Logiciel Antriche - Cahier des charges

.Objectifs

Développer un anticheat performant utilisant C# et Mono, capable de détecter différents types de cheats tout en étant pas trop gourmand sur les performances.

Architecture

- Développement en C# (4.5.2) avec Mono
- Newtonsoft.Json pour la sérialisation des données
- Système de P2P pour les vérifications
- Multi-threading pour optimiser les performances

Problématique

Les tricheurs peuvent facilement stopper les threads Lua traditionnels, rendant inefficaces les systèmes de détection.

Solution proposée

- Système de tick en C# exposé via des exports vers Lua
- Utilisation des boucles natives C# qui ne peuvent pas être bloquer par les cheats habituelles

Problématique

Les vérifications côté client sont vulnérables au “hooking” des fonctions natives (exemple : un tricheur peut modifier IS_PLAYER_VISIBLE pour qu’elle renvoie toujours true). Les vérifications côté serveur seraient idéales mais trop gourmande en ressources pour 2000+ joueurs.

Solution proposée

- Chaque client vérifie les joueurs dans sa zone de proximité
- Les résultats des vérifications sont envoyés au serveur
- Le serveur agit seulement lorsque plusieurs clients signalent le même joueur pour la même infraction

Fonctionnement détaillé

1. Le client exécute des fonctions natives pour vérifier ces joueurs
    - Exemples : IS_PLAYER_VISIBLE, GET_ENTITY_HEALTH, etc.
2. Si une erreur est détectée, le client flag le joueur au serveur
3. Le serveur cumule les rapports provenant de différents clients

4. Si le serveur possède plusieurs “flags” pour un joueur alors il sera banni du serveur

Détection des modifications

- Invisible
- Vie infini
- Armure infini
- Joueur trop rapide
- Téléportation du joueur