Damocles

NyanHeroes 游戏分析报告

2024.05.21

Senna

DAMOCLES LABS

Damocles

目录

Damocles

一、 概要

作为一款 FPS 品类的游戏 NyanHeroes 在其客户端的安全性为 0，由于 STG 品类的游戏对公平性要求极高，如果公平性丧失，对于普通用户的游戏体验，整体的代币产出将会失衡，同时游戏结算部分逻辑疑似存在漏洞。同时游戏在上线初期存在 PDB 泄露的问题，并且为介入任何反作弊系统，因此 Damocles 判定其安全评分为 1 星。

安全性评分： ★ ☆ ☆ ☆ ☆

二、 游戏背景

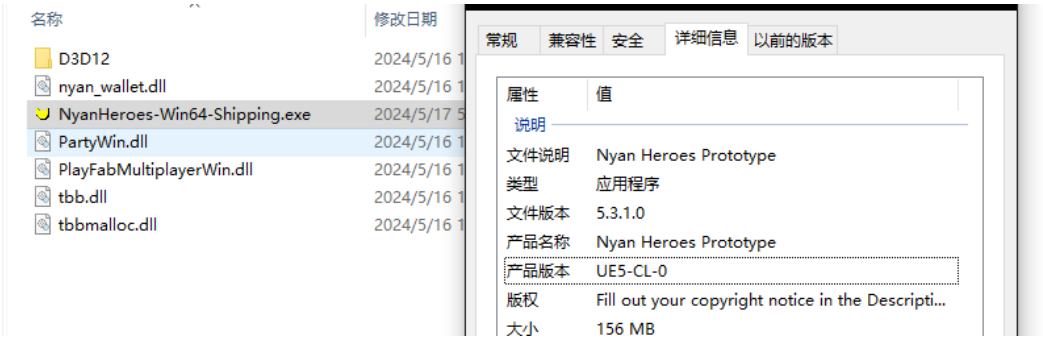
- 进行评估的游戏版本：Beta01
- 游戏类型&游戏引擎：STG&FPS，UE5.3
- 游戏玩法可能存在的问题：
  - 非法移动(修改本地人物属性进行加速)
  - 无后坐力
  - 自瞄
  - 瞬移
  - 结算重放攻击
  - 跳跃等一些本地人物属性的修改
  - 透视
  - 秒杀

三、 游戏安全性分析

游戏代码保护：

分析过程：

1. 由于不同的引擎有不同的分析模式，所以在获取到游戏 EXE 后首先需要确定游戏使用的引擎，通过对游戏基础信息识别我们可以确定该游戏是使用 UE5 进行开发。



2. 通过工具进行 dump UE 的人物结构进行快速定位，定位以后通过 UE 特有的链表结构进行索引与修改

Damocles

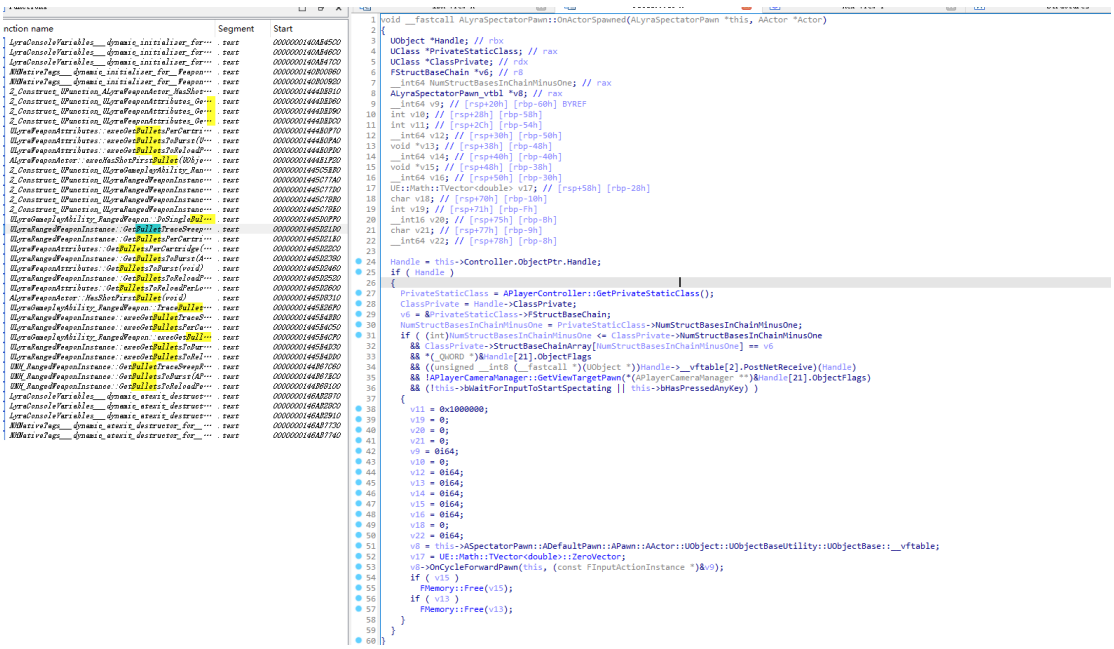
```
/// Class /Game/Lyra/Effects/Blueprints/B_WeaponDecals.B_WeaponDecals_C
/// Size: 0x0068 (104 bytes) (0x000298 - 0x000300) align 8 pad: 0x0000
class AB_WeaponDecals_C : public AActor
{
public:
    FPointerToUberGraphFrame                          UberGraphFrame;
    class USceneComponent*                            DefaultSceneRoot;
    class UNiagaraComponent*                          Decal;
    bool                                              Impact_Trigger;
    unsigned char                                     UnknownData00_6[0x7];
    class UNiagaraSystem*                             Decal_System;
    TArray<FVector>                                   ImpactPositions;
    TArray<FVector>                                   ImpactNormals;
    SDK_UNDEFINED(16,1302) /* TArray<TEnumAsByte<EPhysicalSurface>> */ __um(Impact_Surface_Types);
    TArray<int32_t>                                   Surface_Types;


    /// Functions
    // Function /Game/Lyra/Effects/Blueprints/B_WeaponDecals.B_WeaponDecals_C.AnimMotionEffect
    // void AnimMotionEffect(FName bone, FGameplayTag MotionEffect, class USceneComponent* StaticMesh
    // Function /Game/Lyra/Effects/Blueprints/B_WeaponDecals.B_WeaponDecals_C.Fire
    // void Fire();
    // Function /Game/Lyra/Effects/Blueprints/B_WeaponDecals.B_WeaponDecals_C.ExecuteUbergraph_B_Weap
    // void ExecuteUbergraph_B_WeaponDecals(int32_t EntryPoint);
};

/// Class /Game/Lyra/Characters/Heroes/Abilities/W_JumpTouchButton.W_JumpTouchButton_C
/// Size: 0x0008 (8 bytes) (0x0002C0 - 0x0002C8) align 8 pad: 0x0000
class UW_JumpTouchButton_C : public UUserWidget
{
public:
    class UW_ActionTouchButton_MobileOnly_C*          ActionTouchButton;
};

/// Class /Game/Lyra/UI/Hud/W_ActionTouchButton.W_ActionTouchButton_C
/// Size: 0x0288 (648 bytes) (0x0015A0 - 0x001828) align 16 pad: 0x0008
#pragma pack(push, 0x1)
class UW_ActionTouchButton_C : public ULyraButtonBase
{
public:
    FPointerToUberGraphFrame                          UberGraphFrame;
    class UWidgetAnimation*                           OnClickedTouchInput;
    class UWidgetAnimation*                           CooldownInactiveToActive;
```

并且使用 IDA 进行静态代码分析时发现游戏代码结构完整且毫无保护



因此可以结合 dump 的 UE 数据结构和 PDB 使用反编译工具对游戏代码逻辑进行基本的理解。

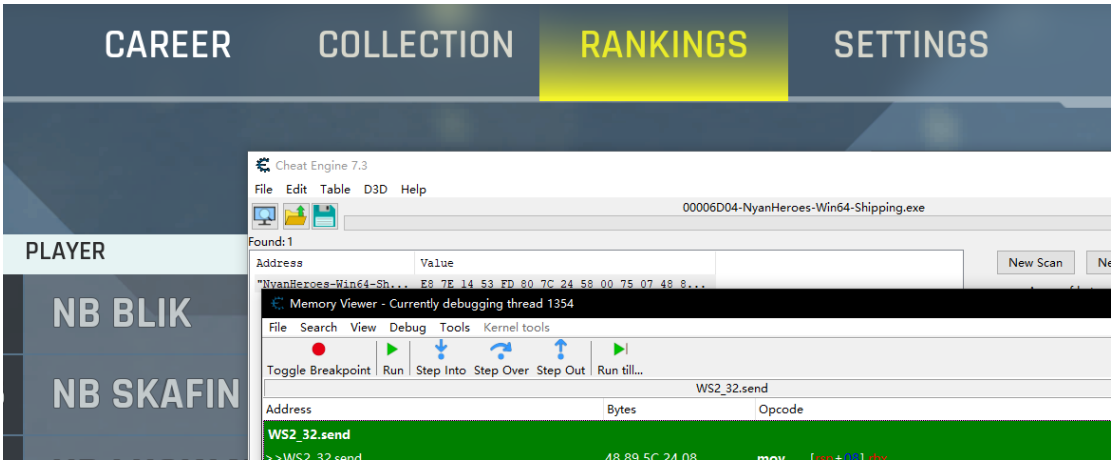
分析结论：

Nyan Heroes 在游戏代码保护方面得分为 0 分，其 client 代码没有任何保护结合 PDB 分析可以一定程度上认定为源码级别分析，也就是说没有任何的门槛手段对抗恶意玩家,同时并未接入任何的反作弊系统，例如免费的 EAC。

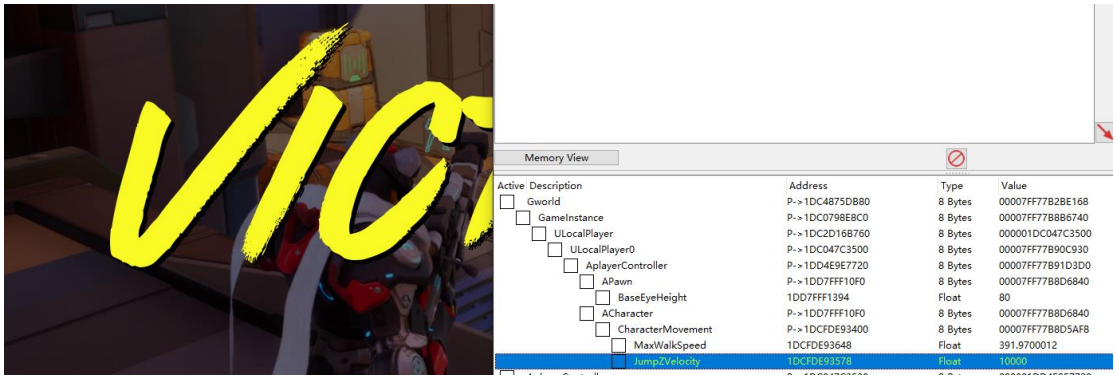
游戏基础反作弊：

分析过程：

1. 在基础反作弊检测方面，我们主要从两个方面进行测试，一个是游戏是否存在反调试，另一个是游戏是否存在读写保护。
2. 在游戏打开状态下使用 CE 进行附加，并且对通用函数进行下断点，发现游戏并没有退出，或者提示



3. 可以直接修改游戏内的人物属性例如速度、跳跃高度等属性，并且 GS 不会进行踢出操作。



分析结论：

1. NyanHeroes 在反作弊对抗上基本保护为 0，缺少针对动态调试，动态分析的对抗，因为对于想作恶的玩家来说成本很低，并且缺少对已经作弊的玩家的检测能力,同时由于 PDB 的泄露问题，可以针对性的分析游戏，例如 FPS 游戏内用来判定子弹是否命中的射线检测机制的影响因子是哪些。
2. 只测试反调试和读写保护两个方面的原因是对于一块外挂来说，找数据与实现功能只需要通过调试和读写就可以实现。如果最基础的两个保护能力都缺失的话，那么一些注入、hook 等检测也毫无意义。

游戏逻辑问题&外挂原理分析

分析过程：

在对游戏进行分析时我们发现 Nyanheroes 存在数据同步不完善的问题，同时我们在分析过程中发现市面上出现了外挂，基于此，我们针对游戏逻辑问题的分析进行扩充，引入外挂原理分析。

Nyan 使用的引擎为 Unreal Engine，在该引擎中的透视与自瞄的实现逻辑是可以套用模板的，其实现主要依赖人物相关的属性，具体为：

自瞄需要修改的属性：APlayerController->Apawn->FRotation { Pitch , Yaw, Roll}

透视需要获取的属性：APlayerController->Apawn->FLocation{X，Y ,Z}

得到这些数据的内存地址以后，通过矩阵转换，将二维数据变为三维数据，然后外挂程序进行计算以后，再在游戏中进行修改，之后则可以实现自瞄，或者透视。同时 UE 开发的 FPS 游戏一般都可以套用子弹追踪模板，因此希望项目方后期注意这方面的检测。

以使用起源引擎的 Apex 为例，其逻辑类似，如下：

Damocles

```
static void EspLoop()
{
        esp_t = true;
        while(esp_t)
        {
                std::this_thread::sleep_for(std::chrono::milliseconds(1));
                while(g_Base!=0 && c_Base!=0)
                {
                        std::this_thread::sleep_for(std::chrono::milliseconds(1));
                        if (esp)
                        {
                                valid = false;

                                uint64_t LocalPlayer = 0;
                                apex_mem.Read<uint64_t>(g_Base + OFFSET_LOCAL_ENT, LocalPlayer);
                                if (LocalPlayer == 0)
                                {
                                        next = true;
                                        while(next && g_Base!=0 && c_Base!=0 && esp)
                                        {
                                                std::this_thread::sleep_for(std::chrono::milliseconds(1));
                                        }
                                        continue;
                                }
                                Entity LPlayer = getEntity(LocalPlayer);
                                int team_player = LPlayer.getTeamId();
                                if (team_player < 0 || team_player>50)
                                {
                                        next = true;
                                        while(next && g_Base!=0 && c_Base!=0 && esp)
                                        {
                                                std::this_thread::sleep_for(std::chrono::milliseconds(1));
                                        }
                                        continue;
                                }
                                Vector LocalPlayerPosition = LPlayer.getPosition();

                                uint64_t viewRenderer = 0;
                                apex_mem.Read<uint64_t>(g_Base + OFFSET_RENDER, viewRenderer);
                                uint64_t viewMatrix = 0;
                                apex_mem.Read<uint64_t>(viewRenderer + OFFSET_MATRIX, viewMatrix);
                                Matrix m = {};
                                apex_mem.Read<Matrix>(viewMatrix, m);

                                uint64_t entitylist = g_Base + OFFSET_ENTITYLIST;

                                memset(players,0,sizeof(players));
                                if(firing_range)
                                {
                                        int c=0;
                                        for (int i = 0; i < 10000; i++)
                                        {
                                                uint64_t centity = 0;
```

分析结论：

1. 对于一款游戏来说，其本地逻辑的安全性与 GS 判定和本地的安全手段息息相关，从当前的游戏表现来看其 GS 对同步的数据缺乏管控，以及同步的数据并不完善，同时用户数据上报缺失，因此其在该处的安全评分为 0。

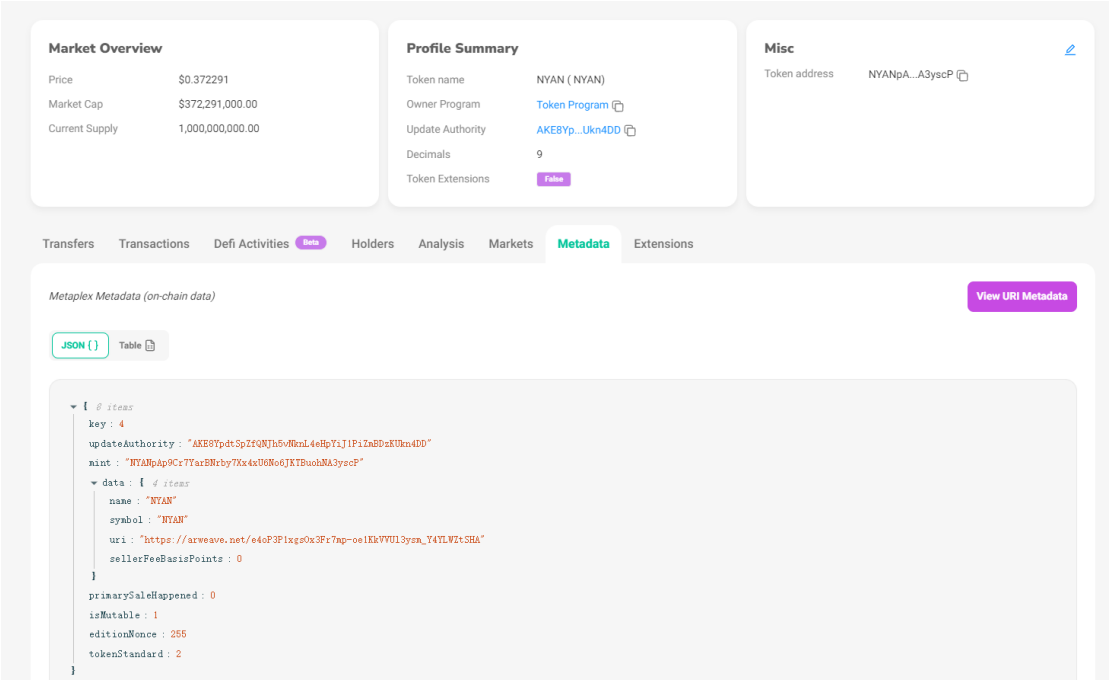
Damocles

游戏协议&Server 安全性分析

当前游戏协议偏少，主要是以对局结算为主，从 Ranking 数据来看疑似存在问题，例如胜场 15 但是 Rank 达到 5566。具体的漏洞在此不做讨论。

WEB3 安全分析：

Nyan 代币为发行在 Solana 上的 SPL Token，其均为统一模板开发， 因此安全性在此不做讨论。



Damocles

关于 Damocles

Damocles labs 是成立于 2023 年的安全团队，专注于 Web3 行业的安全，业务内容包括：合约代码审计，业务代码审计，渗透测试，GameFi 代码审计，GameFi 漏洞挖掘，GameFi 外挂分析，GameFi 反作弊。

我们会在 Web3 安全行业持续发力，并且尽可能多的输出分析报告，提升项目方和用户对 GameFi 安全的感知度，以及促进行业的安全发展。