



# BigTime 游戏分析报告

2023.11.03

Senna

DAMOCLES LABS

# 目录

- 概要
- 游戏背景
  - ◆ 游戏版本
  - ◆ 游戏类型&游戏引擎
  - ◆ 游戏玩法可能存在的问题
- 游戏安全分析
  - ◆ 游戏代码保护
  - ◆ 游戏基础反作弊
  - ◆ 游戏逻辑问题
  - ◆ 游戏 RPC 分析
- Web3 安全分析
  - ◆ 代币合约安全
  - ◆ 游戏内经济系统安全
- 关于 Damocles

## 一、 概要

BigTime 于 2023 年 10.10 号上线代币后引起了 GameFi 的热潮，团队从 9 月份开始关注 BigTime 但是苦于没有资格一直未进行分析，在最近降低注册门槛以后，我们开始针对 BigTime 进行一系列的安全分析与测试，其中包括针对游戏客户端属性篡改，GameRPC 恶意调用测试，代币合约审计等。通过对游戏的整体评估，我们发现该游戏的安全性较差，对于恶意玩家来看，其作弊成本低。并且游戏的分析难度低。如果项目方想持续运营游戏，提升游戏的安全性与公平性应该放在后期运营首位。

## 二、 游戏背景

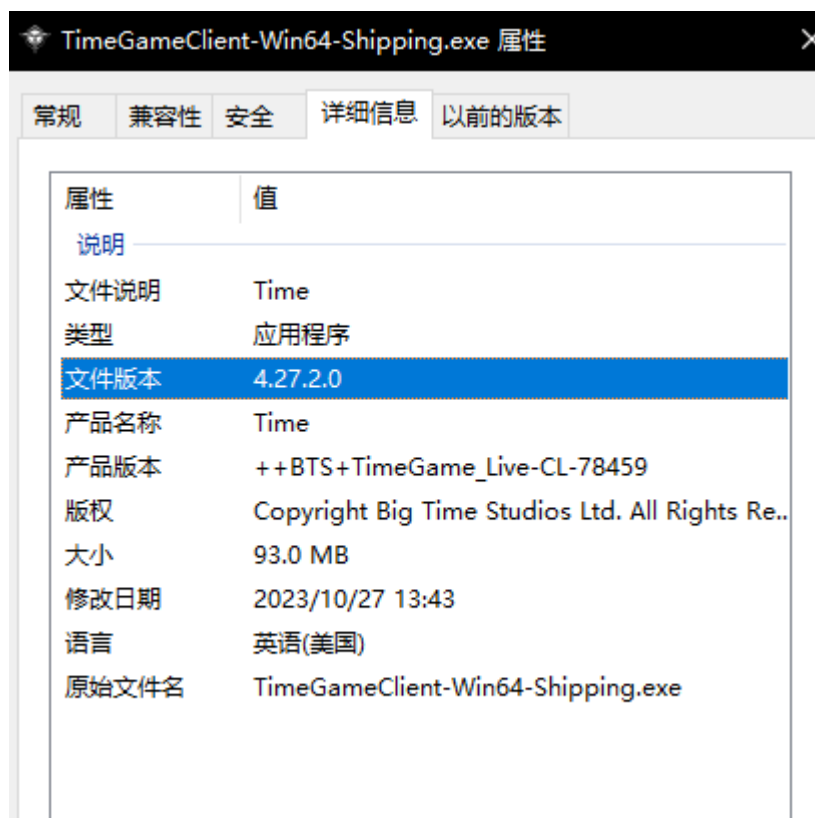
- 进行评估的游戏版本：v0.28-CL#78459
- 游戏类型&游戏引擎：MMORPG，UE4.27
- 游戏玩法可能存在的问题：
  - 非法移动（通过 RPC 进行恶意封包进行瞬移，加速等操作）
  - 加速（游戏内大世界时间，UE 框架下的 deltatime 属性，UE 框架下的时间函数）
  - 一键连段/一键技能循环
  - NFT 锻造加速
  - NFT 随机数操纵
  - 副本结束后的多重结算

### 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 UE4.27.2 进行开发。



2. 将游戏导入 IDA, 发现游戏代码未进行加固, 并且通过 UE4.27 的特征码搜索可以快速定位到 GWorld 变量

```

ext:00000001407FD33D F3 1C 43 75 31 25 03
ext:00000001407FD345 0F C9
ext:00000001407FD348 0F C3
ext:00000001407FD348 7D
ext:00000001407FD34D 4C 4D 75 31 25 03
ext:00000001407FD354 4C 4D
ext:00000001407FD357 75 31 25 03
ext:00000001407FD359 4C 4D
ext:00000001407FD35C 4C 4D
ext:00000001407FD35E 48 8B

```

```

movss xmm0, cs:dword_1407FD35E
xorps xmm1, xmm1
ucomiss xmm0, xmm1
jz short loc_1407FD360
mov rbx, cs:Global_Gworld
test rbx, rbx
jz short loc_1407FD362
mov r8b, 1
xor edx, edx
mov rcx, rbx

```

```

164 }
165 LOBYTE(v11) = dword_145A0C768 != 0;
166 sub_141847220(v12, v11);
167 if ( (qword_145A0EF88 )
168 sub_1453221900();
169 nullsub_2();
170 v17 = *(int *) (a1 + 3136);
171 v18 = 0;
172 LOBYTE(v3) = 0;
173 v68 = 0164;
174 v19 = 0164;
175 LOQWORD(v19) = v3;
176 if ( (int)v17 > 0 )
177 {
178 v28 = *(QWORD *) (a1 + 3128);
179 v21 = 0164;
180 while ( *(QWORD *) ( *(QWORD *) v20 + 640164 ) != Global_Gworld )
181 {
182 v19 = (unsigned int)(v19 + 1);
183 ++v21;
184 v20 += 0164;
185 if ( v21 >= v17 )
186 goto LABEL_37;
187 }
188 v68 = *(QWORD *) ( *(QWORD *) ( *(QWORD *) (a1 + 3128) + 8164 * (int)v19 ) + 176164);
189 LABEL_37:
190 v22 = 0;
191 v23 = 0164;
192 v76 = 0;
193 v7[0] = 0164;
194 do
195 {
196 v24 = *(QWORD *) (v23 + *(QWORD *) (a1 + 3128));
197 v25 = *(QWORD *) (v24 + 0x280);
198 if ( v25 && *(BYTE *) (v25 + 270) & 0x10 != 0 )
199 {
200 Global_Gworld = *(QWORD *) (v24 + 0x280);
201 (* (void __fastcall *) (__int64, __int64, __int64, __int64)) ( *(QWORD *) a1 + 1112164 ) (a1, v24, v21, v19);
202 if ( !v4 )
203 sub_1433092200( *(QWORD *) (v24 + 640), 2164);
204 if ( !byte_145B069C1 && (unsigned __int8)sub_143679E50( *(QWORD *) (v24 + 640) ) )
205 {
206 sub_14318AF00( *(QWORD *) (v24 + 640));
207 sub_143162A50( *(QWORD *) (v24 + 640), 0164, 0164, 0164);
208 }
209 v26 = *(QWORD *) (v24 + 640);
210 v27 = *(BYTE *) (v26 + 267);
211 if ( (v27 & 1) != 0 )
212 {
213 *(BYTE *) (v26 + 267) = v27 & 0xFE;
214 v28 = (QWORD *) sub_143655540(v24 + 208, L"causeevent", 0164);
215 if ( *(QWORD *) (v24 + 536) )
216 v29 = sub_14323AA00();
217 else
218 v29 = 0164;
219 v71 = v29;
220 if ( v28 && v29 )
221 {
222 v66 = 0164;
223 v67 = 0164;
224 sub_1408004D0(&v66, 12164);
225 v30 = HIDWORD(v67);

```

并且可以发现，字符串同样是没有加密。

```

rdata:00000000 C ComboNumber
rdata:00000001 C MComboFinisher
rdata:00000006 C PerComboHitDamageMult
rdata:00000017 C PerComboHitDamageTypes
rdata:00000018 C EndComboTriggeringState
rdata:0000001A C StartComboTriggeringState
rdata:00000024 C BYTE_FogofGhate_CAM_TRIGGERER_COMBO
rdata:0000000E C ComboAnimTags
rdata:00000013 C ComboHeavyAnimTags
rdata:0000000C C ComboUIData
rdata:0000000E C ComboUIData
rdata:00000029 C PerComboHitGASConfigGetTeamMaledowner
rdata:00000025 C PerComboAnimateScaleModifierToOwner
rdata:00000012 C FinisherComboIcon
rdata:0000002C C EFloatingTextDamageModifiers::ComboFinisher
rdata:00000010 C GetComboPrimaryComboAnim
rdata:00000011 C IncrementComboID
rdata:00000011 C IsNotLastComboId
rdata:00000000 C ResetComboID
rdata:00000016 C ToClient_ResetComboID
rdata:00000020 C ComboHitBar
rdata:00000018 C ComboIndex_ForDamageSteps
rdata:00000009 C ComboId
rdata:0000001D C GetComboFinisherIconOverride
rdata:00000012 C CommonComboArrow
rdata:0000000C C ComboButton
rdata:00000009 C ComboBox
rdata:0000001D C MessageLogListingComboButton
rdata:00000015 C EditableComboBox_Add
rdata:00000018 C EditableComboBox_Delete
rdata:00000018 C EditableComboBox_Rename
rdata:00000018 C EditableComboBox_Accept
rdata:00000029 C ToolBar_ToolBarComboButtonLock.Padding
rdata:0000002E C ToolBar_ToolBarComboButtonLock.Padding
rdata:00000033 C ToolBar_ToolBarComboButtonLockComboButton.Color
rdata:00000026 C Menu_ToolBarComboButtonLock.Padding
rdata:0000002B C Menu_ToolBarComboButtonLock.Padding
rdata:00000030 C Menu_ToolBarComboButtonLockComboButton.Color
rdata:00000011 C ComboButtonStyle
rdata:0000000E C ComboBoxStyle
rdata:00000017 C MenuPlacement_ComboBox
rdata:0000001C C MenuPlacement_ComboBoxRight
rdata:00000000 C SComboButton
rdata:00000008 C E:/Jenkins/TimeGame_LiveBuild/workspace/Engine/Source/Runtime/Slat...
rdata:00000021 C SComboBox_CSharpPString >
rdata:00000006 C E:/Jenkins/TimeGame_LiveBuild/workspace/Engine/Source/Runtime...
rdata:0000000F C SComboListType
rdata:00000016 C SComboRowOptionType>
rdata:00000024 C MultiHitType: ToolBarComboButton
rdata:00000002 C E:/Jenkins/TimeGame_LiveBuild/workspace/Engine/Source/Runtime/UMG/...
rdata:00000014 C SComboBox(UObject)
rdata:00000008 C E:/Jenkins/TimeGame_LiveBuild/workspace/Engine/Source/Runtime/UMG/...

```

因此在确定可以通过特征码定位到 Gworld，并且游戏没有加密以后，可

以通过提取 NamePool 特征码通过一些 SDK Dump 工具进行 dump。

```
2861 char pad_250[0x3]; // 0x250(0x03)
2862
2863 void UpdateMinimapTexture(); // Function TimeGame.TG_LevelAnchor.UpdateMinimapTexture // (Final|Native|Public|BlueprintCallable) // @ game
2864 bool CaptureMinimapTexture(struct UWorld* World, struct UTexture2D*& out_Texture, struct FBox& out_LocalSpaceMapBounds); // Function
2865
2866 // Class TimeGame.TG_LocalPlayer
2867 // Size: 0x280 (Inherited: 0x250)
2868 struct UTG_LocalPlayer : UBT5_LocalPlayer {
2869     char pad_258[0x28]; // 0x258(0x28)
2870 };
2871
2872 // Class TimeGame.TG_MeleeWeaponBase
2873 // Size: 0x560 (Inherited: 0x2a8)
2874 struct ATG_MeleeWeaponBase : ABT5_MeleeWeaponBase {
2875     char pad_2A8[0x60]; // 0x2A8(0x60)
2876     struct UTG_SkillTreeComponent* MySkillTreeComp; // 0x300(0x00)
2877     struct FTGInventoryHandle MyInventoryData; // 0x310(0x08)
2878     struct UBT5_AbilitySystemComponent* AbilitySystemComponent; // 0x4B0(0x08)
2879     struct UTG_InventoryItemAttributeSet* AttributeSet; // 0x4C0(0x08)
2880     int32_t ComboIndex; // 0x4C8(0x04)
2881     int32_t ComboIndex_ForDmgSweeps; // 0x4D0(0x04)
2882     bool DamagedSomethingThisSwing; // 0x4D8(0x01)
2883     bool HitSomethingThisSwing; // 0x4E1(0x01)
2884     char pad_4D2[0x6]; // 0x4D2(0x06)
2885     struct USkeletalMeshComponent* SkeletalMeshComp; // 0x4D8(0x08)
2886     bool OnEquippedCalled; // 0x4E0(0x01)
2887     char pad_4E1[0x7]; // 0x4E1(0x07)
2888     struct FSweepParameters ActiveSweepParameters; // 0x4E8(0x68)
2889     struct TArray<struct UMeshComponent*> SecondaryMeshesToCleanup; // 0x550(0x10)
2890
2891     void ToClient_ResetComboID(bool AlsoResetDmgTypeIndex); // Function TimeGame.TG_MeleeWeaponBase.ToClient_ResetComboID // (Net|NetRelia
2892     void ResetComboID(bool AlsoResetDmgTypeIndex); // Function TimeGame.TG_MeleeWeaponBase.ResetComboID // (Final|Native|Public|Blueprint
2893     void OnRep_MyInventoryData(); // Function TimeGame.TG_MeleeWeaponBase.OnRep_MyInventoryData // (Native|Public) // @ game-0x1770200
2894     void K2_OnEquipped(struct ABT5_CharacterBase* Wearer); // Function TimeGame.TG_MeleeWeaponBase.K2_OnEquipped // (Event|Public|Blueprint
2895     bool IsSocketUnlocked(int32_t SocketIdx); // Function TimeGame.TG_MeleeWeaponBase.IsSocketUnlocked // (Final|Native|Public|BlueprintCa
2896     bool IsOnLastComboIdx(); // Function TimeGame.TG_MeleeWeaponBase.IsOnLastComboIdx // (Final|Native|Public|BlueprintCallable|BlueprintP
2897     void IncrementComboID(); // Function TimeGame.TG_MeleeWeaponBase.IncrementComboID // (Final|Native|Public|BlueprintCallable) // @ game
2898     struct FTGWeaponData GetWeaponData(); // Function TimeGame.TG_MeleeWeaponBase.GetWeaponData // (Final|Native|Public|BlueprintCallable)
2899     void GetTargetLockOnDistances(float& out_AcquiredDist, float& out_DesiredDist); // Function TimeGame.TG_MeleeWeaponBase.GetTargetLockOn
2900     int32_t GetNumPrimaryComboAnims(); // Function TimeGame.TG_MeleeWeaponBase.GetNumPrimaryComboAnims // (Final|Native|Public|BlueprintCa
2901     struct FTGInventoryHandle GetMyInventoryHandle(); // Function TimeGame.TG_MeleeWeaponBase.GetMyInventoryHandle // (Native|Public|Blue
```

在获取到游戏 SDK 后可以加速分析。

## 分析结论：

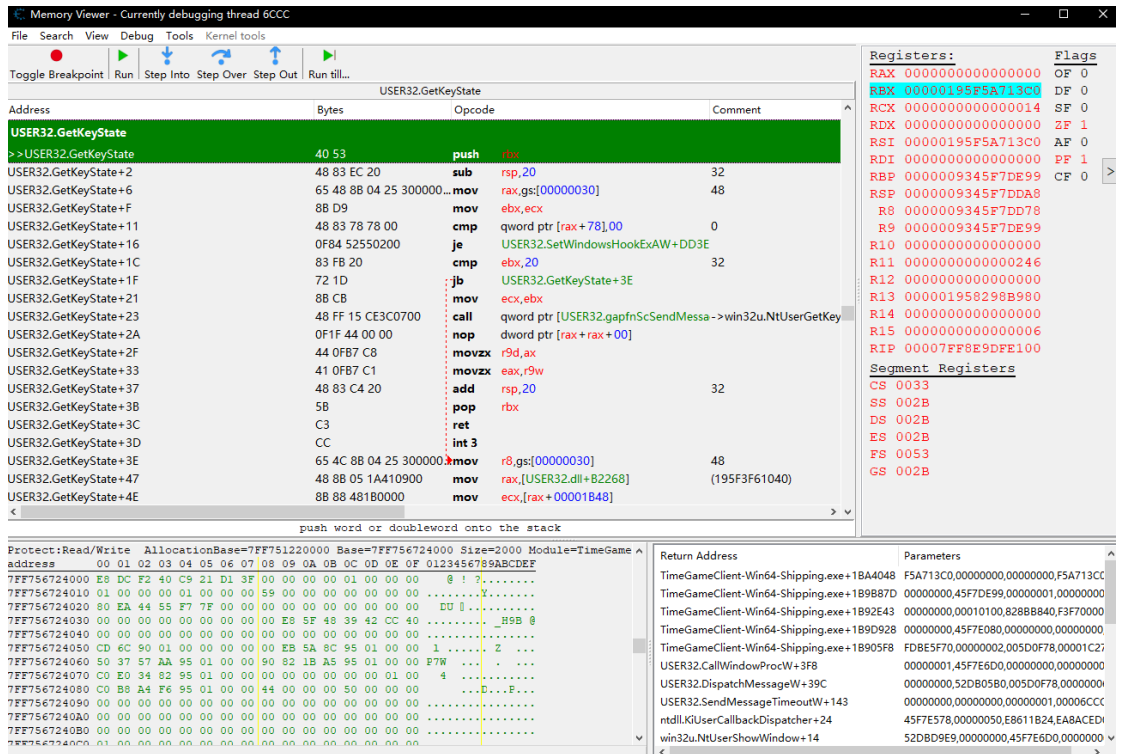
BigTime 在游戏代码保护方面得分为 0，毫无保护。在传统游戏中，往往会采用定制加密，加壳等方式对源码进行保护。由于 BigTime 并没有健全的游戏基础代码保护，导致恶意玩家分析代码的门槛与成本都很低，如果有外挂出现，对正常玩家来说是不公平的，且有一定可能会对游戏的经济模型造成影响。

## 游戏基础反作弊：

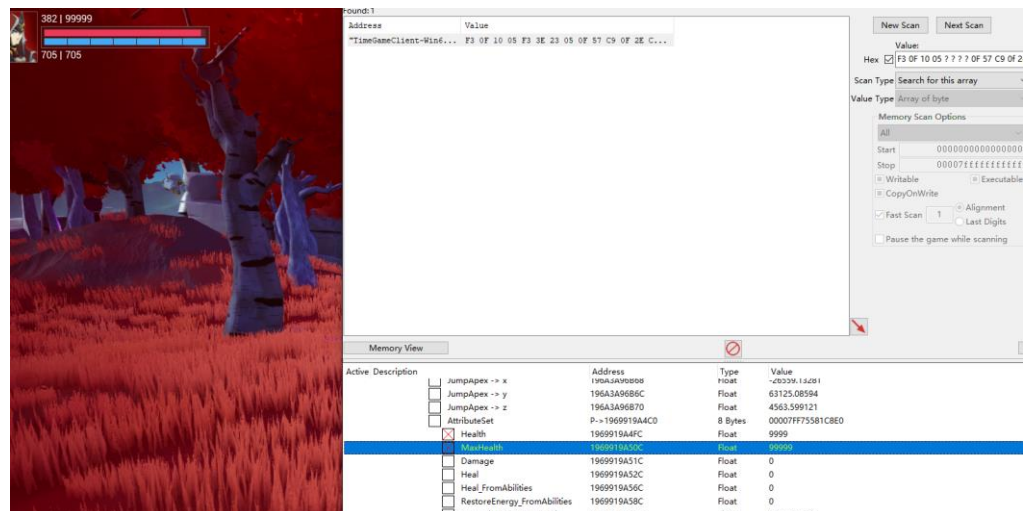
### 分析过程：

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



- 通过 CE 对修改游戏内的 Health 进行修改，发现可以生效并且游戏并没有进行弹窗或者提示。(修改 Health 仅是为了更直观的显示，该字段一般来说都是存储在服务器，本地修改时是没有任何效果的)



## 分析结论：

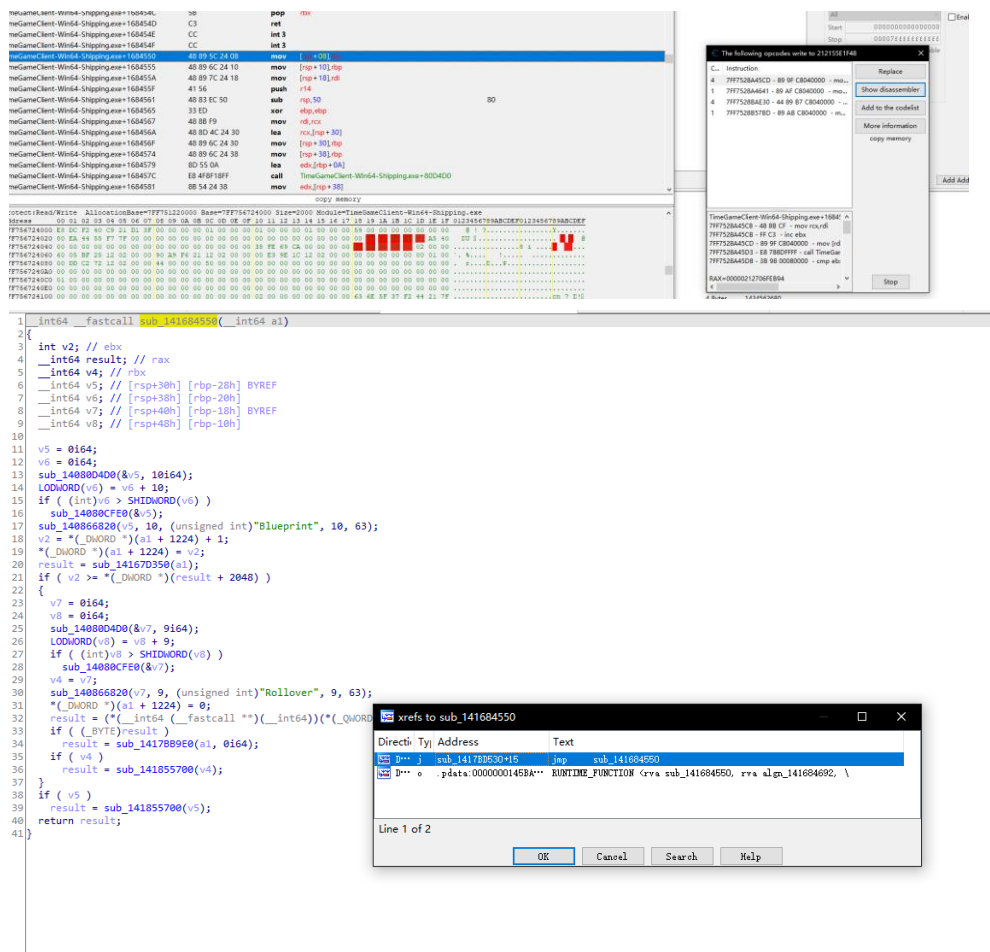
- BigTime 在反作弊能力方面得分为 0，如果存在恶意用户可以任意作弊。

- 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

### 分析过程：

对于基于 UE 开发的 MMO 类型的游戏来说，篡改本地数据的收益很低，原因是 UE 有成型的同步机制，用于各个 Pawn 之间属性同步以及服务端校验，但是通过分析游戏源码来看，很明显 BigTime 并没有将属性同步机制合理运用，还是有部分数据落地，例如 Comboindex 功能，通过对 combo Index 设置写入断点，可以找到写入函数，之后便可以对 combo 功能进行调试。（具体操作会影响公平性，不做演示）



```

1 fastcall sub_141684550(int64 a1)
2 {
3     int v2; // ebx
4     __int64 result; // rax
5     __int64 v4; // rbx
6     __int64 v5; // [rsp+30h] [rbp-20h] BYREF
7     __int64 v6; // [rsp+38h] [rbp-10h] BYREF
8     __int64 v7; // [rsp+40h] [rbp-18h] BYREF
9     __int64 v8; // [rsp+48h] [rbp-10h]
10
11     v5 = 0i64;
12     v6 = 0i64;
13     sub_14088040D(&v5, 10i64);
14     LODWORD(v6) = v6 + 10;
15     if ( (int)v6 > SHIDWORD(v6) )
16         sub_14088CFE0(&v5);
17     sub_140866820(v5, 10, (unsigned int)"Blueprint", 10, 63);
18     v2 = *(DWORD*)(a1 + 1224) + 1;
19     *(DWORD*)(a1 + 1224) = v2;
20     result = sub_14167D350(a1);
21     if ( v2 > *(DWORD*)(result + 2048) )
22     {
23         v7 = 0i64;
24         v8 = 0i64;
25         sub_14088040D(&v7, 9i64);
26         LODWORD(v8) = v8 + 9;
27         if ( (int)v8 > SHIDWORD(v8) )
28             sub_14088CFE0(&v7);
29         v4 = v7;
30         sub_140866820(v7, 9, (unsigned int)"Rollover", 9, 63);
31         *(DWORD*)(a1 + 1224) = 0;
32         result = *(__int64*)(fastcall)(__int64*)(*(DWORD*)(result + 2048));
33         if ( (BYTE)result )
34             result = sub_1417889E0(a1, 0i64);
35         if ( v4 )
36             result = sub_141855700(v4);
37     }
38     if ( v5 )
39         result = sub_141855700(v5);
40     return result;
41 }
  
```

The following operands write to 212121168

Instruction	Address	Disassembly
4	7F7528A6C0 - 89 8F CB40000	mov [rbp+10], rdi
5	7F7528A6C0 - 89 8F CB40000	mov [rbp+10], rdi
6	7F7528A6C0 - 89 8F CB40000	mov [rbp+10], rdi
7	7F7528A6C0 - 89 8F CB40000	mov [rbp+10], rdi

xrefs to sub\_141684550

Direct	Type	Address	Text
00000000	JMP	sub_1417889E0+15	jmp sub_141684550
00000000	CALL	pdata:0000000145BA...	CALL RAX, sub_141684550, rva:141684682, \



### 分析结论:

1. BigTime 的整体游戏逻辑安全问题并不是很突出，但是还是存在一定的安全风险，因次逻辑安全评分为 4 分。
2. 针对某些敏感属性缺少同步机制，更多的应该放到服务端加密。

## 游戏 RPC 分析

由于 RPC 问题较为敏感，再缺少项目方授权下暂不展开分析。目前 BigTime RPC 安全防护为 0，并且对于任意 RPC 包，服务器都会认可，其安全评分为 0。建议项目方针对 RPC 整体安全进行详细审计。下图为部分 RPC 信息。

[illegible]

## WEB3 安全分析：

### 概要：

Bigtime 作为一款链游，在 Web3 设计上可以分为两部分，分别是：基础 bigtime 代币部分，以及游戏内的 WEB3 经济系统部分。该部分设计相对其他游戏相对分离，游戏内负责产出代币与锻造 NFT，同时在 ETH 上部署一个固定流通的代币合约。

### 代币合约安全：

代币基础信息如下：

Address	0x64Bc2cA1Be492bE7185FAA2c8835d9b824c8a194
Symbol	BIGTIME
Owner	0xc3322716475fba83bfc057112247a43f1a1f2c4c( <b>GnosisSafe</b> )
TotalSupply	5,000,000,000

File 5 of 5 : BigTimeToken.sol

```
1 //SPDX-License-Identifier: Unlicense
2 pragma solidity 0.8.4;
3
4 import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
5
6 contract BigTimeToken is ERC20 {
7     constructor(
8         string memory _name,
9         string memory _symbol,
10        uint _totalSupply,
11        address _owner
12    ) ERC20(_name, _symbol) {
13        _mint(_owner, _totalSupply * 10 ** decimals());
14    }
15 }
```

BigTime 代币合约采用向多签名钱包 Mint 代币，然后采用固定供应量的方式进行部署。因为当前代币合约功能简单，所以合约的基础安全性是足够的。通过观察 Owner 钱包的 Tx 信息，可以看到 Owner 钱包再获取到代币以后向几个钱包分别转帐了部分代币。

在 BigTime 中玩家可以进入时空守卫的空间进行锻造时间沙漏，时间沙漏充能等操作，这部分可以直接影响市场平衡的功能有部分代码是存放在本地执行的，虽然不清楚 GS 是如何设计的，但是这种行为属于高危行为。如下

---

---

## 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。