How we organised the team,

we conducted several team meetings during the duration of the project. In our first meeting we had a group call on discord on 05/11/23. During this meeting we discussed which game we’d like to pursue based on the complexity of the hack we could do based on the game and the difficulty of hacking the game. After our discussion we voted and decided on

Counter-Strike: Global Offence. Our second meeting was on 06/11/23 and was when we decided on the hack we wanted to attempt, everyone came up with ideas during the meeting, but we were not able to definitively decide on the hack, but we all had a good idea of the type of things we wanted to do. After the call we all added hack ideas into our discord server and then voted on the one we all liked the most. We ultimately landed on a one shot one kill game mode added to CSGO. We had a third meeting on 09/11/23 where we decided on what kind of work each person would do. Everyone except zeerak could run CSGO on windows so as a group we decided to allocate static analysis using Ida, research on similar game hacks using dll injection and organising the structure and contents of our video and doing the group side of this report. Everyone else began work on the dll, Andrew created a GitHub repository which we then used to share our work.

Challenges

The gun array isn’t of fixed size so working out how to not over read the list was a challenge. The array could increase when a player buys a gun and decrease when the round is over. The array also includes knives, grenades and the terrorist bomb, these can be identified by their clip ammo equalling -1. When looking for the value in Cheat Engine there is multiple values that can be correlated with the size of the array, this is because there are values for: server values, induvial player values, items bought and team values. After trial and error using a server value -2 gave a size that didn’t cause read errors.

The player array is of fixed size to the number of players in the server and it’s an array of pointer to the player object. However, the array has a structure for storing players which made it very confusing to understand. It stores both T and CT players in a random order but only alive players and stored in the front section of the array and dead players get moved to the back. This confused us into thinking only alive players were stored in this list and we would require memory comparison for our own death function. In the end we caught our mistake.

Percentages

|  |  |  |
| --- | --- | --- |
| Name | Contributions | Percentage |
| Craig | * Wrote code for the player death functionality * Created pointer functions with offsets with Andrew * Used IDA to identify some offsets, later tested in Cheat Engine * Created Player object logic with Andrew * Did some paired programming whilst testing and debugging runtime errors * Searched for console offsets in IDA and Cheat Engine | 10% |
| Satvir |  |  |
| Zeerak |  |  |
| Andrew | * Started group communications and repository * Researched games and came up with project ideas * Found every offset we used in Cheat Engine * The code for initializing the DLL including server.dll base address, menu and console interrupt * Created pointer mapper function * Jointly with Craig created pointer functions * Created C++ classes and the logic for the data we required * Created game mode logic * Created logic for gun array * Jointly with Craig created the logic for player * Jointly with Craig doing testing and debugging runtime issues * Provided video segments to Zeerak | 70% |
| Bogdan |  |  |
| Perrin |  |  |