del sistema di gioco condiviso online ha portato l'interesse di vari tipi di utenze, compreso coloro che non hanno alcun interesse di giocare in modo onesto. Un gioco possiede delle regole e tra coloro che vi vogliono partecipare vi saranno coloro che le seguiranno e coloro che le romperanno: nei videogiochi online il concetto rimane lo stesso, cambiano solo il modo. Lungo una serie di prove ed errori, gli sviluppatori di videogiochi hanno provato vari approcci per contrastare coloro che usano software di terze parti per poter svincolare le regole per ottenere la vittoria; questi utenti vengono identificati come *Cheater*(tradotto grossolanamente in baroni) ed essi usano certi programmi noti che permettono di sbloccare ed usare vantaggi inaccessibili con mezzi normali, sfruttando alcune falle presenti nel gioco e modificando parametri via kernel. Questo problema, ha portato gli sviluppatori a creare e implementare software che contrastano questi tipi di utenti, noti come sistemi di anticheat la quale hanno il compito di individuarli e di rimuovere completamente ogni possibilità di poter interagire di nuovo con il gioco (formalmente, dare un ban all'utente). Tuttavia, malgrado gli sforzi di sviluppare un software per contrastare un utente umano essi non bastano per contrastare coloro che non lo sono. Questo studio si concentrerà su un caso d' infestazione di bot e delle possibili forme di contrasto applicabili. I bot sono algoritmi gestiti da utenti umani che entrano nelle partite e disturbano i giocatori umani usando cheat e generando altre forme di fastidio. In questa tesi si vuole introdurre il problema dell'invasione dei bot in Team Fortress 2 mediante 4 fasi: il primo capitolo tratterà dei componenti a cui si faranno riferimento per spiegare il problema, come il funzionamento di un server per un FPS e la gestione

degli input durante la partita, annettendo una piccola introduzione alla storia del gioco online. Successivamente verrà analizzato il problema, ovvero come funziona un cheat per un videogioco online per poi portarsi verso la spiegazione della gestione e origine dei bot baranti. Si passerà poi a parlare della soluzione presente, dell'anticheat, del VAC e di come questo sistema non basti per contrastare non solo i giocatori baranti attuali ma anche i bot. Infine, si cercherà di discutere di possibili soluzioni pensate da alcuni utenti della comunità, indicandone come potrebbero risolvere il problema e come, ultimamente, possano venir superati dagli attaccanti. La tesi concluderà discutendo delle soluzione implementate dagli sviluppatori e dei possibili sviluppi futuri.

Capitolo 1

Multiplayer nei FPS

1.1 First Person Shooter

I First Person Shooter, o sparatutto in italiano, è un genere di videogioco d'azione distinto per l'uso di un'arma da distanza e la visuale di gioco in prima persona. Il primissimi videogiochi che condividevano questi parametri erano molto rudimentali, come Maze e Spasim sviluppati rispettivamente nel 1973 e 1974. Il primo FPS ad ottenere una certa popolarità fu' Battlezone: sviluppato nel 1980 per i cabinati arcade, il gioco consisteva nel controllare un carro armato e di distruggere quelli nemici. Col passare degli anni gli FPS continuavano a salire velocemente sulla scala evolutiva: dalla grafica 2D a quella 3D, dall'avere la visuale di movimento ad X ad avere la visuale libera, dall'avere poca storia a creare interi mondi attraverso narrative uniche. In generale i primi FPS erano per giocatore singolo. Pochi avevano una modalità multigiocatore, e quei pochi che lo avevano erano limitati ad essere giocati in locale attraverso lo *split screen* (in italiano, schermo condiviso), sistema che divideva lo schermo per ogni giocatore, permettendo di poter far giocare a più persone contemporaneamente con lo stesso display; in questo modo ogni giocatore doveva guardare il proprio schermo per poter controllare le sue azioni. L' evoluzioni definite, insieme anche a questa, stavano in linea con quelli degli altri generi di videogiochi, per esempio lo *split screen* veniva usato anche nei giochi di corsa oppure nei *platform*; ma vi è uno sviluppo altamente rivoluzionario che ha reso il genere FPS unico, almeno ai tempi: il multiplayer online.

1.1.1 Storia del Multiplayer e degli FPS

I primi titoli che implementarono una componente di rete per il multiplayer nascono durante gli anni '80: richiedevano 2 computer collegati fra di loro via cavo e dovevano essere ben sincronizzati per la ricezione e il passaggio di informazioni tra di loro. Secondo Ken Wasserman e Tim Stryker riguardo l'analisi del gioco *Flash Attack*(1980), constatarono che questo sistema di gioco generava una sensazione diversa rispetto a quello contro una macchina come si era fatto fino a quel momento. Attraverso il gioco contro un giocatore umano senza condivisione del display/calcolatore implementava 3 nuove variabili:

1. Invece di competere contro la macchina, essa veniva usata come mezzo per poter competere contro un giocatore vero con le proprie strategie uniche rispetto a quelle programmate dalla macchina.

2. Anche se vengono definite delle regole generali per la quale entrambi i giocatori sono al corrente, la vittoria è data a colui che ha sopraffatto il suo avversario pur avendo poca se non nessuna conoscenza delle sue tattiche.

3. Il gioco veniva giocato in tempo reale. La temporalità delle mosse si basavano solo dai riflessi e dall'intelligenza del giocatore.

Col tempo si svilupparono sempre nuove tecnologie, fino alla creazione del primo gioco che permetteva il multiplayer locale via LAN, Spectre(1991) sviluppato per l'Apple Macintosh, con il supporto per ben 8 giocatori per partita. Naturalmente il sistema non poteva ancora definirsi online, siccome le partite venivano svolte all' interno di una rete locale. Finalmente nel 1993 venne rilasciato Doom, sviluppato dalla Id Software; il gioco, oltre ad essere riconosciuto come il gioco che ha definito l'era degli FPS negli anni '90, fu' anche uno dei primi a possedere una modalità multiplayer online: Doom disponeva infatti della modalità *deathmatch* fino a 4 giocatori, essa veniva giocata via internet attraverso il sistema *dial-up*, un tipo di connessione che sfruttava l'uso di un modem digitando un numero telefonico negli *dialer*. Tuttavia questo modello è molto inefficiente per gestire rapidamente le informazioni dei giocatori, limitandone inoltre il numero di giocatori per partita: Doom era talmente popolare che l'online provveduto dalle compagnie telefoniche non riuscivano a stare dietro alla quantità di persone presenti, portando quindi alla riduzione del *bandwith*. Col passare del tempo, avvennero altri nuovi sviluppi tecnologici: dal sistema *dial-up* (Doom e Quake) all'uso dei server, dai giochi principalmente singleplayer con una componente multiplayer a quelli solo multiplayer(Starsiege:Tribes, Unreal Tournament), dall'online esclusivamente su PC all'estensione del sistema per le console(Halo, Call of Duty).

1.2 Come funziona il server

Nel caso di Team Fortress 2, gioco sotto esame in questo elaborato, il multiplayer online usa il sistema di comunicazione client-server con server in Linux. Per poter meglio definire il funzionamento è necessario descrivere alcuni strumenti programmati nel gioco per facilitare il lavoro della comunicazione mantenendo semplice e fluido il gioco per i client in partita.

1.2.1 *Hitbox*

Per poter definire bene le interazioni e limiti tra il mondo di gioco e i giocatori, ogni modello 3D dei personaggi controllati dal giocatore e dei blocchi ambientali presenti nelle mappe di gioco possiedono delle geometrie aggiuntive invisibili ai client che servono al server per poter elaborare gli output a seconda delle collisioni: sia proiettili che elementi fisici del gioco. Questi blocchi sono disposti per coprire approssimativamente il modello del personaggio: questo affinché il server capisca quando avviene un contatto e quale tipo di output rilasciare. Per esempio nella situazione per la quale un giocatore arriva davanti ad un muro, il server vede che la *hitbox* del giocatore sta toccando quella del muro, portandolo a mandare come output il blocco del personaggio, in modo tale che il giocatore non possa attraversarlo.

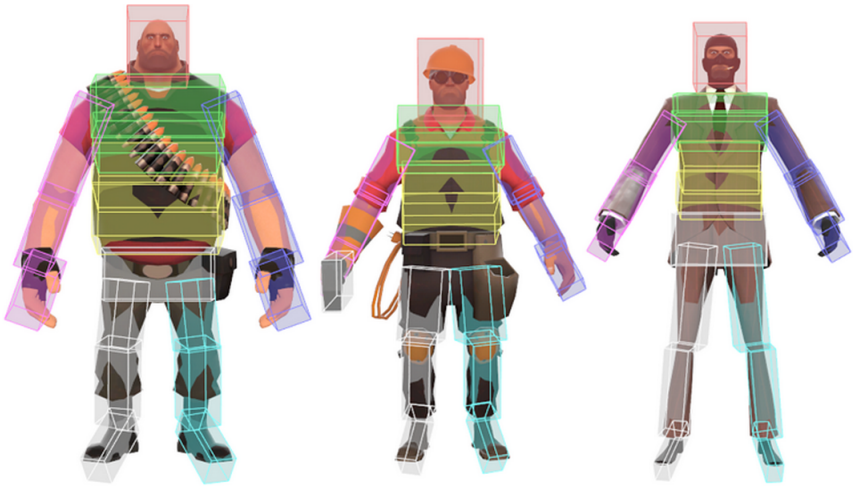


Figura 1.1: *Hitbox* per i modelli usati dai giocatori in Team Fortress 2.

Alcuni blocchi delle *hitbox* che coprono la silhouette del giocatore hanno un colore diverso dagli altri, per esempio la testa. Questo serve per far distinguere al server il valore dei parametri da rilasciare come output: Per esempio il danno

subito alla testa è maggiore rispetto a quello subito in altri punti del corpo. La differenza del tipo di *hitbox* serve anche per rilasciare tipi diversi di output per situazioni speciali: ad esempio la classe Spia, avendo equipaggiato il coltello, se si preme il pulsante d'attacco dopo essersi allineato alla *hitbox* presente dietro la schiena di un avversario, esso lo uccide istantaneamente senza valutare la quantità di salute che disponeva.

1.2.2 Proiettili e l'*hitscan*

In un FPS, tra le varie entità di gioco presenti, vi sono i proiettili, elementi necessari affinché un'arma da fuoco possa compiere il suo lavoro. Nei videogiochi a seconda da quale tipo di arma vengono sparati, i proiettili possiedono implementazioni diverse a seconda del loro scopo o forma: per esempio un proiettile di una pistola ha un'area d'effetto più piccola rispetto a quello di un esplosione di un lanciagranate. I proiettili vengono suddivisi in 3 tipi, ognuno della quale definisce il tipo di implementazione nel gioco: fisici (o normali), *hitscan* e particellari. I proiettili fisici sono entità con un modello visibile, interattivo e disposti di una *hitbox*. In questi casi determinare se il colpo è andato a segno dipende dalla posizione della *hitbox* del giocatore rispetto a quella del proiettile: ad esempio il razzo del Soldato e i colpi di lanciagranate del Demolitore sono fisici poiché possiedono un modello 3D e di conseguenza una *hitbox*. I proiettili *hitscan* sono quelli delle armi da fuoco: non sono veri proiettili, ma "puntatori" che vengono attivati nel momento in cui si apre fuoco. Anche se possiedono un output grafico che mostra il moto dei proiettili, essi non "esistono" nel mondo di gioco. Il funzionamento è tipo come i *laser games*, nel momento in cui viene aperto il fuoco contro un bersaglio puntato dall'arma, esso viene istantaneamente colpito confermandolo attraverso un output visivo e audio. I proiettili particellari si basano sull'uso di effetti particellari che definiscono una *hitbox* approssimativa; il funzionamento è simile ai proiettili fisici, con la differenza che l'*hitbox* non contorna un modello 3D ma vengono direttamente sparati dall'arma, generandone un output visivo. L'unico tipo di arma di Team Fortress 2 che funziona in questo modo è il lanciafiamme del Pyro.

Figura 1.2: Esempio di proiettili particellari. Gli effetti particellari del fuoco sono accompagnate da delle *hitbox* che entrando a contatto con quella di un avversario lo avvolge nelle fiamme

1.3 La gestione degli input di gioco

Negli FPS, il giocatore controlla un personaggio: un modello 3D (spesso) umanoide. Al giocatore è permesso controllare il movimento e le azioni del personaggio. Tutte le informazioni legate a quello che compie il giocatore vengono mandate al server che, una volta ottenute, aggiorna il mondo di gioco nel giro di qualche millisecondo. L'aggiornamento viene visto da tutti i giocatori, oltre a colui che ha compiuto l'azione. Basandoci come soggetto di studio il Source Engine, l'*engine* di gioco usato da Team Fortress 2, usa il proprio sistema di comunicazione server-client chiamato *Networking Multiplayer Source*. Durante la partita il server ha l'ordine di gestire gli input del gioco generate dai giocatori, riportandone gli appropriati output da implementare nei giocatori e nella partita. Gli input sono, in larga parte, generati dai giocatori che, interagendo e compiendo azioni, generano informazioni per la quale il server deve velocemente gestire. Una volta ottenuti i input, viene elaborato un insieme di output audio, grafici e script, per la quale vengono successivamente mandati a tutti i giocatori. Questo in modo da poter mantenere tutti gli utenti al corrente di tutti i aggiornamenti di partita che stanno succedendo in tempo apparentemente reale, come la morte di un giocatore oppure il completamento di un obbiettivo. Nel momento in cui il giocatore si connette al server, essi si mandano piccoli *packet* d'informazione i/o ad alta frequenza, circa 20/30 *packet* al secondo. Per definire meglio come funziona si ponga questo esempio: vi sono 2 giocatori, giocatore 1 e 2; se giocatore 1 spara al giocatore 2, il computer del giocatore 1 manda i *packet* di informazioni che il server richiede per poter applicare aggiornamenti

nell'ambiente di gioco, nel nostro caso, dopo aver letto l'input `+FIRE` mandato dal giocatore 1 [1], viene elaborato l'output di risposta e sistemato all' interno di un *packet* insieme agli altri per gli altri utenti[2]; il *packet* generato possiede l'informazione affinchè nel mondo di gioco si generi l'output elaborato, ovvero gli effetti sonori e visivi dello sparo e la generazione del proiettile adatto, nel caso di quelli *hitscan* viene generato il raggio invisibile[3]. Quando il raggio conferma la collisione contro un nemico[4], il server elabora l'informazione per entrambi i giocatori creandone un insieme di output da mandare nel prossimo *packet*[5]. Come alla fase 3 vengono consegnati tutti gli output del *packet*; nel nostro caso vengono consegnati al giocatore 1 e 2 gli output appropriati per segnalare il colpo a segno[6]: al giocatore 1 viene dato un'audio che segnala che un bersaglio è stato colpito, mentre al giocatore 2 viene dato l'aggiornamento della salute, un grido di dolore e una macchia di sangue nel punto dove è stato colpito.

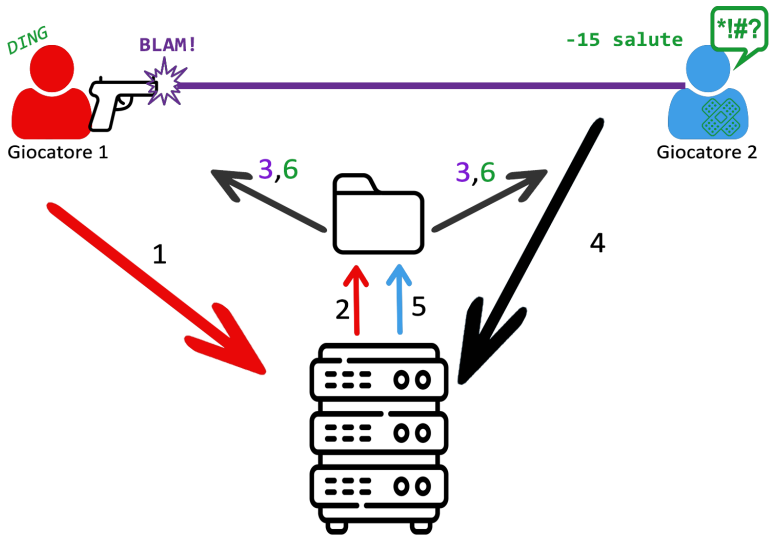


Figura 1.3

La gestione della comunicazione server-client in un videogioco tuttavia, non è così semplice: il server deve compensare il tempo di consegna dell'output agli utenti con vari livelli di latenza, in modo da mantenere il gioco il più leale possibile per tutti e per evitare problemi grafici e tecnici nel gioco. Siccome la lunghezza di banda è limitata, gli aggiornamenti non avvengono immediatamente, il server infatti crea degli *snapshot* di tutti i dati che deve mandare ai giocatori; in questo modo è necessario che i giocatori siano indietro di qualche millisecondo rispetto al server, affinché quest'ultimo abbia il tempo di mandare e ricevere le informazioni a e da tutti i giocatori. Tra le varie tecniche usate per la gestione dei giocatori con livelli di latenze variegate vi sono:

Interpolazione delle entità Invece di spostare l'entità da una posizione all'altra nel momento in cui viene renderizzata l'operazione dal client, l'interpolazione propone di tenere da parte lo *snapshot* dove ne è avvenuto e di farlo partire solo dopo che il server sia riuscito a raggiungere il tempo reale del client. Per esempio se un giocatore sposta con un colpo un barile dalla posizione A alla posizione B, il server non sposta l'oggetto a livello globale immediatamente: il giocatore vede che il barile si è spostato, successivamente il server compie il rendering della posizione a tutti i giocatori; se si perdono dei *packet* nel mentre, per via di una latenza scarsa dal giocatore, il barile rimarrebbe nel punto A. Questo serve per evitare *glitch* grafici.

Predizione dell'input La tecnica propone di attuare ed elaborare gli output del client localmente prima che vengano processati dal server e implementati nel mondo di gioco. Supponiamo di avere una latenza di 150 millisecondi e venga premuto il pulsante per andare avanti, mandando il segnale $+FORWARD$ al server. Senza la predizione, per poter vedere il proprio personaggio muoversi bisognerebbe aspettare non meno di 150 millisecondi per poter ottenere l'elaborazione da parte del server, generando un ritardo dei controlli che comporterebbe frustrazione al giocatore. Con la predizione, il comando, dopo aver mandato l'input al server, elabora localmente il movimento, per fare in modo che quando arriverà l'output esso venga integrato in modo quasi impercettibile. Questo serve per poter mantenere il gioco più fluido per il client anche se diminuisce con l'aumentare della latenza. Sono possibili errori di predizione, generati solo nella sfera locale, evitando problemi di *gameplay*. È impossibile implementare la predizione degli output nel server, poiché richiederebbe saper leggere il futuro.

Bilanciamento *lag* Supponiamo un giocatore colpisca un altro al tempo client di 10.5. Tra il colpo a segno e la generazione dell'output, il bersaglio si è spostato. Senza la compensazione, l'informazione arrivata a tempo di server 10.6, fa implicare che il colpo non è andato a segno. Con la compensazione uso l'interpolazione per poter avere la storia di tutte le posizioni dei giocatori; ad ogni comando arrivato da un client viene fatto un calcolo per stimare quando è stato fatto: `Command Tempo d'esecuzione = Tempo server - Latenza Packet - Interpolaz. client.` Una volta fatto, si spostano solo i giocatori indietro nel tempo e si controlla se il colpo è andato a segno valutandone la posizione nel mondo rispetto a quella del proiettile.

Usando questi sistemi è apparente come l'*hitbox* locale e quello che vede il server non siano uguali. Questo perché bisogna dividere la *hitbox* 'visibile' dal giocatore con quella visibile dal server, per la quale la posizione tiene conto della

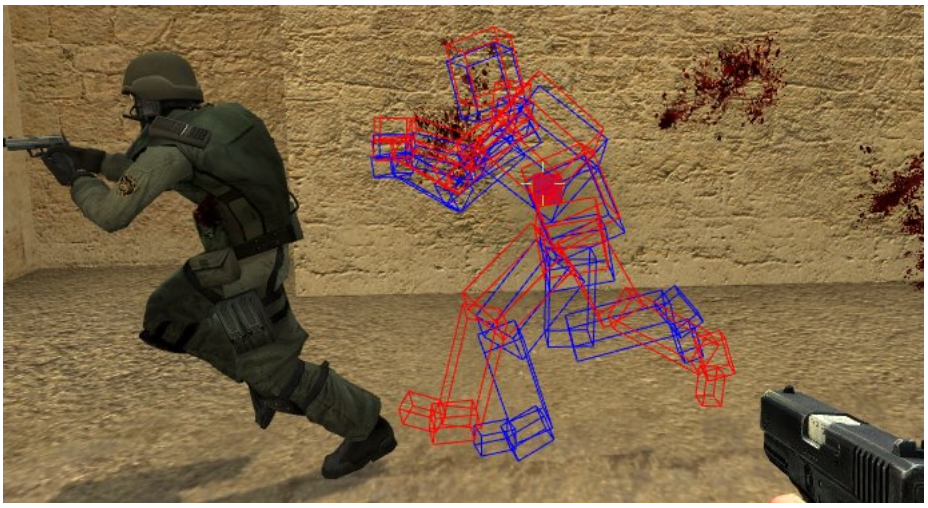


Figura 1.4: L'esempio mostra la situazione in un server con 200 millisecondi di *lag*. I rettangoli rossi mostrano la posizione del giocatore 100ms + interpolazione prima, i blu la posizione ottenuta con la compensazione.

latenza del giocatore. Si potrebbe pensare che una soluzione è quella che il client possa mandare l'informazione del colpo: ovvero quale giocatore è stato colpito e dove. Anche se questo sistema è molto più preciso di quello attuale, esso si basa sull'idea che tutti i giocatori siano onesti. Se il giocatore avesse questo tipo di potere, potrebbe manipolarne i dati che vengono mandati al server: per esempio si potrebbe fare un attacco *man-in-the-middle* usando un software esterno di cheat che ogni volta che il giocatore spari modifichi il *packet* affinchè il colpo sia andato a segno. Pertanto non è possibile che il giocatore (client) sia colui che dia il verdetto su queste cose ma il server (l'arbitro). Il multiplayer di Diablo aveva un problema simile: esso usava un sistema *peer-to-peer* per la gestione dei giocatori, nel senso che il server era solo un mediatore che permetteva la connessione tra i giocatori e non gestisce nessun mondo di gioco. Siccome quindi le informazioni che venivano trasmesse si basavano su quelle locali dei giocatori, un modo per barare era quello di modificare i file di gioco affinchè le armi facciano più danni rispetto a quello che dovrebbero. Nel momento in cui giocatore A dice al server che "ha colpito B con X danni", il server riferisce e fa modificare i parametri localmente a B. Come vedremo più avanti esisteranno software che permettono di compiere questo tipo di operazione e tanto altro.

1.4 Il bot

Spesso il videogioco per poter istruire il giocatore alle sue meccaniche richiede dei 'fantocci' con cui addestrarsi, ovvero un Bot. Il Bot (chiamato anche computer/CPU) è un giocatore controllato dall'intelligenza artificiale che simula il comportamento dei giocatori. La natura prevedibile di questi algoritmi permettono una base di addestramento utile al giocatore neofita. I Bot vengono inoltre usati per simulazioni di gioco e per le operazioni di debugging e testing, motivo per cui si distinguono dagli NPC(*Non-Playable Character*), personaggi non controllabili dal giocatore ma che ne interagiscono. I Bot di gioco si distinguono rispetto ai Bot di *farming* creati dagli utenti: quest'ultimi non sono built in nel gioco, ma sono 'giocatori' controllati da algoritmi con scopo è quello di occupare un posto in partita senza venir buttati fuori. Il *farming* è operazione che consiste nel ripetere per lunghi tempi la stessa operazione guadagnandone delle risorse. Un esempio sono i *farming bots* di Counter Strike: Global Offensive, che rimangono nelle partite per ottenere esperienza per salire di livello e vendere l'oggetto ottenuto grazie al livellamento.

1.5 Team Fortress 2

Team Fortress 2 è un FPS sviluppato dalla Valve rilasciato il 10 ottobre del 2007. Ogni giocatore viene diviso in due squadre, squadra rossa e blu; lo scopo del gioco è quello di battere la squadra avversaria secondo l'obbiettivo della partita. Tra le varie modalità di gioco vi sono cattura la bandiera, controllo dei punti e re della collina; tipiche modalità di qualsiasi altro FPS Online a squadre. Ogni giocatore deve scegliere la classe con cui entra in partita. Le classi sono personaggi che possiedono abilità uniche fra loro, affinché possano ricoprire diversi ruoli per la partita: in totale sono 9 classi diverse divisi per 3 categorie, offensiva, difensiva e di supporto. I ruoli si distinguono per il tipo di abilità e armi che dispongono.

Esploratore Classe con *hitbox* molto piccola ed alta velocità di movimento, dispone di un fucile a pompa con un ampia rosa di colpi. Usato principalmente per il completamento degli obbiettivi e per uccidere giocatori solitari.

Soldato Dispone di un lanciarazzi usato per compiere operazioni di 'controllo della folla'.

Piro Dispone di lanciafiamme che con la sua versatilità porta pressione contro i nemici.

Demolitore Dispone di lanciagranate e di lancia-mine che si attaccano alla superfice, in modo da poter distruggere strutture nemiche e difendere obbiettivi.

Grosso Classe con la salute più alta e velocità di movimento più bassa, dispone di una mitragliatrice a cannne rotanti. La classe viene usata come 'carro armato' della squadra.

Ingegnere Classe che permette di costruire strutture, come torrette da combattimento, macchine di apporvvigionamento e teletrasporti.

Medico Usa la 'pistola medica' usata per poter curare gli alleati. La salute si rigenera automaticamente nel tempo.

Cecchino Possiede il fucile da cecchino, permettendo i colpi alla testa che dispensano il doppio dei danni.

Spia Classe con poca salute con l'abilità di travestirsi come i giocatori nemici. Possiede un coltello che permette di uccidere gli avversari se usato dietro la loro schiena.

Figura 1.5: Le 9 classi di Team Fortress 2. In ordine da sinistra a destra: Piro, Ingegnere, Spia, Grosso, Cecchino, Esploratore, Soldato, Demolitore e Medico.

Il fine del gioco è quello di scegliere la classe adatta in modo tale da poter cooperare efficientemente con gli alleati per poter completare gli obbiettivi e vincere il gioco. In passato Team Fortress utilizava un browser dei server di gioco, lasciando la scelta della modalità e della mappa al giocatore. Oltre ai server uffciali, Team Fortress permette ai giocatori di generare diversi server personalizzati. I cosiddetti server della comunità si distinguono da quelli ufficiali per via della possibilità di di poter implementare, oltre a diverse modalità di gioco e mappe, plug-in che modificano o migliorano lo stile di gioco. Dal 2016, il browser venne rimpiazzato con il sistema di *matchmaking*, che permette di accodare il giocatore nella ricerca degli altri con preferenze di gioco simili (mappe, modalità etc..); trovati abbastanza giocatori il server crea la partita e li inserisce nel gioco.

Man mano che si completano match si guadagnano punti che aumentano il livello di esperienza del giocatore; questo serve al sistema di *matchmaking* per poter raggruppare facilmente con avversari dello stesso livello d' abilità. Per poter giocare a Team Fortress 2 è necessario avere un account Steam, piattaforma di vendita e *launcher* di videogiochi sviluppato dalla Valve Software. Steam, oltre ad offrire servizi di *e-commerce* per videogiochi, possiede un hub per interagire con i giocatori nella piattaforma, condividendo immagini, video, guide e forum per i vari giochi presenti nello store. Il gioco permette l'uso di oggetti cosmetici e possiede un sistema di scambio tra giocatori che, col tempo ha generato un mercato di scambio, completo di venditori online e valute (ovvero i metallo di scarto e chiavi, oggetti ottenibili e/o acquistabili nel gioco). Dopo 15 anni Team Fortress 2 rimane ancora un gioco densamente popolato: nel giugno del 2022 il numero di giocatori medi in gioco stava a 87 mila (fonte steamcharts.com). Uno dei motivi di questo influsso di giocatori è dato dalla scelta del gioco di essere gratuito dal 2011 (ovvero *Free-To-Play*). Come si vedrà successivamente, questa decisione sarà uno dei motivi del problema che affligge il gioco.

1.5.1 Il codice sorgente

La Valve tra il 2017 e 2018 donò i codici sorgenti di Counter-Strike: Global Offensive e Team Fortress 2, ad un gruppo di sviluppatori indipendenti, la Lever Softworks, per un progetto di ricreazione di alcuni giochi cancellati dalla compagnia. Entrambe le build vennero create nel 2017 e veniva continuamente scambiato tra gli utenti della comunità, ma non furono mai pubblicate ufficialmente. Questo fino al 21 Aprile 2020, quando venne rilasciato al pubblico i due codici sorgenti da parte di un ex collaboratore della Lever Softworks. Il movente fu' per tornaconto personale per essere stato allontanato dal progetto a causa di comportamenti inadatti. Secondo gli sviluppatori di Team Fortress, il codice non è quello ufficiale, ma quello donato ai collaboratori richiedenti nel 2017; questo per poter spiegare che non vi erano veri e propri rischi di implementazione di RCE maligni (*Remote Code Execution*), tipo di operazione che permette di poter far attivare codici potenzialmente malware da remoto a tutti coloro che si connettono ad un qualsiasi server pubblico. Seppur questo rischio venne scampato, il leak ha sicuramente incoraggiato più utenti a sviluppare ed usare nuovi cheat e sistemi per generare la futura invasione a cui Team Fortress non era ancora pronto a difendersi.

Capitolo 2

Il Cheat

Per ogni gioco che contiene una certa natura competitiva vi saranno sempre i giocatori che rompono le regole per ottenere la vittoria, concetto presente anche nei videogiochi. Barare nei videogiochi esiste fino agli albori: dal comporre una serie di tasti per poterli usare all' attaccare uno strumento aggiuntivo nella cartuccia di gioco per poter accedere ai cheat, come per esempio il Game Genie per le cartucce di gioco dello SNES. Ma perché barare? Il motivo dell'uso dei cheat è vario; secondo Mia Consalvo e Irene Serrano V., tra i tanti possibili moventi, ve ne sono stati identificati 4: Il primo motivo riguarda il poter superare un problema, per esempio trovare la soluzione di un enigma del gioco oppure la direzione per poter avanzare. In questo caso si fa riferimento a delle guide, sia online che cartacee come i *walkthrought* (in italiano, accompagnamento) o guide ufficiali. Il secondo motivo è per poter avere accesso a tutti gli oggetti del gioco che richiederebbero tempo e dedizione al gioco per ottenerli normalmente. Il terzo motivo è per evitar di perdere tempo nel compiere le stesse azioni per lunghi lassi di tempo con l'intento di ottenere risorse necessarie all'acquisto di oggetti di alto valore. Questo sistema si distingue in 2 metodologie: il *farming*, che consiste nella ripetizione di una o più azioni generandone risorse, e il *looting*, combattere contro gli stessi nemici che rilasciano le risorse. Il sistema viene attuato mediante dei algoritmi, questo perché sono bravi a fare 2 cose: compiere operazioni semplici e compierle per lunghi lassi di tempo. Infine il quarto motivo è per poter avere un vantaggio rispetto agli altri giocatori, facendo entrare in gioco sia i cheat che gli *exploit*, che sono lo sfruttamento di errori di programmazione o della mappa per trarne un vantaggio di gioco contro gli avversari. Per quanto queste motivazioni siano vere e ben presenti, per la situazione studiata vi è un altro fine, ovvero il gusto di infrangere le regole e vederne le reazioni, chiamato in gergo *griefing*. Attraverso l'uso di tool che descriveremo, è possibile a questi

giocatori di godersi il gioco in un modo unico e più personale, per esempio togliere la gravità al gioco per poter simulare l'effetto di essere sulla Luna (denominato, *Moon Gravity*). Ma il divertimento generato dal *griefing* si basa su una natura burlesca, usando i cheat per dar fastidio agli altri giocatori ed osservare le reazioni che comportano: secondo un intervista fatta ad alcuni cheaters, uno dei motivi per usare i cheat è quello infastidire i giocatori in modo tale da ottenerne una reazione; altri invece ammettono di usarli come strategia di gioco per risolvere situazioni troppo difficili; altri lo fanno per venir considerati. Per questi giocatori il cheat è un modo di «divertirsi al gioco in un modo diverso dal solito». Che sia fatto per schernire gli avversari, alimentare il proprio ego o superare un ostacolo, questo sistema richiede nella maggior parte dei casi un software di terze parti per poter funzionare, presenti sia a pagamento che gratuiti.

2.1 Il Cheat come strumento di debugging

Se il cheat può essere dannoso, specialmente nei giochi multigiocatore, allora perché esistono? Il cheat nascono come strumento di debugging per gli sviluppatori; secondo il ricercatore Nguyen Bao, tra i vari tipi di debugger che un gioco dovrebbe avere vi sono quelli per il gameplay. Questo tipo di debugger, insieme agli altri che compongono il debugger di gioco

> "Giocano un ruolo importante per l'industria nello sviluppo del videogioco. Offrono un metodo veloce ed efficente per il processo di produzione di un gioco prima del rilascio sul mercato, specialmente per i programmatori."

. Questo tool di debugging possiede comandi che facilitano la ricerca di problemi da risolvere nel gioco a livello di *gameplay*, come ad esempio l'invincibilità e/o munizioni infinite per le armi per poter raggiungere velocemente una sezione del gioco. Alcuni videogiochi mantengono accessibili i tool di debugging usati per la creazione del gioco; nel nostro caso, Team Fortress 2 usa la console dello sviluppatore. La console dello sviluppatore è il classico strumento di debugging usato nella maggior parte dei videogiochi: si presenta come una finestra simile ad un terminale, contenente il log contenente informazioni di output e una riga di input per i comandi. L'accesso pubblico di questo strumento serve per permettere agli utenti di modificare parametri avanzati come per esempio la posizione della telecamera di gioco o la posizione dell'arma. I sviluppatori, nel caso in cui rilasciano il tool al pubblico, si accertano di mettere delle limitazioni nell'uso dei comandi: nel caso di Team Fortress 2 e di tutti gli altri giochi che usano lo stesso *engine* di gioco (Source Engine), per poter usare alcuni comandi speciali, è necessario digitare nella console `sv_cheats 1`; questo mette il booleano `sv_cheats` a `true`,

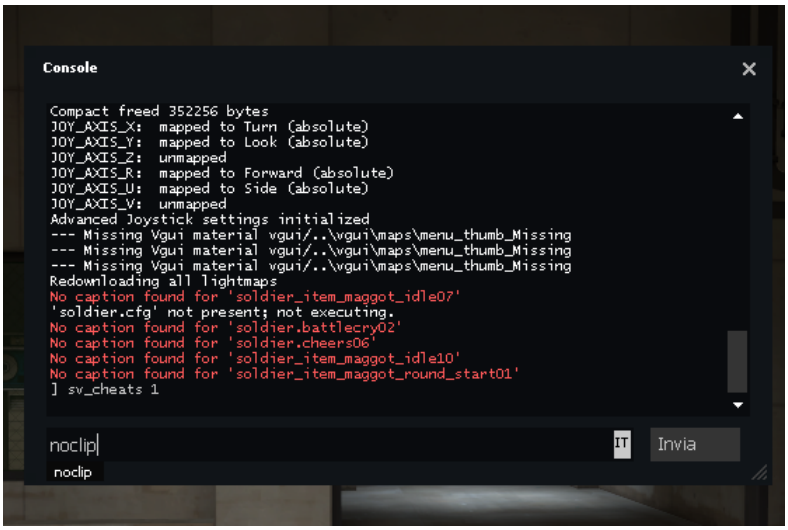


Figura 2.1: La console dello sviluppatore per Team Fortress 2. Vi è la finestra di output superiore e una riga per l'input dell'utente

che permette quindi di poter usufruire dei comandi identificati come cheat nel server attuale, per esempio *noclip*, comando che permette di poter far volare e attraversare le superfici al giocatore. Tenendo come esempio Team Fortress 2, questo è possibile solo se colui che ha scritto il comando è il padrone del server, ovvero l'host, onde evitare di consegnare privilegi illegittimi a chi non dovrebbe permetterselo. Nel caso in cui il tool di debugging non venga rilasciato al pubblico l'unico modo per accedervi è attraverso l'*hack*, ovvero l'azione di modificare il codice sorgente, per poter accedere al tool.

Figura 2.2: Due immagini comparative per distinguere la differenza cambiando i valori di `viewmodel_fov` da 70 a 100

2.2 I Cheat di Terze Parti

Lungo il corso degli anni, programmatori indipendenti hanno sviluppato complicati software che permettono l'uso dei cheat nei videogiochi, chiamati *cheat engines* o *trainers*. Tra questi uno dei più famosi è proprio chiamato Cheat Engine, software open source usato per poter superare le limitazioni di un videogioco singleplayer: esso permette di poter sbloccare i tool di debugging descritte nel capitolo precedente. Il fine di un *Cheat Engine* è quello di permettere l'uso di istruzioni privilegiate manipolando l'accesso alla memoria usata dal gioco. Anche se il programma è usabile solo per giochi non multiplayer, la tecnica di lettura della memoria e di manipolazione è la chiave per poter creare il software di cheat attuali. I *cheat engine* multiplayer non sono troppo difficili da trovare sia quelli gratuiti che quelli a pagamento. Oltre a *cheat engine* pubblici vi sono anche quelli privati, rilasciati da coder indipendenti. Un esempio di *cheat engine* per Multiplayer è Perfect Aim, sito che supporta diversi tipi di cheat per ben 21 titoli diversi, incluso Team Fortress. Un altro sito è il forum SkyCheats, che conta il supporto di *cheat engine* per 35 giochi ed un numero di 180mila membri. Il Cheat Engine più famoso specifico per Team Fortress 2 è LMAOBOX. LMAOBOX è il più popolare per la scelta di essere sia a pagamento che gratuito. Quando un utente usa la versione gratuita è molto palese l'uso di cheat, ma la versione a pagamento, invece, è altamente funzionale, costantemente aggiornato(l'ultimo aggiornamento è avvenuto il 13 ottobre) e soprattutto impercettibile dai giocatori se ben programmato.

2.3 Unire il Bot con i Cheat

Se molti utenti decidono di usare i cheat personalmente nei loro match, altri decidono di farlo via proxy creando armate di algoritmi, conosciuti come bot. Come è stato menzionato prima, esistono algoritmi che, in automatico, compiono azioni di gioco per lunghi lassi di tempo, mantenendo comunque un certo livello di comportamento simile a quello umano per camuffarsi. Molti di questi bot vengono usati per il *farming* e il *looting*, tipico nei giochi MMO(*Massively Multiplayer Online games*, in italiano, giochi multiplayer su larga scala) come World of Warcraft. Tuttavia esistono dei tipi di bot che hanno il compito di dar fastidio attraverso l'uso di Cheat via software esterno. I bot infatti usano vari tipi di cheat non presenti nel gioco di base, come per esempio colpire in automatico la testa di un giocatore senza dover muovere il mouse. Gli altri script vanno dall'essere meno intrusivi come ad esempio lo *spam* nella chat testuale e/o vocale, a veri e propri attacchi tipo *lag*, manipolazione dell'audio dei giocatori e *denial of service* per i giocatori.

2.3.1 Bot Hosting

Il Bot Hosting, consiste nel creare un ambiente atta alla generazione e gestione di bot baranti da rilasciare in un videogioco. Come i cheater, i bot hoster compiono quest'operazione per *griefing*. Non sempre è cosi: alcuni lo fanno per attivismo, per poter spingere la Valve ad aggiornare il gioco , altri invece lo vedono come uno strumento di popolarità, godendosi delle lamentele dei giocatori tra i vari forum e siti di discussione. Alcuni software per la creazione e gestione dei bot vengono rilasciati su vari siti, a volte anche in quelli legati ai *cheat engines*. Tra questi vi è catbot, software che permette di preparare un ambiente di bot hosting in Linux specifico per Team Fortress 2. Per poter creare molteplici istanze di bot vengono usate le macchine virtuali insieme ad un *cheat engine* a scelta. I bot vengono poi "immagazzinati" (hosted) nel cloud o in server per possibili "invasioni" future. Nessun software o sito di bot hosting tiene traccia del numero di bot generati, ma la quantità di memoria richiesta per poter far funzionare 1 bot è di circa 700mb; potenzialmente quindi, in una memoria da 8gb è possibile mantenere fino a 7-8 bot contemporaneamente.

2.4 L'invasione dei Bot su Team Fortress 2

Figura 2.3: La tabella di punteggi con i giocatori presenti. I bot sono coloro con lo stesso nome e immagine di profilo (OMEGATRONIC). Quando sono abbastanza, si raggruppano e distribuiscono tra le squadre, usando come classe il cecchino (simbolo del mirino)

Dall'inizio del 2020, Team Fortress 2 sta subendo un attacco su larga scala da diversi tipi di bot baranti. Questa invasione origina dal 2018 col rilascio della

prima versione ufficiale del software open source di bot hosting per Team Fortress 2, cathook. A detta dei collaboratori del programma, il progetto era partito nel 2016 come un altro *cheat engine*. Col passare del tempo e la collaborazione di altri utenti, il software si è evoluto in uno strumento più sofisticato, arrivando così a creare il primo ambiente di bot hosting open source per Team Fortress 2. La motivazione della creazione del software non ha un vero fine concreto: il programma esiste per interesse personale degli sviluppatori. Data la natura open source del programma, molti utenti hanno iniziato ad usare questo software, modificandone valori e migliorando la compatibilità con altri *cheat engine*. Dopo il leak del codice sorgente del gioco, gli attacchi si intensificarono sempre di più, in modo da incrementarne la gravità dell'evento e il danneggiamento dell'immagine a scapito della comunità.

Capitolo 3

Contrastare il cheat

Per quanto ve ne siano di innocui al livello di *gameplay*, i cheat nei giochi multiplayer nella maggior parte dei casi vengono usati per motivi egoistici, comportando fastidio per i giocatori che lo subiscono. Avere questi poteri, conoscendo il vasto mondo dell' informatica, è possibile che alcuni utenti possano sfruttare comandi speciali per poter compiere azioni molto pericolose. Fortunatamente esistono strumenti implementabili esternamente o internamente al gioco che hanno il compito di limitare il numero di giocatori baranti nelle partite. Le tecniche vengono distinte dall' uso di software addizionale alla gestione diversificata a seconda dei valori posseduto da ciascun utente. Questo capitolo si concentrerà a parlare di alcune tecniche per la limitazione dell'interazione tra giocatori leali e sleali, ed ai sistemi per identificare coloro che usano i trucchi nei loro calcolatori.

3.1 Il ban

Secondo la definizione di Wikipedia, il termine ban (traducibile come "bandire") viene utilizzato per riferirsi ad una serie di atti in una piattaforma online che consentono di vietare l'accesso e/o l'interazione con gli altri utenti come conseguenza per non aver seguito le regole. Il ban di un utente si basa sull' username, IP o indirizzo email. Il ban può essere automatico o manuale, a seconda dell'implementazione e del tipo di atto commesso. Gli utenti possono contribuire al ban di un utente attraverso la segnalazione: più vi sono segnalazione, maggior probabilità che venga emanato il ban. Il ban può avere una durata temporanea oppure permanente. Il ban è un ottimo strumento per poter allontanare utenti che non rispettano linee guida di interazione sociale nei siti o servizi online, nel nostro caso in un videogioco. Instaurare un ban ad un giocatore (in gergo, *ban-*

ning, italianizzato in bannare) blocca o limita l'accesso al gioco e/o ai server da cui è stato emanato.

3.2 Anticheat

Con la continua minaccia dei cheat e l' immensa quantità di giocatori da controllare nel gioco, gli sviluppatori richiedono di inserire meccanismi di rilevamento e contrasto automatici in modo da poter star dietro ai cheaters. Il principale programma di contrasto che viene solitamente implementato è l'anticheat. L'anticheat è un software addizionale di un videogioco con il compito di limitare e potenzialmente rimuovere i cheater basandosi sul loro comportamento in partita. Il funzionamento varia a seconda dell'implementazione: per esempio esso può avere il meccanismo di ancoramento(*hooking*), può operare in user space oppure in kernel space. Normalmente, il software controlla se l'utente sta compiendo azioni non consentite attraverso l'analisi della memoria centrale e, prontamente, procede alla rimozione dalla partita con conseguente ban o sospensione dal gioco. Ogni anticheat possiede un livello di intrusione, elemento che definisce la mole di permessi ed accessi consentiti nel calcolatore per poter prevenire e identificare potenziali cheater.

3.2.1 Il meccanismo di ancoramento

In informatica, il termine *hooking* definisce una serie di tecniche per alterare il comportamento del sistema operativo intercettando le informazioni che vengono passate in memoria tra i dispositivi, come chiamate di funzioni e/o passaggio di messaggi. In un anticheat, l'ancoramento serve per poter controllare a livello kernel l'insieme di istruzioni eseguite comparando con le informazioni ottenute nel server della partita. L'analisi viene compiuta per poter identificare esecuzioni di operazioni sospette. Attraverso un esempio spiegato dai ricercatori Lehtonen e Samuli Johannes, in Windows vi sono 3 funzioni molto comunemente usati dai *cheat engine*: `WriteProcessMemory`, `CreateRemoteThread`, e `SetWindowsHookExA`. Nel momento in cui vengono usati, il comando viene fatto passare al driver di anticheat, affinché ne identifichi il funzionamento e del motivo per cui viene chiamato. Quando l'algoritmo di identificazione eseguito in locale conferma la presenza di cheating, manda un report al server adibito alla gestione delle segnalazione, comportandone poi l'insieme di punizioni generate da quest'ultimi, come l'uscita dalla partita e, potenzialmente, al ban. Questi anticheat dispongono di Database da cui prendere riferimento per poter cogliere le *system call* tipicamente usate dai cheat. La tecnica ha generato polemiche per la sua natura intrusiva, simile al funzionamento di una *rootkit*: ovvero un tipo di

malware che sfrutta il sistema di *hooking* per poter ottenere il controllo del sistema operativo della macchina senza venir scoperto dal client. Vi è da aggiungere, inoltre, che alcuni cheat funzionano a livello di kernel per poter mascherare le proprie operazioni e permettendo così di poter superare i controlli dell'anticheat.

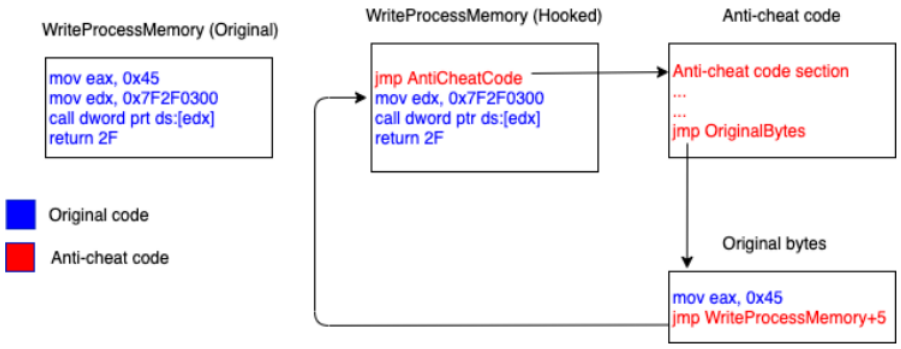


Figura 3.1: Due esempi di esecuzione comandi senza *hooking* e con. Come spiegato da Lehtonen e Samulu Johannes, quando si esegue un comando sospetto, il driver di anticheat attua il comando `jmp`, che permette di passare all'indirizzo dove vi è il programma di controllo per cercare se vi sono usi di programmi di cheating.

3.2.2 Il Database

In aiuto per l'anticheat, il server con cui comunica possiede un Database dei comandi 'sospetti'. Il Database, oltre a contenere le combinazioni di operazioni usate dai *cheat engine*, contiene altri tipi di informazioni con la quale sono stati associati col comportamento degli engine di cheating precedentemente segnalati dal software di contrasto. I dati ottenuti dai calcolatori dei bari vengono registrati e categorizzati per tipo e contesti in cui opera: questo serve per future analisi dei giocatori per poterli identificarli e punirli facilmente e velocemente. Esistono anche Database pubblici dei giocatori segnalati per aver usato cheat, utile per gli altri giocatori per poter facilmente identificare i potenziali bari nelle partite. Per quanto possa essere utile il Database dei giocatori bannati, contro i bot diventa obsoleto: a differenza di un essere umano, l'algoritmo non ha problemi di perdita di progressi e di duplicazione dell' account. Bannare un bot è come pulire la sabbia dalla spiaggia.

3.3 Anticheat con *deep learning*

Il *deep learning* è un metodo di creazione di intelligenza artificiale attraverso il consumo di dati da parte della macchina. In altre parole, è far imparare la macchina a riconoscere e compiere operazioni attraverso esempi che vengono fatti ingerire alla macchina: invece di creare una macchina che riporta l'output y=f(x) secondo un input x, ne creo una che mi restituisce la f per ottenere la y. Secondo questo sistema la macchina, dopo aver osservato migliaia di esempi, impara a definire e ricreare informazioni per la quale non riuscirebbe a fare attraverso la semplice programmazione. Un esempio è Dall-e, IA che a seconda della frase inserita dall'utente tenta di ricreare visivamente quello che è stato descritto dall'input mischiando appropriatamente le immagini presenti online. Questa tecnologia può essere implementata per molti scopi, sopratutto per l'anticheat: creare una macchina *deep learning* in grado di riconoscere pattern di gioco per poter distinguere i giocatori normali da quelli baranti. Per poter creare un anticheat *deep learning* servono 2 risorse, la tecnica *feature learning* (o *representation learning*) e il *big data*. Il *big data*, in italiano mega dati, è una raccolta di dati informatici estesa in termini di volume, velocità e varietà. Questi dati nel nostro caso sono clip di gioco generali, contenente sia giocatori normali che quelli che usano cheat. Con il *big data* si passa poi al *feature learning*, tecnica che mette in input le informazioni presenti nel *big data* affinché la macchina riconosca i pattern per poter riconoscere ed in alcuni casi emularne il comportamento, in altre parole il computer viene "allenato" a riconoscere i cheater: nel caso dei dati a cui facciamo riferimento, gli sviluppatori definiscono quando avviene il cheat nella clip video. Il processo richiede molto tempo e molti dati, data la sua natura. Yoshua Bengio, Aaron Courville e Pascal Vincent descrivono meglio il problema del *feature learning* spiegando come aumentando il livello di astrazione o numero di variabili all'interno di un contesto, genera maggior imprecisione. Precisamente:

> Significa che due input possano venir interpretati in modo completamente differente[...] Le rappresentazioni associate con i molteplici input possono essere complessi per via della mappatura tra l'input e le rappresentazione che potrebbero falsare e rivelare input che hanno forme generalmente complicate in distribuzioni semplici, dove le relazioni tra le variabili sono facili e con molte condizioni. (Bengio Y., Courville A., Vincent P., 2013, Sezione 10)

Questo significa che, per poter funzionare al meglio, la macchina richiede dell'input umano, sia per donare dati che per definire gli sbagli dell'IA, generando così molti tentativi e quindi richiederebbe molto tempo per poter essere completamente addestrato. Una volta ottenuta un anticheat basato su *deep learning* esso lo si può usare per poter valutare situazioni di cheat per cui la macchina non si

è addestrata e vedere se riesce a comportarsi nel modo appropriato.

3.4 *Valve Anti-Cheat*

Team Fortress 2 usa come Engine di gioco il Source Engine e, come tutti gli altri giochi che lo usano, possiede come anticheat il VAC (*Valve Anti-Cheat*). Il VAC è un anticheat sviluppato dalla Valve Software, stessa produttrice del Source Engine. Il VAC venne implementato per la prima volta nel 2001 per lo sparatutto Counter Strike; inizialmente l'anticheat, una volta identificato lo sfruttatore, bandiva il giocatore dal gioco per un tempo di 24 ore; tutt'oggi, il ban è diventato permanente. Il suo funzionamento non è stato mai completamente spiegato dagli sviluppatori, in modo da mantenere all'oscuro gli hacker desiderosi di aggirarlo. Un altro meccanismo che il VAC ha sperimentato per identificare i potenziali cheater è il recupero di dati degli ultimi siti acceduti dal calcolatore: dal 2014, VAC spediva i dati della cache DNS agli sviluppatori per controllare potenziali acquisizioni di cheat. Gli indirizzi IP associati alla distribuzione di cheat vengono registrati in appositi server speciali, ovvero i DRM(*Digital Rights Managment*); nel momento in cui notano che l'utente ha compiuto certe interazioni con un sito registrato, la Valve intensificava le analisi dell'utente per poi emanarne un ban. Questo esperimento tuttavia ebbe vita breve per due motivi:

1. Gli hacker si accorsero dello scan, portando quindi a modificare la cache della DNS dei propri clienti per non farli scoprire.
2. La compagnia perderebbe credibilità e supporto se venisse fuori che gli sviluppatori stanno spiando i siti visti dagli utenti. Secondo Gabe Newell fondatore della Valve Software: «Se venisse fuori l'idea(degli utenti) che 'Valve è crudele - Stanno tenendo traccia dei siti dove andiamo', da' molta attenzione e beneficio ai cheater ed ai creatori di cheat»[2015]

3.4.1 Funzionamento

L'anticheat manda una *challenge* al giocatore sospettato: se viene riportata la risposta sbagliata, il giocatore viene segnalato come potenziale cheater. L'anticheat procede successivamente con una analisi dei processi e della memoria. Se viene riscontrata una anomalia, viene mandato un avviso agli sviluppatori insieme alla parte di codice con l'anomalia. Gli ingegneri testano il codice e comparano l'anomalia con i software precedentemente riscontrati registrati nel database; se non presente, viene aggiunta l'anomalia nella lista. Se il cheat viene trovato, il giocatore viene marchiato ma non bandito, per fare in modo che non se ne accorga. Il ban può avvenire in un tempo indeterminato che si dirama dai giorni

ai mesi. Una volta avvenuto il ban dal gioco, viene inserito in modo permanente un avviso nel profilo dell'utente che identifica il numero di VAC ban ottenuti e le ore passate dall'ultimo ricevuto; il cheater non può nascondere questo avviso in nessun modo e non potrà partecipare a nessun evento importante usando questo profilo(campionati, tornei ufficiali etc..).

3.5 Counter Strike: Global Offensive

Counter Strike: Global Offensive è un FPS Multiplayer gratuito sviluppato e rilasciato dalla Valve nel 2012. Il gioco è attualmente il più giocato nella piattaforma Steam, con un numero di giocatori consecutivi che arriva dai 500 mila ai 1.3 milioni. Il videogioco possiede alcune cose in comune con Team Fortress 2 : il gioco è gratuito, è un FPS dove sono possibili le uccisioni con singolo colpo alla testa, i giocatori vengono divisi in squadre raggruppate attraverso il *matchmaking* e infine possiede oggetti cosmetici e un sistema di mercato. Eppure per molti aspetti la situazione su Counter Strike è nettamente diversa rispetto a quella di Team Fortress. Sebbene alcune similarità, oltre al fatto che sono stati sviluppati dalla stessa compagnia, i due titoli possiedono alcune differenze importanti:

***Matchmaking* Prime** Entrambi i titoli possiedono questo sistema ma Counter Strike ne dispone di due tipi, quello normale e Prime. In origine Counter Strike era a pagamento, ma quando avvenne la transizione a gioco gratuito i sviluppatori hanno voluto creare un sistema per "far valere i soldi" per coloro che in passato lo hanno acquistato. Quindi il sistema Prime, oltre a dare altri vantaggi minori, permette al giocatore di poter usare un *matchmaking* esclusivo per coloro che hanno acquistato il gioco in passato oppure comprato il Prime. Questa è una soluzione marginalmente ottimale poiché costringerebbe i bot a comprare il Prime per poter "giocare" contro i giocatori normali invece di venir relegati nei match del *matchmaking* gratuito spingendo però i giocatori a comprare i privilegi del Prime.

Competitività e Popolarità Anche se entrambi i giochi hanno una scena competitiva, il livello di seguito tra i due giochi è nettamente marginale. Il campionato di Team Fortress 2 con più spettatori fu' Imsonnia del 2018 con quasi 13 mila persone; La Major di Stoccolma per Counter Strike del 2021 ebbe un massimo di spettatori di circa 2.74 milioni. La ragione dei risultati di questi dati è molto semplice: Team Fortress 2 non è nato come un gioco competitivo a differenza di Counter Strike. Per cui Valve si è concentrato maggiormente su quest'ultimo per mantenerne l' incremento dei giocatori.

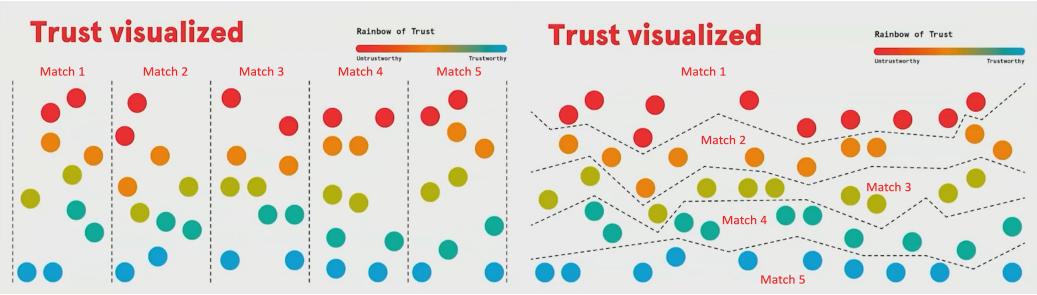
L'uso di nuove tecnologie Affinché Counter Strike mantenga una scena competitiva ben attiva, data la natura altamente lucrativa che ne deriva, Valve ha creato interessanti sistemi per il contrasto di coloro che barano. Essi sono il sistema di Overwatch e il Trust Factor. Nel gioco inoltre vi è una versione ancora migliorata del VAC che usa il *deep learning* chiamato VACNET.

3.5.1 Trust Factor

Dal 2016 in Counter Strike: Global Offensive quando un giocatore si accoda per il *matchmaking* il gioco analizza, oltre i parametri del tipo di partita selezionati, dei valori speciali assegnatogli dagli sviluppatori chiamato Trust (o affidabilità). Il Trust Factor è un insieme di valori non modificabili e visionabili dal giocatore che approssimativamente descrivono lo stile di gioco del giocatore e la sua "sportività". Seppur non vi è una lista precisa, i fattori che vengono valutati si basano sul tempo di gioco, statistiche di fine partita e interazioni nelle partite e all'interno della piattaforma di Steam. A seconda del tipo di comportamento, il *matchmaking* raggruppa i giocatori che possiedono un livello di affidabilità simile, dividendo così i giocatori onesti contro coloro che hanno comportamenti scorretti, tipo nocività e/o uso di cheat. Il Trust Factor non identifica chi bara, ma coloro che hanno un comportamento che potrebbe portare queste persone a barare, come avere molteplici account diversi con e senza ban, linguaggio scurrile contro altri giocatori, sfruttare *glitch* e errori di gioco come vantaggio ed altro.«La cosa bella di Trust è che non riduce la percentuale di cheat ma riduce l'impatto delle persone che li usano»(John McDonald, 2018, Conferenza GDC)

3.5.2 Overwatch

I dati ottenuti per poter creare VACNET provengono da un sistema implementato nel 2014 chiamato Overwatch. Overwatch è un programma pensato dalla Valve, che permette alla comunità di Counter Strike di poter analizzare clip di gioco e valutarne la presenza e conseguente uso dei cheat. Le clip sono lunghi quanto la partita, permettendo di poter visionare sia il campo da gioco che ogni giocatore presente; agli analisti non viene mostrato il nome dei giocatori per accertarsi che il verdetto sia il più onesto possibile. Durante la visione della clip, gli analisti tengono nota delle attività dei giocatori via un modulo: esso chiede informazioni sull'uso di cheat nel gioco, come assistenza per la mira, per il movimento, visiva oppure *griefing*. Se nella clip viene trovato uno o più sospetto che subiscono abbastanza segnalazioni come bari, vengono emanati i ban. Inizialmente il sistema ha accettato un piccolo numero di collaboratori per poterne testare il programma: il progetto dimostrava i suoi frutti dimostrato



(a) *Matchmaking* senza Trust Factor (b) *Matchmaking* con Trust Factor

Figura 3.2: Funzionamento del *Matchmaking* con e senza Trust Factor. I punti rappresentano i giocatori e la loro affidabilità attraverso la variazione di colore(dal massimo di colore blu al minimo in rosso). Attraverso il Trust Factor i giocatori con livelli di affidabilità simili vengono raggruppati tra di loro nelle partite, a differenza dei normali *matchmaking*.

dall'aumento del numero di cheater confermati e bannati. Dalla metà del 2015, Valve annuncia la partecipazione pubblica al programma, aumentandone esponenzialmente il numero di partecipanti. Ben presto, però, il numero di conferme rimase allo stesso livello rispetto a quando vi erano pochi partecipanti. secondo uno degli sviluppatori di Counter Strike John McDonald, durante la conferenza al GDC del 2018, il motivo era per il tipo di cheat che venivano analizzati: le conferme rimanevano sempre allo stesso livello perché venivano confermati solo quei cheat che erano ben palesi, come l'*aimbot*. Il numero di dimissioni divenne alto quasi quanto il numero totale dei casi poiché, sempre secondo McDonald, vi era difficoltà da parte dell'umano nel distinguere con certezza e senza ragione d'alcun dubbio tra un giocatore onesto molto bravo con quello che nasconde molto bene l'uso di cheat.

3.5.3 VACNET

Il VACNET è la versione migliorata del VAC Anticheat attualmente in uso su Counter Strike: Global Offensive. Quel che rende speciale questa versione rispetto a quello originale è per l'implementazione del meccanismo di *deep learning*. VACNET è un software anticheat che sfrutta la tecnologia del *deep learning* per poter trovare facilmente i cheater. Per poter ottenere la quantità di dati necessaria per potersi allenare, la Valve decise di inserirlo all'interno del programma Overwatch. Una volta che la macchina ha visionata la clip e elaborato la risposta, gli sviluppatori decidono se il verdetto è legittimo oppure sbagliato, per poi ricominciare il processo da capo. Per quanto si è risolto il problema di recuperare i dati con cui allenarsi, manca quella spinta in più per far riconoscere alla

macchina dati che un umano non riuscirebbe ad elaborare facilmente. Quando la macchina analizza una parte della clip di gioco, valuta un insieme di condizioni presenti in un database che si dividono in altre sotto condizioni: per esempio, se un giocatore spara ad un altro la macchina valuta con quale arma, se il colpo è andato a segno, dove è atterrato il colpo, se è un colpo alla testa oppure no. Tra le varie condizioni si aggiungono anche quelle umanamente complesse da calcolare, come gli aggiustamenti della visuale tra un frame e l'altro, il percorso della linea di tiro e quali sono i possibili elementi riscontrabili tra un tick e l'altro e tanti altri dati. Tutte queste condizioni vengono valutate per ogni entità di gioco per ogni finestra d'azione interessante presente nella clip, andando a creare una quantità di dati valutativi impressionante e che richiederebbe molto hardware per la gestione delle elaborazioni(circa 1650 processori). Quando il numero di conferme iniziano a calare, gli sviluppatori aumentano il numero di clip da far elaborare alla macchina.

Figura 3.3: Grafico che mostra i risultati del programma Overwatch da Settembre 2014 al Gennaio 2018. Il grigio sono il numero di clip totali, rosso e blu rappresentano rispettivamente le clip con giocatori giudicati colpevoli e in blu quelle scartate. La parte arancione segnala l'introduzione di VACNET al programma.

3.6 Livello di Intrusività di un Anticheat

Prima di iniziare a sviluppare un anticheat, una delle domande più importanti che gli sviluppatori fanno è: quanto ci possiamo fidare dei giocatori? Quanto

possiamo spingerci all'interno di un calcolatore per poter confermare le nostre supposizioni di uso di cheat? Quando si sviluppa un anticheat bisogna valutare il livello di intrusività del software. L'intrusività si intende come l'insieme di permessi operazioni e tecniche che un software richiede ed usa per poter funzionare all'interno di un compilatore. Principalmente il livello varia a seconda se il software funziona solo a livello utente oppure richiede di funzionare a livello kernel. In un compilatore, i programmi eseguiti a livello utente possiedono un numero limitatissimo di operazioni macchina e, nel caso vengono richieste il sistema operativo blocca le altre operazioni per poter risolvere la richiesta fatta dal programma. Un programma eseguito a livello kernel, invece, dispensa di tutte le operazioni offerte dalla macchina dove lavora; queste operazioni sono molto potenti poiché controllano l'intera infrastruttura del calcolatore. L'uso inconscio e scorretto delle funzioni presenti in modalità kernel possono portare a crash di sistema, errori irreversibili oppure il controllo totale della macchina da parte di altre persone. Per questo bisogna distinguere i due tipi di anticheat: quelli che operano a livello utente e quelli a livello kernel. Quando gli anticheat operano entra in gioco il suo livello di intrusività. Questo livello varia a seconda di quanto accessi diversi vengono consentiti al software per poter funzionare: il livello va' dal chiedere l'accesso di lettura di un singolo dispositivo, all'avere molteplici tipi di operazioni possibili per ogni dispositivo presente nel sistema. Un altro fattore che aiuta a variare il livello è l'attivazione del software: nella maggior parte dei casi, il software rimane dormiente fino a quando non viene attivato, per esempio quando viene fatto partire il gioco. In altri casi invece il software parte insieme al Boot di sistema, in modo da poter controllare se nell'inizializzazione vi è la presenza di cheat. Questo tipo di comportamento è simile a quello di una *rootkit*, tipo di malware per la quale viene installato un programma inizializzato al boot che permette all'attaccante di poter avere pieno controllo della macchina vittima. Per quanto possa essere comodo avere un intero quadro del funzionamento del computer per poter segnalare potenziali anomalie generate dall'uso di cheat, l'eticità della tecnica è grigia: siccome il suo funzionamento è simile all'avere una *rootkit*, molti utenti non vedono di buon occhio il poter "lasciare in mano" agli sviluppatori il controllo del proprio computer, anche se di solo controllo. Il rischio viene dalla possibilità che un cracker scopri il modo di poter sfruttare il software per poter accedere al computer dei molteplici utenti che hanno scaricato il software. Secondo A. Maario, V. K. Shukla, A. Ambikapathy e P. Sharma, uno dei problemi maggiori dell'anticheat a livello kernel riguarda le possibili vulnerabilità, come *false flags* che fanno bloccare driver vitali per il calcolatore, oppure *exploit* che permettono a gruppi di terze parti di poter implementare modifiche maligne all'interno del sistema.

A livello kernel, rispetto ai bug locali specifici dell'applicazione, qual-

> siasi vulnerabilità presente nel codice del drive potrebbe causare instabilità per tutto il sistema. Una grave svista del driver, tipo un buffer overflow, permetterebbe ad attaccanti maliziosi a installare programmi a livello esterno per la quale potrebbe rivelarsi pericoloso. (Anton Maario; Vinod Kumar Shukla; A. Ambikapathy; Purushottam Sharma, 2021)

Un esempio di anticheat presente a livello kernel Vanguard, presente nell'FPS Valorant della Riot Games. Esso è composto da due parti: la parte che funziona a livello utente attivabile all'avvio del gioco, e il driver a livello kernel attivato all'avvio del computer, in modo da poter controllare se non vi sono state modifiche del boot. Per via del suo funzionamento, esso è stato sotto l'onda di molte critiche da parte della comunità videoludica, portando alla luce i rischi di hijacking e/o di trafugazione di dati del computer attraverso l'analisi dei dispositivi hardware presenti. Secondo una delle filosofie della Riot Games:«Vanguard è una soluzione che ci aiuterà a raggiungere l'obbiettivo di totale competitività permettendoci di continuare ad adattare il nostro arsenale nella guerra contro i cheater.»(Team di sicurezza della Riot, Apr 17 2020)

Capitolo 4

Discutere delle soluzioni

4.1 Fare il quadro della situazione

Come è stato descritto, Team Fortress 2 è un FPS multiplayer a squadre per PC che sta subendo un'infestazione di algoritmi generati e spediti da utenti esterni che sfruttano cheat via software di terze parti, denominati bots. Il gioco usa un software speciale, chiamato anticheat, col compito di analizzare i dati dei giocatori e della partita per poter identificare l'uso di cheat e rimuovendo gli sfruttatori con conseguente sospensione d'accesso futuri al gioco. Tuttavia questo sistema non è in grado di poter stare al passo con la mole di bot generati e alle varie tecniche atte al mascheramento virtuale delle operazioni illecite. A differenza di un cheater umano, l'unico obbiettivo di coloro che controllano questi algoritmi è frustrare gli altri giocatori in modo da poter spingere i giocatori ad abbandonare la partita.

4.2 Le armi dell'attaccante

Il bot sfrutta meccanismi che si dividono per due tipi: di importuno e di difesa. I bot si dividono a seconda dell'autore che li ha generati: ogni bot proveniente dallo stesso bot hoster, possiede i stessi cheat e comportamenti condiviso con i compagni della stessa branca. Le diverse branche di bot vengono distinte dalle immagini di profilo che usano, dal nickname e dal tipo di comportamento elaborato. Quando si parla di comportamento si intende dei tipi di cheat e script usate dalle macchine a seconda del contesto.

Figura 4.1: Schema descrittivo di un bot con la classe cecchino.

4.2.1 Meccanismi di importuno

I meccanismi di importuno sono l'insieme di cheat o azioni che vengono usati per dar fastidio ai giocatori. Questi sono tipi di script/cheat usati:

Aimbot Cheat che permette di mirare contro un bersaglio e sparare in modo automatico: questo è lo script più comune dei bot. I bot lo sfruttano attraverso le classi che richiedo colpi ben precisi e potenti, come il cecchino. Vi sono diversi tipi di *aimbot*, come per esempio il TriggerBot, che preme il pulsante di fuoco automaticamente nel momento in cui il bersaglio si trova sotto il mirino. Oltre ad avere diversi tipi di aimbot è possibile modificare la posizione dei colpi e il rateo di fuoco tra un colpo e l'altro. Vi è un enorme lista di possibili tipi e variazioni di *aimbot*, come per esempio lo Spinbot, bot che permette di usare l'*aimbot* mentre si gira su se stessi guardando

in alto o in basso: questo serve per poter confondere i giocatori.

Micspam L'atto di disturbare i giocatori attraverso la chat vocale di gioco mediante l'uso di suoni molto forti e/o musica: molto comune, di solito viene messa musica o audio contenente linguaggio offensivo e/o scurrile. Il termine viene da Mic, abbreviazione per *microphone*, microfono in italiano, e il termine *spam* che in gergo definisce il continuo ripetere di un azione. A differenza degli altri tipi di attacco, il *micspam* è risolvibile mutando il giocatore che abusa della chat vocale, dato che Team Fortress lo permette.

Chatspam Concettualmente come il *micspam* ma con la chat testuale: consiste nell'ostruire la chat di testo con frasi inutili, offensivi oppure blocchi vuoti per scoraggiare gli altri giocatori alla comunicazione testuale. In passato era molto comune, ma dopo l'aggiornamento del 16 giugno del 2020 la chat testuale venne ristretta agli account premium, negandone l'uso ai giocatori non muniti. Il sistema nega ad alcuni bot di poter sfruttare la chat liberamente, costringendoli ad acquistare una copia premium del gioco.

Audiospam Simile al *micspam* con la differenza che si usano suoni built in nel gioco: comune e raro, nel senso che vi sono due modi per farlo. Il primo metodo richiede un particolare oggetto facilmente ottenibile che concede di eseguire suoni a comando, chiamato Rumorogeno. L'altro metodo richiede software di terze parti per poter usare suoni inaccessibili con mezzi normali e di farli sentire a tutti i giocatori.

SpeedHack Cheat che permette di modificare la velocità(*speed*) di movimento dei giocatori: non comune, principalmente usato da personaggi che possiedono armi da corta distanza come un fucile a pompa, dato che minore è la distanza dal bersaglio maggiori sono i danni subiti. Le classi che vengono usate sono tutti coloro che possiedono armi da corta gittata: Scout, Soldato, Piro, Ingegnere e Grosso.

CritHack In Team Fortress 2 quando si spara un'arma vi è una chance che i colpi diventino critici, ovvero compiono il doppio dei danni. Questo accade quando si riempie un "secchio dei colpi" che, una volta colmo, viene scaricato con dei colpi critici. I bot modificano i parametri del "secchio" affinché possano ottenere i critici con frequenza maggiore: raro, principalmente usato dai cheater umani. Le classi che solitamente usano questo sistema sono lo scout e raramente l'ingegnere, poichè sono veloci e dispongono del fucile a pompa che permette una rosata di proiettili molto ampia da poter colpire e uccidere i giocatori con pochi colpi.

Uso di *NavMesh* Per potersi muovere lungo la mappa automaticamente, i bot richiedono dei grafi di navigazione, chiamati *Navigation Meshes*. Questi grafi invisibili ai giocatori sono presenti nella maggior parte delle mappe, in modo da permettere ai bot d'allenamento di potersi muovere lungo la mappa di gioco. I bot baranti possono sfruttare i NavMesh ufficiali presenti nelle mappe oppure quelli personalizzati implementati dal bot hoster. Alcuni bot si muovono nella mappa di gioco grazie alla generazione automatica a run time di grafi di navigazione, rendendo non necessario l'input umano.

4.2.2 Meccanismi di difesa

Furto del Nickname Per evitare di venire cacciati dalla partita, i bot usano un sistema per poter rubare il Nickname e l'immagine profilo di un giocatore; questo serve per confondere i giocatori durante il voto di espulsione nel votare il giocatore sbagliato. Nel momento in cui la vittima lascia la partita, che sia per espulsione errata o no, il bot sceglie un nuovo giocatore da 'derubare'. In passato la pratica era molto comune, tuttavia dopo l'aggiornamento del 16 Settembre 2021 è diventato molto raro. Vi sono delle soluzioni: La prima è fare in modo che la vittima faccia partire il voto di espulsione contro l'impostore; la seconda consiste nell'aprire la console, digitare `status`, controllare il tempo di gioco dei due giocatori, prendere l'identificativo del bot e digitare `callvote kick (ID dell'utente)` per far partire il voto.

Voto d'espulsione Team Fortress 2 possiede un sistema di voto permettendo ai giocatori di decidere democraticamente di compiere modifiche al gioco o espellere giocatori: affinchè un voto venga stabilito come approvato, esso richiede che il 60% della squadra abbia votato a favore. Nel momento in cui un giocatore si unisce nella partita, oppure durante il corso del gioco, il bot inizia un voto di espulsione contro di esso. Questo, oltre a generare fastidio nella vittima, serve per poter far mantenere i bot in partita, rimuovendo giocatori che potenzialmente potrebbero espellere il bot. Questo sistema è altamente efficente in grandi gruppi di bot che collaborano per votare l'espulsione dei giocatori umani.

ServerBOoTing Un *TriggerBot* è uno script che attiva un azione automaticamente dopo che è avvenuta una condizione. Il *ServerBOoTing* è un tipo di *TriggerBot* che manda false perdite di connessione al server ai giocatori, costringendoli ad uscire e rientrare nella partita: raro. Di solito si attiva nel momento in cui il bot viene votato per l'espulsione, in modo che i giocatori non riescono in tempo a votare per la conferma causandone un

annullamento dell'espulsione e permettendo ai bot di rimanere nel gioco. Dopo che i giocatori lasciano la partita a causa della perdita della connessione, il server dopo il mini isolamento tenta di ristabilire la connessione con i giocatori persi, permettendo a questi di poterci rientrare, portandoli a ricominciare il circolo vizioso con conseguente rinuncia e abbandono totale della partita.

4.3 Le possibili soluzioni ed i loro problemi

Avendo quindi appreso come funziona Team Fortress 2, i cheat, l'anticheat e i bot, bisogna ora discutere delle possibili soluzioni per la situazione. Discutendo delle possibili soluzioni vi è Shounic, ricercatore che si impegna a studiare il funzionamento delle meccaniche di gioco di Team Fortress 2 analizzando il codice sorgente; questa sezione si baserà molto sulla sua tesi riguardo alle soluzioni pensate dalla comunità ed a i pro e contro che ne comportano.

4.3.1 Bannare tutti gli account Steam con lo stesso nickname associato ai bot

I bot provenienti dalla stessa branca vengono 'marchiati' con lo stesso nickname alcuni seguiti con un numero, ad esempio OMEGATRONIC(3) oppure Twilight Sparkle. L'idea è quella di bannare gli account di tutti quelli che hanno lo stesso nome associato ai bot. Facendo così, si rimuoverebbero temporaneamente tutti i bot presenti e anche futuri.

Problemi I bot possono trovare un modo diverso per potersi identificare senza dover usare il nickname, come ad esempio l'immagine di progilo oppure altri tipi di dato presenti nell'account di Steam, permettendo così di poter usare nomi generati a caso.

Tanti falsi positivi: dato che su Steam è possibile mettere qualsiasi nickname, è logico pensare che vi è una probabilità molto alta che chi possiede il nome Twilight Sparkle sia un giocatore normale. Dato che esiste una branca di bot con quel nickname, implementando questo sistema porterebbe al ban ingiustificato contro persone innocenti, portando il sistema ad essere obsoleto.

Esistono programmi che permettono di generare account di Steam: nel momento in cui un bot viene bannato, basterebbe usare il generatore di account per crearne un altro e tornare su Team Fortress gratuitamente, risolvendo nulla.

4.3.2 Bannare l'indirizzo IP dei bot

Ogni giocatore richiede, come per ogni servizio online, un indirizzo IP, i bot provenienti dallo stesso bot hoster, possiedono lo stesso IP. L'idea è quella di creare un ban all' indirizzo IP dell' hoster. In questo modo si riesce a bannarne l'accesso, senza che nessuno venga ingiustamente bloccato...

Problemi ...Tranne per tutti gli altri utenti che si trovano nella stessa rete locale a cui l'indirizzo IP è stato bannato. Il dispositivo contentente l'indirizzo IP bannato non permetterebbe di giocare a persone con account di Steam diversi.

Gli indirizzi IP non sono permanenti, dato che il router distribuisce gli indirizzi ad ogni dispositivo collegato. Basterebbe quindi farlo ripartire per generare nuovi indirizzi e quindi liberare l'hoster.

I bot possono usare una VPN: la VPN, ovvero la *Virtual Private Network* è una rete privata a pagamento o gratuita che sfrutta la rete pubblica per il trasporto di dati. La VPN, oltre a poter usare le reti pubbliche in modo sicuro, permette di collegarsi a server DNS differenti con indirizzi IP diversi; questo comporterebbe quindi un effetto mulinello dove i bot che vengono bannati via IP si rivolgono alla VPN per poter continuare a giocare usando una nuova IP, facendo ricominciare il ciclo del mulinello.

4.3.3 Bannare l'hardware dei bot

Simile con l'IP, la maggior parte dei bot originano dallo stesso calcolatore che li manda. L'idea consiste nell'identificare il bot hoster attraverso le informazioni ottenut da parte del VAC dei componenti hardware dell' attaccante e bannarlo: questo per fare in modo che il ban rimanga anche se si prova a cambiare l'account o l'indirizzo IP. Costa molto aggirare il ban, poiché costringe l'attacante ad acquistare dei nuovi componenti hardware.

Problemi Esistono programmi che permettono di poter cambiare gli identificativi e/oo l'indirizzo MAC per poter camuffare la scheda di rete, come per esempio Technitium MAC Address Changer.

I bot per poter fare le loro azioni richiedono solo un processore per poter comunicare col server della partita; quindi basterebbe un semplice Rasberry Pi, device che funziona esattamente come un computer con costi molto bassi(circa 35$).

Buona parte dei bot non stanno su macchine reali ma su molteplici macchine virtuali che emulano il funzionamento dei componenti hardware. Un

esempio è Hyper-V, la macchina virtuale introdotta da Microsoft nel 2017 con Windows 10. Come qualsiasi programma per la creazione di *enviroment* virtuali, Hyper-V permette la virtualizzazione hardware, aiutando così al camuffamento dei dispositivi usati dal calcolatore 'dietro' a quella ospite virtuale. Attraverso questo sistema diventa impossibile poter identificare l'hardware dell'attaccante. Hyper-V permette di avere più di un'istanza attiva, consentendo di avere virtualmente più calcolatori. Hyper-V infine permette di poter caricare sistemi operativi diversi da Windows.

4.3.4 Mettere giocatori invisibili

Dato il funzionamento del cheat, l'*aimbot* punta automaticamente e istantaneamente nelle *hitbox* dei bersagli. L'idea è quella di mettere giocatori invisibili oppure dentro ad aree inaccessibili, affinché si possano facilmente identificare e ostacolare coloro che fanno uso di *aimbot* e quindi la maggior parte dei bot, costringendoli a sparare in punti a caso evitando i giocatori.

Problemi Siccome non vi è alcuna distinzione vera e propria tra un cheater e un giocatore non è possibile mandare informazioni al giocatore senza che il cheater lo sappia; questo significa che nel momento in cui si crea un giocatore invisibile, basterebbe che il bot hoster scopra il modo con cui vengono resi invisibili e programmare gli algoritmi per ignorarli.

Molti bot hanno diverse politiche di scelta del bersaglio: invece di bloccarsi sui primi giocatori che si trovano a portata di tiro, alcuni sono programmati nel cambiare bersaglio ad ogni sparo, altri invece scorrono una lista di obiettivi e tanto altro. Implementado i giocatori invisibili, il problema continua a persistere.

4.3.5 Rimuovere le *NavMesh* ufficiali

Come spiegato prima, anche se Team Fortress 2 richiede giocatori veri per poter giocare ad una partita, sono stati inseriti grafi di navigazione lungo alcune mappe per aiutare i bot d'allenamento a muoversi efficientemente nelle partite di training. Siccome alcuni bot sfruttano questi grafi, l'idea è quella di rimuoverli.

Problemi Come è stato constatato prima, rimuoverli non risolverebbe il problema, poiché la maggior parte dei bot non usa i grafi di navigazione ufficiali.

Anche se venissero tolti, sia quelli ufficiali che quelli presenti online, essi sono molto facili da programmare nel bot, creando grafi completamente personalizzabili e alcuni anche a *runtime*.

4.3.6 Captcha

Il *Completely Automated Public Turing-test-to-tell Computers and Humans Apart*(CAPTCHA) è il sistema creato da Google per distinguere gli utenti umani che si connettono ad un servizio da un bot attraverso test audiovisivi e/o testuali non completabili da una macchina. L'idea è quella che per ogni giocatore che vuole accodarsi al *matchmaking* deve prima risolvere un test CAPTCHA per confermare che non sia un bot.

Problemi Esistono programmi che permettono di superare il CAPTCHA: ad esempio *Buster* è un'estensione browser per Google Chrome che permette di superare automaticamente i CAPTCHA con test audio. Alcuni CAPTCHA, per ragioni di accessibilità, usano test audio per coloro che non possono completare quelli visivi. *Buster* lo sfrutta riconoscendo i pattern dell'audio per poter risolvere il test, permettendo così alla macchina, oltre che all'umano, di superare il test.

Anche se si rimuovesse il test audio nel CAPTCHA, vi è 2Captcha: un sito che con 1 dollaro permette di far completare fino a 1000 CAPTCHA a dipendenti umani; secondo il sito 2Captcha nel 2022 vi sono circa 500 dipendenti nel risolvere i CAPTCHA classici e ben 1500 per risolvere quelli basati su JavaScript. Anche se si rimuovesse 2Captcha, esso non è l'unico ad offrire questo servizio, vi sono infatti tanti altri siti dello stesso tipo.

4.3.7 Autenticazione umana

Simile a come funzionerebbe con una banca, si potrebbero implementare sistemi per poter autenticare un giocatore attraverso qualcosa che possiede via *One Time Password*(OTP). Per quanto riguarda Team Fortress 2 vi sono 3 possibili varianti di autenticazione dell'account:

Steam Guard Steam Guard è il principale sistema di autenticazione degli account Steam sviluppato dalla Valve. Esso è un' applicazione per telefono che sviluppa una OTP ogni 10 secondi. Ogni volta che si entra nel proprio account di Steam lo Steam Guard, se attivato, richiede di usare l'OTP presente nell' applicazione come misura aggiuntiva di autenticazione. L'idea è quella di portare la richiesta di autenticazione via Steam Guard su Team Fortress per potersi accodare al *matchmaking*. Il sistema può aiutare, poiché richiederebbe comprare tanti telefoni diversi per poter funzionare.

Tuttavia esistono online diversi autenticatori ed emulatori di Steam Guard: questo per poter facilitare il sistema di acquisto e scambio degli oggetti cosmetici di Team Fortress 2. Un esempio è l'autenticatore open source

chiamato WinAuth: esso trasforma il computer in un dispositivo di autenticazione richiesto per Steam. Il bot hoster non dovrebbe fare altro che aprire tante istanze del programma per il numero totale di bot che possiede. Questo tipo di attività viene già implementato da alcuni siti esterni di mercato del gioco, come scrap.tf oppure STN.trading, dove vi sono un gran numero di account bot usati per facilitare le operazioni di scambio commerciale.

OTP col numero di telefono Sistema di autenticazione simile a come viene usato da tanti altri esercizi online e non, l'idea è quella di compiere l'autenticazione attraverso il numero di telefono.

Anche questo sistema non risolve completamente il problema per via dell'esistenza di siti come 5Sim. 5Sim vende numeri di telefono provenienti da quasi tutti gli stati del mondo e di riceverne i messaggi per pochi centesimi; 5Sim, a detta del sito è studiato con l'intento di bypassare le verifiche del numero di telefono. 5Sim inoltre permette di dare numeri di telefono specificamente per vari servizi online che lo richiedono, incluso Steam. Il costo per ottenere un numero di telefono varia per paese, ma si trova sotto 1 dollaro. Basterebbe quindi comprare un numero di telefono, scriverlo su Steam, aspettare l'OTP per poi inserirlo, bypassando il sistema.

Carta d'identità Altro sistema usato dai pubblici uffici, l'accodamento in *matchmaking* è possibile solo mostrando una carta d'identità che attesti di essere un giocatore umano.

Esistono siti che permettono di creare vere e proprie carte di identità, con bollo, linea visibile alla luce e codice a barre, come ID Creator. Anche se la creazione fisica richiederebbe sulle centinaia di dollari, virtualmente è gratuito.

Anche se non esistessero questi siti, se Valve decidesse di controllare tutte le carte d'identità di ogni giocatore, il sistema non può essere troppo sofisticato data il rischio di un' enorme problema di *bottleneck*. Basterebbe avere un identificativo abbastanza credibile da non farsi distinguere dalle altre migliaia di ID al minuto che il supporto di steam dovrebbe visionare.

4.3.8 Rimuovere il supporto di Linux

Team Fortress 2 supporta Windows, MAC OS e Linux. La principale applicazione usata per la gestione dei bot, ovvero cathook, funziona solo su Linux, rendendolo di conseguenza il nucleo dei bot. L'idea è quella di rimuovere completamente il supporto a Linux per il gioco.

Problemi Se un programma viene scritto per Linux, è possibile che lo si possa scrivere anche per gli altri sistemi operativi. Andando ad analizzare meglio la progettazione del programmi usati dai bot hoster essi richiedono 3 parti vitali: la parte che usa i cheat ed altre operazioni in partita, la parte che nasconde il bot dall'anticheat e la parte per l'accodamento al *matchmaking* e raggruppamento dei bot automatico. Le prime due parti possiedono già programmi ormai ben presenti e documentati sugli altri sistemi, per esempio vi è il famigerato programma di cheat LMAOBOX, funzionante sia su Windows che su MAC OS. Mancherebbe da implementare il sistema di raggruppamento automatico dei bot nel *matchmaking*. Siccome cathook è open source, non serve troppo impegno per replicarlo.

Valve si è sempre impegnato nella lotta per il mantenimento dell'open source nel mondo dell'informatica: secondo il fondatore della casa di sviluppo, Gabe Newell, data la natura open source di Linux esso permette di mantenere lo spazio informatico e videoludico più sano. Se si decidesse di creare un gioco che compete contro un manifatturiere di un sistema operativo, per esempio Microsoft con Windows, quest'ultimi possono scegliere di non renderlo supportabile nel loro OS. Per questo motivo la Valve continua a mantenere il supporto a Linux dandone vari contributi come ProtonDB, un software di importazione che permette di poter usufruire giochi supportati da Windows in Linux senza problemi. Tutti i server multiplayer della Valve usano Linux, compreso Team Fortress 2. Rimuovere il supporto a Linux andrebbe contro gli ideali della compagnia con conseguente spreco di risorse per lo sviluppo delle tecnologie compiute.

4.3.9 Implementare un *cooldown* per il *matchmaking*

Il funzionamento dell'idea è simile alla sospensione dell'inserimento del codice di sicurezza di un telefono: dopo 3 tentativi falliti di inserimento del PIN di sblocco, il telefono blocca la possibilità di riprovare il codice per 1 minuto; ogni 3 tentativi falliti viene attivato il blocco con un tempo incrementato: da 1 minuto si passa ai 3 minuti, 5 minuti, 10 minuti, fino ad arrivare ai 30 o 60 minuti ed oltre. L'idea è quella di applicare questo sistema di *cooldown* ogni volta che il giocatore subisce un espulsione: per ogni espulsione subita al giocatore viene attivato un *cooldown* per l'accodamento al *matchmaking*. Per ogni espulsione viene incrementato il tempo d'attesa: più voti il giocatore subisce, maggiore l'attesa. Il sistema di *cooldown* del *matchmaking* viene usato in altri videogiochi online, per esempio Counter Strike e la serie di Halo.

Problemi I bot non possiedono alcun limite temporale: a differenza degli umani, i bot non dormono; anche se subiscono l'ennesima espulsione che gli causa

un'attesa molto lunga, essi si riattiveranno istantaneamente nel momento in cui viene rimosso il blocco allo scadere del tempo.

L'ora di attività di punta dei bot è durante le ore del giorno in cui ci sono pochi giocatori, dalla mezzanotte alle 7 del mattino. Questo significa che se durante il giorno accumulano le attese con le espulsioni, di notte dominano tutti i server in cui si trovano. Raggruppandosi fra di loro nei server ufficiali possono compiere il voto di espulsione a tutti i giocatori veri che tentano di entrare nella partita, bloccandogli l'accesso al *matchmaking* per qualche minuto a causa del *cooldown*.

4.3.10 Creare un sistema esclusivo di *matchmaking*

Un'altra idea è quella di implementare il sistema di Prime *matchmaking* usato da Counter Strike: Global Offensive, costringendo i bot hoster a dover pagare l'accesso di circa 15 euro per ogni bot generato, invece di rimanere delegati nelle partite "gratuite".

Problemi L'idea si basa col presupposto che i bot hoster non dispendono di supporto economico. Se il bot hoster possiede abbastanza soldi da poter acquistare il Prime per ogni bot generato, allora il problema rimane.

Esistono siti o utenti che vendono il pass e/o account di Steam con il Prime incorporato a prezzi molto bassi. Attraverso questo sistema è possibile raggirare l'acquisto del Prime ufficialmente a 15 euro con quello dell'account Steam con il Prime a meno di 10 euro.

Se si implementasse un sistema per creare un *matchmaking* esclusivo a pagamento per un gioco gratuito con la promessa di risolvere un problema che è stato dimostrato che non lo risolve, toglierebbe la fiducia ai giocatori. Valve non vuole rischiare di compiere questa mossa perché comporterebbe una grave perdita di fiducia da parte dei consumatori e di guadagni per via del mercato di scambio e acquisto presente all'interno del gioco.

4.3.11 Implementare il Trust Factor

L'idea è quella di implementare il Trust Factor come quello di Counter Strike: Global Offensive. Il Trust Factor ha il compito di assegnare un valore ad ogni giocatore, valutato secondo i suoi comportamenti in gioco, dentro e, in parte, fuori da Steam; questo in modo che i giocatori con lo stesso valore vengano raggruppati nelle stesse partite. Implementado il sistema, i bot verebbero raggruppati tra di loro, in modo da non disturbare i giocatori onesti.

Problemi Il sistema predice e assume il comportamento dell'utente, quindi è possibile che il sistema possa inserire il giocatore nel "buco dei bari" perché il Trust gli ha dato un valore che lo identifica come un giocatore che bara assiduamente. Alcune persone attestano di essere stati messi contro cheaters poiché possiedono un comportamento tossico, senza però aver usato alcuna assistenza illegittima. Inoltre, è difficile vedere la differenza tra un giocatore ed un bot: se i bot smettessero di usare cheat e trovano un altro sistema per dare fastidio, il sistema di Trust non riuscirebbe ad identificarli e a categorizzarli correttamente nella creazione delle partite. Un esempio sono i bot presenti su Counter-Strike che rimangono in partita per generare esperienza necessaria per aumentare il livello; ad ogni livellamento il gioco rilascia un oggetto, tipicamente un'arma o una cassa, che viene prontamente venduta per del guadagno.

Valve, come con il VAC, non ha mai definito come funzioni il valore Trust, come vederne il proprio e come migliorarlo. Solo Valve è a conoscenza del suo funzionamento vero e proprio. Significa quindi che nessuno all'infuori di un dipendente della Valve, può alterare il funzionamento del sistema Trust. Questa è un'arma a doppio taglio poiché, come è stato definito, il sistema può aiutare come può danneggiare.

4.3.12 Implementare VACNET

L'idea è quella di implementare l'anticheat con *deep learning* sviluppato dalla Valve già presente su Counter Strike. Esso è molto più preciso rispetto al VAC normale per via del *deep learning*, riuscendo a distinguere il baro da un giocatore onesto studiandone i pattern visivi.

Problemi In Counter-Strike tutte le armi che vengono usate usano solo proiettili hit scan quindi non sono fisici. Team Fortress 2 possiede armi che sparano proiettili fisici, ovvero colpi che vengono creati nel mondo di gioco, hanno una *hitbox* e interagiscono con la geometria e le altre entità fisiche come i giocatori o gli oggetti dell'ambiente, essenzialmente tutte le entità con una *hitbox*. VACNET non è stato 'allenato' per distinguere i pattern dei proiettili fisici. Per poter identificare il moto di un proiettile fisico, come un razzo oppure una granata, servirebbero altri parametri per poter identificare la legittimità del colpo; il che significa dover generare tante altre clip di gioco da dare in pasto all'algoritmo, operazione che richiederebbe molto tempo e molta più potenza di calcolo.

Anche se il sistema VACNET funzionasse perfettamente, sarebbe comunque difficile distinguere i giocatori/bot che barano da quelli che sono molto

bravi. Il VACNET è *deep learning*, quindi richiede un minimo di input umano in modo da poter distinguere quando il giocatore è valutato barante e quando no. Vi sono giocatori veramente molto bravi per la quale l'uomo si confonde. Se l'uomo si può confondere, conseguentemente anche la macchina si può confondere. Il sistema quindi potrebbe generare dei falsi positivi, bannando persone che non stavano barando. Per non parlare anche dei possibili sviluppi futuri di *cheat engine* con strumenti molto più sofisticati che permettono camuffamenti ancora più efficenti, per la quale l'algoritmo non riuscirebbe a rilevare.

4.3.13 Implementare Overwatch

L'idea si basa sull'implementare il sistema Overwatch, già presente per Counter Strike. Il sistema permetterebbe ai valutatori comunità di visionare le clip delle partite dove vi sono potenziali bot e aggiudicarne un verdetto di conferma o non. Una volta che il bot subisce abastanza segnalazione esso viene bannato.

Problemi Il sistema non funziona come dovrebbe, dimostratosi dal grafico mostrato da John McDonald al GDC 2018 (pagina 27 3.3). Quindi se si scegliesse di implementare il sistema, esso non risolverebbe il problema.

Valve banna solo i giocatori per cui non ci sia alcuna ragione di dubbio che stiano barando. Le regole del *form* mettono scrupolosamente in chiaro che le segnalazioni del tipo di cheat devono essere fatti solo quando sono palesi. Ma come è stato definito prima, a volte differenziare un bot da un giocatore diventa difficile. Man mano che passano gli anni, la tecnologia migliora sempre di più affinché il bot mimichi più similmente il comportamento umano.

4.3.14 Aumentare il livello di aggressività e intrusione del VAC

Gli Anticheat possiedono dei livelli di intrusività. I livelli si distinguono a seconda di quante operazioni e accessi vengono consentite all'anticheat all'interno del proprio calcolatore. L'idea è quello di aumentare il livello di intrusività del VAC, seguendone l'esempio di Vanguard.

Problemi Come definito prima, i bot sono stabiliti su delle macchine virtuali e non su calcolatori fisici. Usando gli *environment* virtuali come Hyper-V diventa difficile decifrare il funzionamento del sistema operativo della macchina ospite. Questo significa che, nel mentre che il gioco gira sulla

macchina virtuale, nella macchina ospite attivo il software di cheat. In questo modo è possibile raggirare qualsiasi anticheat, anche quelli altamente intrusivi come quello di Valorant, dato che vedono il funzionamento del sistema operativo dove sta girando il gioco, ovvero la macchina virtuale.

Anche se si implementasse un intero sistema operativo per giocare ai giochi di Steam è comunque possibile trovare il modo di modificare il funzionamento dell'hardware esterno per poter barare, come tastiera e mouse. John McDonald al GDC 2018 nega l'idea implementativa di un sistema operativo personalizzato per via dell'alto livello di intrusività che ciò comporterebbe. Inoltre aggiunse che:

> Alla fine se l'utente ha accesso fisico al loro sistema non posso far nulla per confermare con certezza che non l'abbia modificato. [...] (I sviluppatori) possono chiedermi se il mio sistema compie questa operazione, quindi mi basterebbe fare un hijack della funzione e mentire.

(John McDonald, GDC 2018)

4.3.15 Denunciare i creatori dei Cheat

Alcune compagnie hanno denunciato con successo alcuni creatori di Cheat e hoster di bots per i loro giochi, per esempio la Blizzard con i cheater su Overwatch nel 2017 oppure Epic Games per Fortnite nel 2022. Secondo questa idea, Valve dovrebbe denunciare i creatori di cathook e degli altri cheat presenti per Team Fortress 2.

Problemi Le denunce sono limitate per certi tipi di giurisdizioni. Una buona fetta di creatori di questi tipi di software sono di origine russa e cinese, paesi per la quale il sistema giudiziario occidentale ha potere limitato, compreso quello americano della Valve.

Anche se riuscissero a denunciare e a eliminare i siti che dispensano i programmi per barare, altri ne verranno creati generando un effetto mulinello.

4.3.16 Giocare nei server della comunità

L'ultima idea è quella di giocare nei server della comunità. Questi server non sono ufficiali ma vengono creati e gestiti da utenti della comunità. Per accedervi non vi è il sistema di *matchmaking*, ma si opta per la ricerca usando un browser apposito che elenca tutti i possibili server presenti, distinguendone le varie variabili presenti nel gioco. Attraverso questi server è possibile implementare

diversi plugin e anticheat specifici che, oltre a modificare il gioco, permettono di bannare automaticamente i bot dalla partita. Inoltre è possibile avere dei moderatori per poter gestire al meglio sia gli utenti umani baranti che quelli automatici. Anche se questa soluzione non risolve il problema, essa è la migliore presente fra le altre.

Problemi Molti di questi server sono di bassa qualità e non riescono a gestire troppi giocatori, portando a rallentamenti nel gioco.

Alcuni di questi server possiedono abilità e vantaggi premium, che possono dare un vantaggio nel gioco a coloro che hanno donato gli autori, svantaggiando gli altri giocatori.

Manca un sistema di filtraggio più descrittivo. Per molti giocatori è difficile trovare il server con la modalità interessata, poiché ve ne sono tante e l'UI non aiuta molto a decifrare i parametri presenti nel gioco. Per esempio, vi sono server per cui vi è una sola mappa 24/7 dove ci sono i critici, ma un giocatore desidera la stessa mappa 24/7 senza i critici.

Per un giocatore nuovo, il *matchmaking* pubblico è molto comodo per poter capire come funziona il gioco, poiché viene accodato con persone della sua stessa abilità. I server della comunità, oltre ad essere difficile per un neofita capire in quale andare, non sono dei buon sistemi per introdurre il gioco per la prima volta, dato che sono frequentati da persone che possiedono vari livelli di esperienza che potrebbero rovinare quella dei principianti che hanno appena iniziato a giocare.

Soluzione Secondo Shounic, una soluzione è quella di creare e curare un sistema di Pagine Gialle dei server. Questo sistema dovrebbe possedere un'opzione di filtro per le modalità, delle variabili e delle mappe, affinché i giocatori possano scegliere il proprio server che rispetti le proprie preferenze. Infine, Shounic propone di creare server per le persone nuove dove poter imparare le meccaniche di gioco, bannando tutte le persone che possiedono una certa quantità di esperienza di gioco.

4.4 Cosa sta facendo Valve

Da quando Team Fortress 2 ha subito la prima invasione, vi sono stati dei lenti sviluppi per il contrasto dei bot. Una delle "mosse" più controverse implementate avvenne nel 16 giugno 2020, rimuovendo la possibilità di scrivere nella chat testuale per i giocatori non paganti. Lentamente il team dietro a Team Fortress ha ostacolato i bot e migliorato il codice sorgente del gioco, azioni che,

Figura 4.2: Il browser dei server della comunità. Per ogni server viene indicato con delle icone se è privato, usa VAC e permette i replay di gioco[0]; i server vengono inoltre descritti dal nome[1], tipo di modalità[2], numero di giocatori presenti/limite massimo[3], numero di giocatori controllati dal gioco[4], nome della mappa[5], latenza del server[6] e da tag che descrivono minimamente le variabili e plugin presenti [7]

tuttavia, non bastano nel grande schema del problema ancor presente tutt'oggi. Per quanto i progetti sviluppati per Counter Strike per il contrasto del cheat siano ormai di pubblico dominio, Valve è da sempre rimasto molto riservato sui dettagli e sulle tecnologie in costruzione. Ufficialmente la Valve non ha mai definito quanti sviluppatori stiano dietro la gestione della crisi dei bot: voci di corridoio dicono che ve ne siano 4, di cui 2 lo gestiscono come progetto secondario. Alcuni dicono che ve ne siano 5-6, tutti sviluppatori provenienti da altri progetti della casa di sviluppo e spinti a trovare soluzioni importanti per debellare la minaccia. Tutte speculazioni che, purtroppo, non fanno luce sulle vere intenzioni che la Valve desidera implementare, mantenendo il gioco in uno status quo per niente piacevole. Recentemente la Valve sta implementando ban basato sul nickname per le branche di bot più famosi, ad esempio Does Hotter e OMEGATRONIC, soluzione raggirabile facendo modificare il nome dei bot o comunque generando nomi casuali e distinti.

Capitolo 5

Conclusione

Sebbene gli sviluppatori si impegnano per poter mantenere il fair play nel videogioco online, quando si tratta di combattere contro degli algoritmi baranti la situazione diventa grigia. Come è stato visto, per quanto esistano sistemi che aiutano al contrasto dei bot in un videogioco online, è inevitabile che vi siano diversi modi per poter superare queste difese. Le domande più importanti che un ingegnere di sicurezza informatica si chiede sempre è: Basterà? Quanto mi posso fidarmi dell'utente? Quanto posso spingermi per assicurarmi che il peggio non possa accadere? Domande che, come abbiamo visto nell'elaborato, vengono discusse attraverso l'implementazione degli anticheat con i vari livelli di intrusione, della gestione degli input e output da parte del server di gioco e da tanti altri elementi. Come per ogni software, è impossibile creare il sistema di sicurezza perfetto: su 16 soluzioni solo 1 è considerabile relativamente ottimale, e si basa sul 'non vedere il problema' nascondendosi nei server della comunità, mandtenendo a piede libero i bot in quelli ufficiali. Anche se si trovasse la soluzione finale al problema, rimane la necessità di mantenerlo costantemente aggiornato, dato che anche i problemi si evolvono. Per quanto la situazione possa sembrare senza speranza e pessimistica, bisogna vedere il bicchiere mezzo pieno: per quanto le tecnologie implementate per Counter Strike dimostrano alcune pecche, esse rimangono promettenti nel mondo della sicurezza informatica per i videogiochi online, specialmente attraverso lo sviluppo e implementazione dell'anticheat con *deep learning* VACNET. Una perpetua sfida tra i due gruppi: gli sviluppatori lottano per contrastare i bari e mantenere il gioco il più onesto possibile, gli attaccanti lottano per sviluppare tecnologie in modo da superare le difese dei difensori e, alcuni, per il proprio divertimento. Una perpetua lotta che esiste fin dai tempi dei primi calcolatori.

Bibliografia

[1] *2Captcha*. URL: https://2captcha.com/.

[2] Yoshua Bengio, Aaron Courville e Pascal Vincent. «Representation Learning: A Review and New Perspectives». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8 (2013), pp. 1798–1828. DOI: 10.1109/TPAMI.2013.50.

[3] Mia Consalvo e Irene Serrano Vazquez. «Cheating, Social Network Games and the Role of Platforms». In: *2014 47th Hawaii International Conference on System Sciences*. 2014, pp. 1687–1694. DOI: 10.1109/HICSS.2014.216.

[4] Leo Kelion. *Overwatch 'cheat-maker' told to pay $8.6m to Blizzard*. BBC. 2017. URL: https://www.bbc.com/news/technology-39490317.

[5] Tim Stryker Ken Wasserman. «Multimachine Games». Inglese. In: *Byte* 5.15 (1980), pp. 24–40.

[6] Victoria Kennedy. *Fortnite cheater forced to pay up after Epic lawsuit*. Eurogamer. 2022. URL: https://www.eurogamer.net/fortnite-cheater-forced-to-pay-up-after-epic-lawsuit-damages-will-go-to-charity.

[7] Samuli Johannes Lehtonen. «Comparative Study of Anti-cheat Methods in Video Games». Tesi di dott. 2020.

[8] *LMAOBOX project*. URL: https://lmaobox.net/.

[9] LuckyStar. *TF2 What is REALLY Going On With Bots: I Asked Developers*. Youtube. 2020. URL: https://youtu.be/M7aynH_ZwWo?t=170.

[10] Anton Maario et al. «Redefining the Risks of Kernel-Level Anti-Cheat in Online Gaming». In: *2021 8th International Conference on Signal Processing and Integrated Networks (SPIN)*. 2021, pp. 676–680. DOI: 10.1109/SPIN52536.2021.9566108.

[11] John McDonald. *Robocalypse Now: Using Deep Learning to Combat Cheating in Counter-Strike: Global Offensive*. Youtube. 2020. URL: https://www.youtube.com/watch?v=kTiP0zKF9bc&ab_channel=GDC.

[12] Gabe Newell. *LinuxCon & CloudOpen North America 2013 - Linux & Gaming*. Youtube. 2013. URL: https://www.youtube.com/watch?v=rCGMiT0CQAI.

[13] Bao Nguyen. «Developing a Game Debugger with Unreal Engine 4». Tesi di dott. 2021.

[14] Nullworks. *catbot*. Github. 2019. URL: https://github.com/nullworks/catbot-setup.

[15] Nullworks. *cathook**. Github. 2021. URL: https://github.com/nullworks/cathook.

[16] Jeremy Peel. *Gabe takes to Reddit to clear up Valve Anti-Cheat rumours; "Do we send your browsing history to Valve? No."**. PCGamesn. 2015. URL: https://www.pcgamesn.com/counterstrike/gabe-takes-reddit-clear-valve-anti-cheat-rumours-do-we-send-your-browsing-history-valve-no.

[17] *Perfect Aim*. URL: https://perfectaim.io/.

[18] Shounic. *saveTF2 and why solving TF2's bot problem isn't that simple: a review of the current circumstances*. Youtube. 2022. URL: https://www.youtube.com/watch?v=SgkgsgaBBCA.

[19] *SkyCheats*. URL: https://www.skycheats.com/.

[20] Valve Software. *Assistenza di Steam*. Steam. URL: https://help.steampowered.com.

[21] Valve Software. *Valve Developer Community*. Inglese. URL: https://developer.valvesoftware.com/wik.

[22] *SteamCharts*. URL: https://steamcharts.com/.

[23] The Riot Security Team. *A Message About Vanguard From Our Security and Privacy Teams*, RiotGames. 2020. URL: https://www.riotgames.com/en/news/a-message-about-vanguard-from-our-security-privacy-teams.

[24] *Team Fortress 2*. Valve Corporation. URL: https://www.teamfortress.com/.

[25] *Team Fortress 2 Wiki*. URL: https://wiki.teamfortress.com/wiki/.

[26] Toofty. *The Truth Behind the TF2 Bot Crisis*. Youtube. 2020. URL: https://youtu.be/qyI5u-SQUYc?t=140.