GYMNASIUM JANA KEPLERA

Parléřova 2/118, 169 00 Praha 6

Cheat do hry OpenArena

Maturitní práce

Autor: Kryštof Kus

Třída: 4.A

Školní rok: 2024/2025

Předmět: Informatika

Vedoucí práce: Igor Vujovič

Praha, 2025

Gymnázium Jana Keplera

Kabinet informatiky

ZADÁNÍ MATURITNÍ PRÁCE

- *Student:* Kryštof Kus
- *Třída:* 4.A
- *Školní rok:* 2024/2025
- *Vedoucí práce:* Igor Vujovič
- *Název práce:* Cheat do hry OpenArena
- URL repozitáře: https://github.com/Krys-Kus/maturita

Pokyny pro vypracování:

Cílem této práce je vyvinout vlastní verzi několika běžných funkcí cheatů pro videohru OpenArena. Zejména aimbot, wallhack a triggerbot za využití nástroje Cheat Engine na sbírání dat z paměti.

Prohlášení

Prohlašuji, že jsem svou práci vypracoval samostatně a použil jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů. Nemám žádné námitky proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších předpisů.

V Praze dne 3. března 2025 Kryštof Kus

Poděkování

Děkuji vedoucímu práce Igoru Vujovičovi za investici svého času do konzultací mého projektu a za jeho trpělivost.

Abstrakt

Tato práce se zabývá vývojem a implementací cheatů pro hru OpenArena. Cílem bylo vytvořit funkční aimbot, wallhack a triggerbot, přičemž byl zkoumán jak externí, tak interní přístup k jejich realizaci. Nejprve byla vyvinuta externí verze aimbotu a triggerbotu, ale při pokusu o implementaci externího wallhacku se ukázaly zásadní technické překážky. To vedlo k rozhodnutí přejít na interní řešení, které bylo realizováno injektováním vlastního kódu do běžícího procesu hry. Práce popisuje principy injektování a hookování funkcí. Během analýzy binárního kódu OpenAreny byla použita reverzní inženýrská technika k identifikaci relevantních paměťových adres a funkcí. Výsledkem je plně funkční interní cheat, který umožňuje automatické zaměřování, detekci nepřátel skrze stěny a automatickou střelbu. Práce se také věnuje problémům, které během vývoje nastaly, a hodnotí možnosti dalšího vylepšení.

Klíčová slova

Reverzní inženýrství, DLL injekce, Hookování funkcí, Interní cheaty, Externí cheaty

Abstract

This work focuses on the development and implementation of cheats for the game OpenArena. The goal was to create a functional aimbot, wallhack, and triggerbot, exploring both external and internal approaches to their implementation. Initially, an external version of the aimbot and triggerbot was developed, but significant technical obstacles emerged when attempting to implement an external wallhack. This led to the decision to switch to an internal solution, achieved by injecting custom code into the game's running process. The work describes the principles of code injection, hooking, and function manipulation. During the analysis of OpenArena's binary code, reverse engineering techniques were used to identify relevant memory addresses and functions. The result is a fully functional internal cheat that enables automatic aiming, enemy detection through walls, and automatic shooting. The work also discusses the challenges encountered during development and evaluates possibilities for further improvement.

Keywords

Reverse engineering, DLL injection, Function hooking, Internal cheats, External cheats

Obsah

1. Teoretická část

1.1 Úvod

Tato dokumentace se zabývá problematikou vývoje herních cheatů, konkrétně v kontextu hry OpenArena. Cílem této práce bylo analyzovat a implementovat různé techniky používané při tvorbě cheatů, přesněji byl jako cíl této práce byl stanoven vývoj cheatů známých jako aimbot, wallhack a triggerbot, a tím zjistit jak technicky a časově náročný může proces vývoje cheatů být. Projekt byl realizován výhradně pro studijní a výzkumné účely, přičemž veškeré testování probíhalo na lokálním serveru pouze proti počítačem řízeným protivníkům (botům).

Hra OpenArena byla zvolena z následujících důvodů: Jedná se o open-source projekt, což umožňuje detailní analýzu zdrojového kódu a usnadňuje implementaci různých modifikací. Neobsahuje žádný anti-cheat systém, což eliminuje nutnost obcházení bezpečnostních opatření a umožňuje soustředit se na samotnou mechaniku cheatů. Umožňuje lokální testování bez zásahu do online herního prostředí, čímž je zajištěna etická a právní nezávadnost výzkumu.

K práci byl využit nástroj Cheat Engine, který umožňuje analýzu a úpravu hodnot uložených v paměti běžících procesů. Tento software byl zvolen z několika důvodů. Jednak jsem s ním měl předchozí zkušenosti, což zjednodušilo proces implementace jednotlivých technik. Dále je dostupnost informací o tom, jak Cheat Engine funguje a jaké funkce nabízí, poměrně vysoká, což umožňuje snadné pochopení jeho principů. V neposlední řadě se jedná o intuitivní a relativně snadno použitelný nástroj, který je široce využíván v oblasti herního hackingu a reverzního inženýrství.

Obecně lze říci, že obtížnost vývoje cheatů závisí na několika faktorech, mezi které patří zejména přítomnost anti-cheat systémů, složitost herního enginu a míra ochrany proti reverznímu inženýrství. Vývoj cheatů pro hry bez ochranných mechanismů a s dostupným zdrojovým kódem by neměl být tolik náročný ve srovnání s moderními tituly. U moderních online titulů s pokročilými bezpečnostními opatřeními může tento proces trvat řádově měsíce až roky, přičemž často vyžaduje pokročilé znalosti reverzního inženýrství, dynamické analýzy a obfuskace kódu.

Tato práce se zaměřuje na samotný proces tvorby cheatů a jeho technické aspekty. Výsledky projektu přispívají k pochopení metod využívaných při modifikaci herního softwaru a poskytují přehled o možnostech manipulace s herní logikou a mechanikami.

1.2 Cheat Engine

V rámci tohoto projektu byl nástroj Cheat Engine využíván k analýze a manipulaci s hodnotami uloženými v paměti hry OpenArena. Videohry ukládají různé herní informace, jako je zdraví, munice nebo pozice hráče, na specifické paměťové adresy. Paměťová adresa je unikátní identifikátor určitého místa v operační paměti, kde je uchovávána konkrétní hodnota. Tyto adresy však nejsou konstantní – často se mění s každým novým spuštěním hry, což komplikuje jejich přímou úpravu. Cheat Engine umožňuje skenovat paměť hry a sledovat změny hodnot na těchto adresách v závislosti na zvoleném vstupu. Pomocí opakovaného filtrování lze identifikovat konkrétní adresu odpovídající hledané herní hodnotě, například počtu životů nebo munice. Jelikož se adresy při každém

spuštění hry mění, Cheat Engine poskytuje nástroj nazvaný pointer scan, který umožňuje vytvářet pointermapy. Tyto mapy generují cestu k statické adrese, která se nemění a lze ji tedy spolehlivě využít pro další úpravy. Po nalezení statické adresy lze v Cheat Engine hodnoty na této adrese zmrazit (zabránit jejich změně hrou) nebo manuálně upravit, čímž lze dosáhnout požadovaných efektů, jako je nekonečné zdraví nebo munice. Tento proces byl v projektu využíván k nalezení adres, na kterých hra ukládá do paměti hodnoty nutné k vytvoření mnou zadaných cheatů. *(Andriesse, 2018)*

1.3 Teorie za jednotlivými cheaty

Aimbot je cheat, jehož cílem je automaticky zaměřovat nepřátele a usnadnit tak hráči přesnou střelbu. Funguje na principu vyhledání nejbližšího nepřítele nacházejícího se v zorném poli hráče a následného přepsání hodnoty úhlu pohledu v paměti hry dříve, než je tato hodnota odeslána serveru. Tím je zajištěno, že mířidla vždy směřují přesně na protivníka. Aby byl aimbot dostatečně efektivní, musí být schopen dynamicky vyhodnocovat situaci a plynule přeskakovat mezi cíli v reálném čase. Správná implementace zahrnuje práci s paměťovými adresami určující úhel pohledu hráče, čímž lze dosáhnout přesného a plynulého pohybu zaměřovače. *(Erickson, 2008)*

Zatímco aimbot zajišťuje přesné zaměření na cíl, triggerbot představuje jednodušší, avšak účinný nástroj, který automatizuje samotnou střelbu. Jeho hlavní funkcí je detekovat okamžik, kdy je aimbot správně zaměřen na nepřítele, a bez prodlení spustit střelbu. Tento proces probíhá bez nutnosti manuálního zásahu hráče, čímž se eliminuje jakákoli prodleva mezi zaměřením a výstřelem. Implementace může být realizována buď prostřednictvím analýzy barevných pixelů na obrazovce, nebo přímým čtením paměťových hodnot hry, které určují, zda je cíl v mířidlech.

Kromě zaměřování a automatizované střelby existují i metody umožňující získat výhodu prostřednictvím lepšího přehledu o situaci na bojišti. Wallhack umožňuje hráči vidět všechny nepřátele bez ohledu na překážky, čímž eliminuje omezení způsobená herním prostředím. Principem jeho fungování je neustálé sledování pozice všech protivníků a přepočítávání jejich trojrozměrných souřadnic na dvourozměrné souřadnice obrazovky. Jakmile jsou tyto souřadnice získány, lze kolem nepřátel vykreslit objekty, například obdélníky, které hráči umožní jejich nepřetržitou viditelnost. *(Erickson, 2008)*

1.4 Interní a externí cheaty

Při vývoji cheatů lze k jejich implementaci přistupovat dvěma základními způsoby – jako interní nebo externí cheaty. Tyto přístupy se zásadně liší nejen svou technickou realizací, ale i tím, jakým způsobem interagují s herním procesem. Každý z nich má své výhody i nevýhody, které ovlivňují jejich efektivitu, možnosti detekce a celkovou komplexitu vývoje.

Interní cheaty fungují přímo uvnitř herního procesu. Jsou obvykle realizovány formou injekce vlastního kódu do běžící hry, což umožňuje přímou manipulaci s herní logikou a pamětí. Díky tomu lze měnit hodnoty v reálném čase, upravovat mechaniky hry a dokonce rozšiřovat její funkcionalitu. Tento přístup nabízí vysokou flexibilitu a výkon, protože kód je vykonáván přímo v rámci hry, což umožňuje rychlé a efektivní operace. Hlavní nevýhodou interních cheatů je jejich vysoká detekovatelnost – vzhledem k tomu, že pracují uvnitř procesu hry, jsou často jednoduchým cílem anti-cheat systémů, které monitorují změny v paměti nebo načítání neautorizovaných modulů.

Oproti tomu externí cheaty pracují mimo samotný herní proces a nevyžadují přímou injekci kódu do hry. Obvykle fungují tak, že čtou a zapisují data do paměti hry z jiného programu, což znamená, že nejsou tak snadno detekovatelné běžnými metodami kontroly integrity. Nejčastější implementace externích cheatů zahrnují čtení paměti pro získání informací o herním stavu (například pozic nepřátel pro wallhack) a simulaci uživatelských vstupů (například pohybu myši pro aimbot nebo automatického stisku spouště u triggerbotu).

Velkou výhodou externích cheatů je, že jejich implementace obvykle nevyžaduje pokročilé znalosti reverzního inženýrství či programování na úrovni assembleru. Často stačí využít dostupné nástroje, jako je Cheat Engine, nebo pracovat s existujícími knihovnami pro čtení a zápis do paměti. Díky tomu jsou externí cheaty snadněji přístupné i pro méně zkušené vývojáře. Nevýhodou je jejich omezená rychlost a přesnost, jelikož nepracují přímo v rámci herního enginu a jsou závislé na rychlosti čtení a zápisu do paměti. Z hlediska detekce bývají externí cheaty méně nápadné, protože nemodifikují herní soubory ani nevstupují přímo do procesu hry. Výběr mezi těmito dvěma přístupy závisí na konkrétní hře, přítomnosti anti-cheat ochrany a požadavcích na funkčnost a bezpečnost cheatu. *(Andriesse, 2018)*

1.5 Injektování a hookování

Aby bylo možné přímo manipulovat s herním kódem a měnit jeho chování, je nutné do něj vložit vlastní instrukce. K tomu slouží injektování a hookování, dvě klíčové techniky používané při vývoji interních cheatů.

Injektor v kontextu Windowsu a tvorby cheatů funguje tak, že donutí proces načíst DLL soubor, který není jeho součástí. Nejprve je nutné získat handle okna cílového procesu a poté ID tohoto procesu, což umožní přístup k samotnému procesu skrze jeho handle. Klíčovou funkcí pro načtení našeho DLL je LoadLibraryA, která přijímá jako parametr cestu k souboru ve formě char*. Adresa funkce LoadLibraryA se získá z knihovny kernel32.dll, ke které se přistupuje pomocí GetModuleHandleA. Poté se s využitím GetProcAddress získá přesná adresa této funkce. Aby bylo možné předat DLL procesu, je uvnitř něj alokovaná paměť, kam se zapíše cesta k souboru. Jakmile je vše připraveno, funkce CreateRemoteThread donutí proces spustit LoadLibraryA s parametrem obsahujícím cestu k DLL, čímž se zajistí jeho načtení a spuštění cheatů. *(Sikorsky, Honig, 2012)*

Hookování umožňuje zasahovat do běhu hry a upravovat její chování v reálném čase. V kontextu operačního systému Windows slouží k přesměrování běhu určité funkce v cílovém procesu do našeho vlastního kódu. Díky tomu je možné manipulovat s daty, která měla funkce k dispozici, nebo je zcela přepsat podle potřeby. Z důvodu zachování funkcionalita hry, je nutné zajistit správné vykonání všech instrukcí, které byly přepsány při vkládání vlastního skoku. To znamená, že po provedení našeho kódu musí dojít k návratu na správné místo, aby se předešlo nekonečné smyčce při vykonání stejných instrukcí pořád dokola. Z toho důvodu je zapotřebí vypočítat adresu na kterou se vrátit jako adresu hooku, která je posunutá o počet přepsaných bajtů. Klíčovým krokem je správné uložení a následné obnovení stavu registrů, aby nedošlo k jejich nechtěné modifikaci. Při určování správného místa pro hookování je často využíván nástroj Cheat Engine, který umožňuje analyzovat a disassemblovat spustitelný kód hry. Minimální počet přepsaných bajtů při hookování je pět, ale protože se skok musí provést nad celými instrukcemi, může být tento počet vyšší. Existuje několik typů hooků – nejčastěji se umisťují na začátek funkce, ale běžně se také používají hooky uprostřed nebo na konci funkce, v závislosti na požadovaném efektu. *(UnKnoWnCheaTs, 2008, Sikorsky, Honig,*

2012)

2. Implementace

2.1 Implementace externího aimbotu a triggerbotu

Implementace externího aimbotu a triggerbotu začala volbou externího přístupu, jelikož jedním z cílů této práce bylo zjistit časovou a technickou náročnost vývoje cheatů. Externí metoda byla vhodnou volbou, protože se zdála být variantou snazší na implementaci.

Prvním krokem bylo nalezení paměťové adresy, která ukládá hodnotu týmu každého hráče – číslo 1 pro červený tým a číslo 2 pro modrý tým. Tato adresa se ukázala jako klíčová, protože její blízké okolí v paměti obsahovalo strukturu s informacemi o hráči a všech ostatních účastnících hry. Při dalším zkoumání se ukázalo, že změny ve hře, například pohyb hráče po mapě nebo otáčení kamerou, způsobují dynamické úpravy hodnot v určitých paměťových adresách. Tímto způsobem bylo možné identifikovat adresy, kde jsou uloženy souřadnice hráče a úhly kamery. Jakmile byly tyto adresy nalezeny, bylo možné určit vzorec v paměti – offset mezi hráči zůstal konzistentní, což umožnilo iteraci přes všechny postavy ve hře. Samotný aimbot tak pracuje s daty z paměti tím, že nejprve načte tým hráče, jeho souřadnice a následně projde maximálně 15 dalších hráčských struktur (protože výchozí maximální počet hráčů ve hře je 16). Mezi hráči je od adresy týmu ke každé další potřebné informaci – jako jsou souřadnice či úhel pohledu – vždy stejný offset, což umožňuje efektivní přístup k těmto hodnotám. Aimbot následně vypočítá směrový vektor mezi pozicí hráče a jeho protivníky a vybere toho, který je nejblíže na základě délky vektoru.

Další důležitou informací byly úhly pohledu – pitch (vertikální natočení) a yaw (horizontální natočení). Hodnoty těchto úhlů odpovídaly směru, kam se hráč dívá, ale na rozdíl od souřadnic hráče nebylo možné je přímo přepsat nebo zmrazit, což naznačovalo, že nemusí být skutečnými adresami používanými hrou k řízení pohybu kamery. Přesto bylo možné jejich hodnoty číst a na jejich základě spočítat korekci pohledu. Aby aimbot fungoval správně, bylo nutné převést směrový vektor mezi hráčem a nejbližším nepřítelem na úhly pitch a yaw, které odpovídají pohledu přímo na cíl. Odečtením těchto hodnot od aktuálního úhlu pohledu hráče se vypočítala delta – rozdíl mezi stávající orientací a požadovaným směrem. Tato delta sloužila jako vstup do funkce simulující pohyb myši, která automaticky posouvala zaměřovač hráče tak, aby vždy směřoval na nejbližšího protivníka.

Během prvního testování aimbotu jsem narazil na problém, kdy se kamera začala nekontrolovatelně otáčet. Tento problém byl způsoben nesprávně vypočítanými hodnotami pohybu myši, které vedly k nadměrnému posunu pohledu hráče. Řešením bylo clampování vstupu pohybu myši, což znamená omezení maximální vzdálenosti, o kterou se myš může v jednom kroku posunout. Důvodem pro toto omezení je, že výpočet cílového úhlu je někdy nepřesný a pokud by se pohled hráče v jediném kroku posunul příliš výrazně, mohl by aimbot snadno „přestřelit“ cíl a způsobit nepředvídatelné chování. Díky tomuto opatření se pohyb pohledu stává stabilnějším. I přes zavedené omezení se však stále občas vyskytoval problém, kdy se kamera protáčela. Tento jev byl způsoben chybným výpočtem delty úhlu yaw, tedy rozdílu mezi aktuálním úhlem pohledu hráče a požadovaným směrem k nepříteli. Problém vznikl při přechodu mezi kladnými a zápornými hodnotami úhlu yaw. Například pokud měl hráč yaw 150° a měl se otočit na -150°, správná hodnota změny yaw by měla být +60°, avšak nesprávným výpočtem mohla být určena jako -300°, což vedlo k prudké rotaci kamery. Tento problém byl způsoben tím, že hodnoty úhlu yaw existují v rozsahu -180° až 180°. Pokud aimbot vypočítal změnu yaw větší než 180° nebo menší než -180°, znamenalo to, že se neotáčí po nejkratší možné dráze, ale místo toho provádí zbytečně dlouhou rotaci přes celý

kruh. Řešením bylo přizpůsobit výpočet tak, aby aimbot vždy volil nejkratší cestu, tedy pokud byl vypočtený rozdíl yaw větší než 180° nebo menší než -180°, místo něj se použil doplněk do celého kruhu. Tím bylo zajištěno, že aimbot provádí otočení vždy kratší cestou a nikdy se neotáčí o více než půl kruhu v opačném směru.

Triggerbot byl implementován tak, aby reagoval na aktivitu aimbotu. Místo neustálé detekce nepřátel jednoduše kontroloval, zda je spuštěna funkce pohybující myší, tedy zda aimbot aktivně zaměřuje cíl. Pokud ano, triggerbot automaticky simuloval stisk tlačítka myši a zahájil střelbu. Za účelem vyhnutí se neefektivnímu plýtvání municí, byl přidán spínač, který umožňoval triggerbot ručně zapnout nebo vypnout podle potřeby. Tím měl hráč možnost rozhodnout, kdy bude automatická střelba aktivní.

2.2 Implementace externího wallhacku

Dalším krokem v projektu byla snaha o implementaci externího wallhacku. Vzhledem k tomu, že jsem již měl funkční externí aimbot, předpokládal jsem, že obdobným způsobem bude možné realizovat i wallhack. Základní princip byl jasný – využít již existující funkci pro vyhodnocení nepřátel a čtení jejich aktuální pozice a následně vykreslit okolo jejich souřadnic obdélník, jehož velikost se bude dynamicky měnit v závislosti na vzdálenosti od hráče.

Plánovaný postup spočíval v převodu trojrozměrných souřadnic nepřátel na dvourozměrné souřadnice odpovídající jejich poloze na obrazovce. Tyto souřadnice měly tvořit střed obdélníku, který by se zobrazoval v překryvném transparentním okně a neustále se aktualizoval podle pohybu nepřátel ve hře. Předpokládal jsem, že hlavní výzvou bude optimalizace výkonu, případně eliminace chyb způsobených opožděním při čtení paměti, nicméně velmi brzy jsem narazil na zásadní problém – samotné vytváření transparentních oken nebylo tak jednoduché, jak jsem očekával.

Vytvoření okna s vykresleným obrazcem fungovalo bez problémů. Nicméně zajistit, aby toto okno vždy překrývalo herní obrazovku se ukázalo jako značně nespolehlivé. V některých případech okno fungovalo správně, ale při dalším spuštění zůstávalo skryté nebo bylo překryto jinými aplikacemi. Největším problémem se však ukázalo dosažení průhlednosti – jakmile jsem se pokusil nastavit okno jako transparentní se přestalo zobrazovat úplně. Po několika hodinách experimentování s různými metodami vykreslování jsem došel k závěru, že externí přístup v tomto případě nebude dostačující. Místo hledání složitých a nestandardních způsobů, jak tento problém obejít, jsem se rozhodl změnit přístup a pokusit se implementovat wallhack interně.

2.3 Implementace interních cheatů

Při neúspěšné implementaci externího wallhacku bylo nutné rozhodnout se pro jiný přístup. Interní cheaty fungují na principu vložení vlastního kódu přímo do běžícího procesu hry, čímž umožňují přímý přístup k datovým strukturám a funkcím herního enginu. Tímto způsobem lze efektivněji manipulovat s hrou, aniž by bylo nutné číst a interpretovat hodnoty z paměti externí aplikací.

Měl jsem na výběr, zda interně implementovat pouze wallhack a ponechat aimbot spolu s triggerbotem jako externí programy, nebo zda celý systém přepracovat do interní podoby. Rozhodl jsem se pro druhou variantu, aby výsledný produkt byl kompaktní, snadno ovladatelný a všechny che-

aty mohly pracovat společně bez nutnosti komunikace mezi externím procesem a hrou. Tato volba mi navíc poskytla příležitost lépe se seznámit s interním přístupem a prozkoumat možnosti přímé interakce s herním enginem.

Přechod z externího na interní řešení byl usnadněn tím, že většina kódu aimbotu a triggerbotu mohla být zachována ve své původní podobě, případně s minimálními úpravami. Největší změnou bylo, že místo čtení hodnot z dynamicky se měnících adres v paměti bylo nyní možné přímo pracovat se strukturami hráčů uvnitř kódu OpenAreny. To eliminovalo potřebu manuálního vyhledávání adres a jejich aktualizace při každém spuštění hry. Díky přímému přístupu k datovým strukturám bylo možné snadno získávat klíčové informace, jako jsou pozice hráčů nebo týmová příslušnost, což výrazně snížilo množství proměnných a zlepšilo čitelnost samotného kódu. Dostat se však k samotnému přepisování aimbotu a triggerbotu do interní podoby stálo poměrně velkou časovou investici do zkoumání teorie a základů této implementace, která celý projekt výrazně zdržela.

2.4 Implementace injektoru

Implementace injektoru v podstatě odpovídala teoretickému postupu popsanému v předchozí kapitole, kdy hlavním úkolem bylo zavést DLL soubor do procesu hry. Prvotní verze injektoru byla jednoduchá a obsahovala pevně zadanou cestu k DLL souboru uloženou přímo jako absolutní cesta odpovídající struktuře mého počítače. Tento přístup však nebyl univerzální, protože by nefungoval na jiném zařízení s odlišnou adresářovou strukturou. Pro funkci injektoru na různých zařízeních, bylo nutné implementovat dynamický přístup k DLL souboru. Relativní cesta k OpenAreně se totiž liší od relativní cesty k injektoru, což znamená, že jednoduché použití pevně definované cesty nebylo možné. Řešením bylo získání absolutní cesty k DLL souboru v době běhu programu, což umožnilo jeho správné načtení bez ohledu na to, kde se injektor a hra nacházejí.

2.5 Interní implementace aimbotu a triggerbotu

Implementace interního aimbotu v zásadě vycházela z již hotového externího řešení, přičemž hlavní rozdíl spočíval v tom, že nyní bylo možné přímo přepisovat hodnoty úhlů pohledu hráče díky nalezení adresy, která umožňovala jejich úpravu. Tento přímý zásah do paměti hry odstranil nutnost simulace pohybu myši, což vedlo k plynulejšímu a přesnějšímu zaměřování. Jedním z prvních vylepšení, které v externí verzi chyběli, bylo zavedení podmínky, která kontroluje, zda je hráč stále naživu. Dříve totiž docházelo k problému, kdy aimbot zůstával zaměřený na nepřítele i po jeho smrti, dokud jeho tělo nezmizelo. Další úpravou bylo mírné snížení cílového bodu při výpočtu směrového vektoru k nepříteli. Odečtením několika jednotek na výškové ose se zajistilo, že pokud nepřítel stojí, aimbot míří do středu jeho těla, zatímco v případě, že se skrčí, se zaměřuje přímo na jeho hlavu.

Interní aimbot je implementován pomocí hookování funkce CL_CreateCmd, která má za úkol vytvořit strukturu obsahující informace o hráčově vstupu, jež jsou následně odesílány na server. Díky tomu nemusí aimbot běžet v nekonečné for smyčce jako v jeho externí verzi. Aimbot je připojen na konec této funkce, tedy těsně předtím, než je vytvořený usercmd předán zpět volající funkci. Tento přístup umožňuje provádět úpravy přímo v této struktuře a manipulovat tak s pohledem hráče. Klíčovým aspektem této implementace je, že po provedení změn nedochází k návratu zpět

do CL_CreateCmd, ale skok je proveden přímo do funkce, která ji volala. Tento přístup zajišťuje plynulou integraci aimbotu do běžného herního cyklu a eliminuje možné konflikty nebo nežádoucí opětovné přepisování hodnot. Adresa funkce CL_CreateCmd byla identifikována pomocí disassembleru Ghidra, který umožnil nalezení její adresy na základě jména uvedeného ve zdrojovém kódu OpenAreny.

Interní triggerbot byl implementován téměř identicky jako jeho externí verze. Stále umožňuje aktivaci a deaktivaci v reálném čase během toho, kdy je aimbot zaměřen na nepřítele. Tato jednoduchá logika byla ponechána, protože se mi během vývoje nepodařilo najít adresu funkce, která by určovala, zda je hráčův pohled přímo zaměřen na jiného hráče. Triggerbot je hookován stejným způsobem jako aimbot, tedy těsně před návratem z CL_CreateCmd.

2.6 Implementace wallhacku

Interní wallhack funguje tak, že prochází seznam všech hráčů ve hře a kontroluje, zda jsou naživu. Pokud ano, okolo jejich modelu je vykreslen obdélník – červený pro nepřátele a modrý pro spoluhráče. Na rozdíl od neúspěšné externí verze zde není potřeba vytvářet překryvná okna nebo řešit transparentnost.

K určení správné polohy obdélníku na obrazovce se nejprve spočítá směrový vektor mezi hráčem a ostatními postavami ve hře. Tento vektor je následně převeden na doplňkové úhly pitch a yaw, tedy úhly, které by bylo nutné nastavit, aby se hráčova kamera zaměřila na daného protivníka. Tyto úhly se dále transformují do dvourozměrných souřadnic na obrazovce, které určují přesnou polohu obdélníku. Souřadnicový systém obrazovky lze popsat tak, že osa X má hodnotu 1 na levém okraji a -1 na pravém, zatímco osa Y má hodnotu -1 dole a 1 nahoře, přičemž střed obrazovky odpovídá bodu (0,0). Pro správnost výpočtu, je nutné vzít zápornou hodnotu pitch a yaw, protože OpenArena definuje jejich záporné hodnoty směrem nahoru (pitch) a doleva (yaw). Výsledné hodnoty jsou normalizovány vydělením pitch šířkou zorného pole ve stupních a yaw násobkem šířky zorného pole a základního poměru stran OpenAreny, který je 4:3. Tím je zajištěno, že obdélník vždy odpovídá skutečné poloze hráče v herním prostoru a správně se přizpůsobuje různým rozlišením obrazovky.

Pro samotné vykreslování je použito OpenGL. Nejprve je ukládán jeho aktuální stav, poté jsou provedeny potřebné změny pro vykreslení obdélníků a nakonec je OpenGL navráceno do původního stavu, aby se nenarušilo renderování samotné OpenAreny. Tento postup je nezbytný, protože OpenArena si během vykreslování spravuje vlastní OpenGL kontext a jakékoliv nevrácené změny by mohly způsobit grafické chyby nebo artefakty v obrazu. Uložením a následným obnovením se stavu zajišťuje, že hra pokračuje v renderování správně i poté, co se wallhack vykreslí.

Funkci wallhacku hookuji na SDL_GL_SwapBuffers, která je volána těsně předtím, než je na obrazovce vykreslen nový frame. Hookování probíhá na začátku této funkce, což znamená, že obdélníky označující nepřátele i spoluhráče se vykreslí ještě předtím, než se samotná hra překreslí na obrazovce. Tato funkce se nachází v knihovně SDL.dll, kterou OpenArena využívá pro správu grafiky. Její adresu jsem získal pomocí GetProcAddress, což umožňuje dynamicky načíst adresu funkce z dané knihovny během běhu programu. Díky tomu lze provést hook i bez nutnosti přímé manipulace s binárním souborem OpenAreny.

Aby se předešlo vykreslování obdélníků i pro hráče, kteří nejsou na obrazovce, bylo nutné zavést kontrolu viditelnosti. Jelikož jsou převedené souřadnice hráčů reprezentovány v soustavě s rozsa-

hem od -1 do 1, implementoval jsem podmínku, která ověřuje, zda se obdélník nachází v těchto mezích. Pokud by jeho okraje přesahovaly mimo tuto oblast – tedy mimo obrazovku – nevykreslí se.

3. Technická dokumentace

3.1 Instalace, spuštění a použití

Cheaty půjdou spustit pouze na operačním systému Windows.

K instalaci je zapotřebí z repozitáře stáhnout soubory OADLL.dll a OAInjector.exe. Pro jednodušší spuštění je doporučeno je stáhnout do stejné složky, nebo je do jedné přemístit. Dále je nutné stáhnout a extrahovat soubor openarena-o.8.8.zip, ve kterém je kopie oficiálně distribuované verze OpenAreny zkompilované pro Windows, pro kterou tyto cheaty fungují. Tento odkaz, ke stažení tohoto zip souboru, naleznete na odkazu poskytnutém v README repozitáře. Jedná se o třetí stranu jelikož, oficiální stránky OpenAreny v době vypracování práce nefungují. Celý extrahovaný soubor umístěte na plochu a z ní hru spouštějte. Je to nejjednodušší způsob jak se vyhnout problémům s přístupem administrátora.

Spusťte openarena.exe a začněte hru na lokálním serveru. V nabídce herních map vyberte typ týmový typ hry, nejvíce je doporučen Team Deathmatch nebo Capture The Flag. Přidejte počítačem ovládané hráče, boty, před začátkem hry nebo po jejím spuštění. Jakmile hra běží a jsou přítomni nepřátele a spoluhráči můžete manuálně přetáhnout soubor OADLL.dll do souboru OAInjector.exe, tím se cheaty spustí a měla by se otevřít konzole OpenAreny, ve které budou vypisovány informace o přítomných hráčích. Můžete se vrátit do hry a cheaty používat.

Během hry je možné jednotlivé cheaty vypnout a zapnout. Aimbot se dá vypnout stisknutím klávesy F1 a wallhack klávesou F2. Triggerbot je nutné spouštět manuálně klávesou F. Triggerbot je možné spustit a vypnout během toho co se aimbot zaměřuje na hráče.

Závěr

Úspěšně jsem splnil stanovené cíle a podařilo se mi vyvinout aimbot, wallhack i triggerbot. Celkový čas strávený na vývoji, výzkumu a učení se dosáhl téměř sedmdesáti hodin. Přestože jsem nakonec dosáhl požadovaného výsledku, zpětně vidím určité chyby v přístupu.

Za jednu z největších chyb považuji rozhodnutí začít s vývojem externích cheatů. Problémy s externím wallhackem jsem mohl předvídat, kdybych se na začátku více zaměřil na analýzu možností a ne pouze na implementaci externího aimbotu. Kvůli tomu se celý projekt protáhl déle, než bylo nutné. Na druhou stranu mi tento postup umožnil naučit se základy injektování a hookování, což je znalost, kterou bych jinak opomenul.

Největším přínosem projektu byla schopnost orientovat se v binárním kódu OpenAreny a porozumět struktuře hry při hledání potřebných adres. Získané zkušenosti mohou být užitečné i v jiných oblastech reverzního inženýrství nebo vývoje softwaru pracujícího s cizím kódem.

Pokud jde o možnosti zlepšení, vidím prostor především v triggerbotu. Ten by fungoval spolehlivěji, kdyby dokázal přímo rozeznávat, zda se kamera dívá na nepřítele. Dalším vylepšením by bylo přesnější určování týmové příslušnosti hráče, což by umožnilo funkčnost cheatů i v herních módech, které nejsou týmové.

Celkově byl tento projekt náročnou, ale cennou zkušeností, která mi pomohla rozšířit mé znalosti v oblasti práce s pamětí procesů, analýzy binárního kódu a grafického renderingu.

Seznam použité literatury

[And18] Dennis Andriesse. *Practical Binary Analysis: Build Your Own Linux Tools for Binary Instrumentation, Analysis, and Disassembly*. San Francisco: No Starch Press, 2018.

[Eri08] Jon Erikson. *Hacking: The Art of Exploitation. 2nd ed*. San Francisco: No Starch Press, 2008.

[SH12] Michael Sikorsi a Andrew Honig. *Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software*. San Francisco: No Starch Press, 2012.

[Unknd] UnknownCheats. *Naked Hook*. [cit. 2025-02-18]. n.d. URL: `https://www.unknowncheats.me/forum/direct3d/54194-naked-hook.html`.

Seznam obrázků

Seznam tabulek