Assault CUBe mod menu

Valentyn Repinskyi

Inhaltsverzeichnis

[1 Vorbereitung 1](#_Toc105970687)

[1.1 Lernen 1](#_Toc105970688)

[1.2 Software 1](#_Toc105970689)

[1.2.1 Cheat Engine 1](#_Toc105970690)

[1.2.2 Visual Studio 1](#_Toc105970691)

[1.2.3 ReClass 1](#_Toc105970692)

[1.2.4 Interactive Disassembler 2](#_Toc105970693)

[1.2.5 Assault Cube 2](#_Toc105970694)

[2 Durchführung 3](#_Toc105970695)

[2.1 Cheat Engine / ReClass 3](#_Toc105970696)

[2.1.1 Pointer für Munition erstellen 3](#_Toc105970697)

[2.1.2 Entity base pointer finden 3](#_Toc105970698)

[2.1.3 Die nötigen Pointer über die Entity List herausfinden 4](#_Toc105970699)

[2.1.4 Recoil finden 4](#_Toc105970700)

[2.1.5 Recoil Funktion finden 4](#_Toc105970701)

[2.2 Visual Studio 5](#_Toc105970702)

[2.2.1 Konsolenkursor entfernen 5](#_Toc105970703)

[2.2.2 Funktion gotoXY 5](#_Toc105970704)

[2.2.3 Funktion MoveWindow 5](#_Toc105970705)

[2.2.4 Pointer erstellen 6](#_Toc105970706)

[2.2.5 Darstellung 6](#_Toc105970707)

[2.2.6 Cheat - Aktivierung 6](#_Toc105970708)

[3 Anhang 8](#_Toc105970709)

[3.1 Tagesberichte 8](#_Toc105970710)

[3.2 Recoil Funktion 9](#_Toc105970711)

# Vorbereitung

## Lernen

Um etwas über das Material für das Modding von Spielen zu erfahren, habe ich den folgenden Beitrag im Internet benutzt: https://guidedhacking.com/threads/ghb1-start-here-beginner-guide-to-game-hacking.5911/

Diese Seite enthält alles, was man wissen muss, wenn man mit dem Hacken beginnen will. Auf dieser Seite habe ich mir zahlreiche Tutorials zum Thema Game Hacking angesehen. Zuerst lernt man, wie der RAM-Speicher funktioniert, dann kann man mit Tools wie Cheat Engine oder ReClass direkt mit dem Hacken loslegen. Aber wenn man solche Dinge automatisieren will, muss man eine Programmiersprache lernen. Auf der folgenden Website: learncpp.com, habe ich 21 Kapitel durchgearbeitet und unterschiedliche Aufgaben absolviert. Da diese Sprache nicht meine erste ist, war es für mich viel einfacher, sie zu lernen. Während ich C++ lernte, programmierte ich meine eigenen kleinen Spiele, Tools usw. Dadurch bekam ich auch einen Überblick darüber, wie Spiele auf dieser Sprache aufgebaut sind und wie man sie hacken kann.

## Software

Zuerst um mit der Arbeit anzufangen, bereite ich die Folgenden Programme vor.

### Cheat Engine

Auf der offiziellen Webseite vom Cheat Engine (Abk. CE) <https://www.cheatengine.org/> installiere ich das Tool, um damit herausfinden zu können, wie ein Spiel/eine Anwendung funktioniert, und Änderungen daran vornehmen können.

### Visual Studio

Zunächst installiere ich die integrierte Entwicklungsumgebung (IDE) Software für die Anwendungsentwicklung Visual Studio von Microsoft auf meinen Computer. Dazu muss die Erweiterung für das Programmieren auf der Sprache C++ installiert werden. Mit dieser IDE werde ich den Code für das Mod Menu implementieren.

### ReClass

Als nächstes kommt das Programm ReClass, das die Prozess-Handles eines anderen Prozesses verwendet, um auf die Daten des Ziels zuzugreifen. Hiermit wird die Suche von Offsets vereinfacht.

### Interactive Disassembler

Das Programm IDA wird zuletzt für das Cheaten benötigt, dieser ermöglicht Binärcode in Assemblersprache umzuwandeln, um leichter den Programmcode von dem Spiel zu verstehen.

### Assault Cube

Zuletzt installiere ich das Spiel Assault Cube, für welches dann dieses Mod Menu erstellt werden soll.

# Durchführung

## Cheat Engine / ReClass

### Pointer für Munition erstellen

1.Temporäre RAM-Speicher Adresse für die Munition im Magazin finden.

2.-> "Find out what accesses this address"

3.-> Im Spiel die Menge an Munition verändern und CE betrachten welche Adresse auf diesen Wert zugreift.

4.-> Die Adresse in [ ] eckigen Klammern kopieren, Offset abspeichern

5.-> Adresse und Offset in die pointerTabelle reinkopieren

Address = Value = ?

base ptr -> address + offset4 = address

base ptr -> address + offset3 = address

base ptr -> address + offset2 = address

static base -> address + offset1 = address

6.-> Ab Schritt 3 wiederholen, solange bis man die static base pointer address findet

7.-> "Add Address Manually" ->Pointer Häckchen ->static base address und alle offsets eintragen.

8. Pointer erstellt.

### Entity base pointer finden

1. - > Rechtsklick Munition Pointer

2. - > "Find out what accesses this address"

3. - > Im Spiel die Munitionsanzahl verändern

4. - > [edi]-Adresse kopieren und dann nach diesem hex Wert "New Scan"

5. - > Wenn es schon die statische Adresse für das [edi] vorhanden ist, hat man den Pointer gefunden

### Die nötigen Pointer über die Entity List herausfinden

-> "Memory View" -> "Tools" -> "Dissect data/structures"

-> Die Enity base address einsetzen -> "S

|  |
| --- |
| Recoil finden |
|  |

|  |
| --- |
| -> Zuerst füge ich ReClass den ac\_client.exe Prozess an |
|  |

|  |
| --- |
| -> Danach füge ich die statische Entity Player Object Adresse in ReClass hinzu |
|  |

|  |
| --- |
| -> Es werden alle Offsets aufgelistet, die mit dieser Adresse verbunden sind |
|  |

|  |
| --- |
| -> Der Offset für Aktuelle Waffenmunition liegt bei 0x374 |
|  |

|  |
| --- |
| -> Zunächst ändere ich den Typ dieser Adresse auf Pointer, und füge 256 Bytes hinzu, damit die verbundenen Adressbereiche aufgelistet werden |
|  |

|  |
| --- |
| -> Danach finde ich die aktuelle Waffe als String dargestellt, diese ändere ich wieder auf Pointer |
|  |

|  |
| --- |
| -> Ich füge dem Pointer 256 Bytes hinzu und ganz unten finde ich Adressen Werte die vielleicht etwas mit der Aktuellen Waffen zu tun habe |
|  |

|  |
| --- |
| -> Dies prüfe ich indem, ich die Werte auf Int16 Typ wechsle und somit sehe gleiche Werte, die ich im Spiel beobachten kann |
|  |

|  |
| --- |
| -> Zunächst versuche ich herauszufinden, welche der Adressen dem Recoil entsprechen, indem ich die Werte einfach bearbeite und die Folgen im Spiel beobachte |
|  |

|  |
| --- |
| -> Zwei Recoil Adressen wurden gefunden. Eine bewegt die Spieler Sicht nach oben und andere nach links und rechts |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| Recoil Funktion finden |
|  |

|  |
| --- |
| -> Da die zwei Adressen nicht ausreichen, um vollständig Recoil zu stoppen, versuche ich die Funktion im Disassembler zu finden |
|  |

|  |
| --- |
| -> Nun kopiere ich die Recoil Adresse und gehe zu IDA Anwendung |
|  |

|  |
| --- |
| -> Mit IDA öffne ich die exe Datei von diesem Spiel |
|  |

|  |
| --- |
| -> Danach öffne ich das Plugin Class Informer, welches mir die Namen der Strukturen auflistet |
|  |

|  |
| --- |
| -> Ich suche nach dem Namen "Weapon" und wähle eine Waffe-Struktur aus |
|  |

|  |
| --- |
| -> Jetzt gehe durch alle Funktionen und schaue ob sie für mich interessant sein könnten |
|  |

|  |
| --- |
| -> Ziel ist es eine Funktion zu finden die den Recoil berechnet und aufruft |
|  |

|  |
| --- |
| -> (Man findet sie in dem man sich den Code ausließt und herausfindet wo die zwei Offsets für die Recoil Adressen sich befinden) |
|  |

|  |
| --- |
| -> In der CallRecoil Funktion findet man die Adresse wo die Funktion maybeRecoil aufgerufen wird |
|  |

|  |
| --- |
| -> CE Memory View -> Suche 0046378E |
|  |

|  |
| --- |
| -> Setze ein Breakpoint um diese Adresse zu debuggen und herauszufinden welche Adresse sie aufruft |
|  |

|  |
| --- |
| -> Nun gehe ich zu der Adresse und ändere die Assembler Anweisung auf ret 8 (Mit RET wird ein mit dem CALL-Befehl aufgerufenes Unterprogramm beendet) |
|  |

-> So hat man in dem Spiel keinen Recoil

## Visual Studio

### Konsolenkursor entfernen

Damit der Konsolenkursor nicht in der Konsole blinkt, schalte ich ihn mit einer Standartfunktion aus MSDN-Webseite.

### Funktion gotoXY

Um die Werte in der Konsole zu ändern, wie z.B. Heath auf 999 zu setzen, muss ich in der Konsole zu einer bestimmten Position springen und den Text an dieser Stelle umschreiben.

### Funktion MoveWindow

Damit die wichtigsten Sachen in der Konsole angezeigt werden, wird eine Funktion zur Anpassung der Konsolengröße benutzt.

### Pointer erstellen

Damit alle gefundenen Pointer eine Adresse im Spiel zugeordnet werden könnten, erstellet man eine Variable mit der Modulebase Adresse. Diese kann man auch im IDA als Imagebaseaddress nachprüfen.

LocalplayerAddress aus dem Cheat Engine zu der Module Base Address zufügen.

Pointer zu folgenden Adressen mittels Funktion FindDMAAddy erstellen: xyz - Positionen, health, superJump.

### Darstellung

Großes Titel ACMOD mittels Unicodes (wchar) und einer Art Seite einfügen.

Bibliothek für Farben in der Konsole implementieren.

### Cheat - Aktivierung

Mittels Funktion GetAsyncKeyState Tasten für die Cheat-Aktivierung hinzufügen.

Für den Health-Cheat werden fünf verschiedene States eingefügt. Zuerst reset auf 100 und Health-Subtract. Danach Funktion NOP (Funktion mit toten Bytes ersetzen) und anschließend ADD-Cheat (Beim Damage eigenes Leben addieren).

Für ADD und SUM wird die Funktion PatchEX benutzt, um die ausgewählten Bytes zu ersetzen.

Um für das Leben einen bestimmten Wert zu setzen benutzt man aus der windows.h Bibliothek die Funktion WriteProcessMemory, um einen bestimmten Wert einer Adresse aus dem RAM-Speicher zu ändern.

Zunächst für das Munition-Patch benutze ich nur einen Cheat, die Munition bei jedem Schuss zu addieren. Dafür schreibe ich nur ein Byte dieser Funktion um. Und somit ändert sich die ASMBLER Anweisung von dec zu inc (decrement, increment).

Zunächst NOPe ich die Funktion des Recoil und setze die wieder zurück.

Für superJump wird die jump Adresse auf vier floats umgeschrieben. dh., wenn ich dies ausführe, springt der Spieler auf vier LE nach oben. Diese Funktion verbinde ich dann mit dem Leerzeichen und somit, wenn man einmal springt, bleibt alles ganz normal und wenn man zwei Mal die Taste drückt, aktiviert sich der Cheat.

Teleport Funktion

Zuerst wird die gewünschte stelle mit einer Taste ausgelesen und wenn die Koordinaten nicht 0 sind, teleportiert man sich auf die gewünschte Stelle.

Speed Hack

Mit dem Cheat Engine sucht man nach Adressen, welche die xyz Position beeinflusst und am Ende patcht man jeweils ein Byte pro laufende Richtung. Also verändert sich die maximale Geschwindigkeit von 1f auf 2f. Im Game Design wird für die Bewegung eine maximale float-Zahl zum Multiplizieren der Geschwindigkeit benutzt und diese schreibe ich somit um.

Und jeweils pro Funktion wird der Cheat zb. NOP/ADD... in der Konsole mittels gotoXY den Wert umschreiben.

Um das Cheat zu beenden wird die Taste 0 auf dem Numpad benutzt und alle Hacks werden aus dem Spiel entfernt.

# Anhang

Projekt auf GitHub mit Bildern und Cheat Tables: <https://github.com/A2E9/AC_External>

## Tagesberichte

**14.02.2022**

Erlernen der C++ Sprache: Fundamental Data Types: learncpp.com

**21.02.2022**

Installieren und Konfigurieren der Software: Cheat Engine, ReClass, AC

**28.02.2022**

Installieren und Konfigurieren der Software: Visual Studio und IDA. (lange Installation). Lernen der C++ Sprache: Compound Types: References and Pointers: learncpp.com

**07.03.2022**

Pointer für Munition mittels Cheat Engine erstellen.

**14.03.2022**

Entity base Pointer finden und damit die nötigen Pointer herausfinden.

**21.03.2022**

Recoil Funktion finden.

**28.03.2022**

Weitersuche der Recoil-Funktion.

**04.04.2022**

In Visual Studio neues Projekt erstellen und für das Mod Menu vorbereiten.

Mod Konsole anpassen.

**25.04.2022**

Verweis auf Pointer erstellen.

**02.05.2022**

Programmierung von Cheats. Tasten für Cheat-Aktivierung einbinden

**09.05.2022**

Programmierung von Cheats. Leben, Munition, Recoil, Speed Hack

**23.05.2022**

Programmierung von Cheats. Teleport, Jump Hack, Cheat Exit

## Recoil Funktion

