Off The Grid Security Analysis

2024.07.30

Senna

DAMOCLES LABS

Contents

OTG (Off The Grid) is a 3A-level battle royale GameFi FPS game. The game utilizes the Easy Anti-Cheat (EAC) solution for anti-cheat measures, which provides basic protection during gameplay. However, since EAC is a free anti-cheat engine, bypassing its protection is not complex. Additionally, the game lacks customized anti-cheat measures for cheat functionalities, resulting in a low detection rate for cheating players. With cheats being sold in the market, the overall security rating of the game is assessed at 3 out of 5.

Security Rating: ★★★☆☆

Game Background

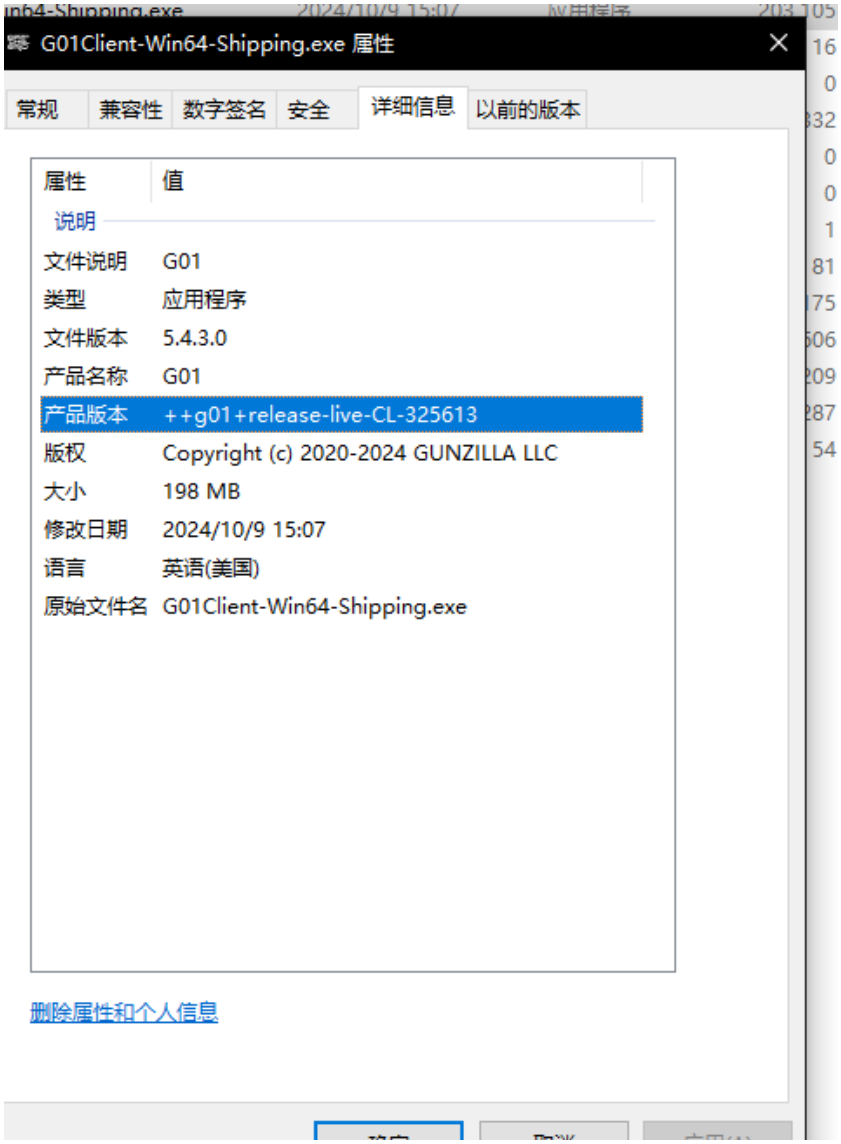
- GGame Version Assessed: ++g01+release-live-CL-325613
- Game Type & Game Engine: FPS, UE 5.4.3
- Potential Gameplay Issues:
  - Aimbot
  - Wallhacks
  - No recoil
  - Bullet tracking
  - Speed hacks
  - Custom scripts with hidden protocol vulnerabilities

Game Security Analysis

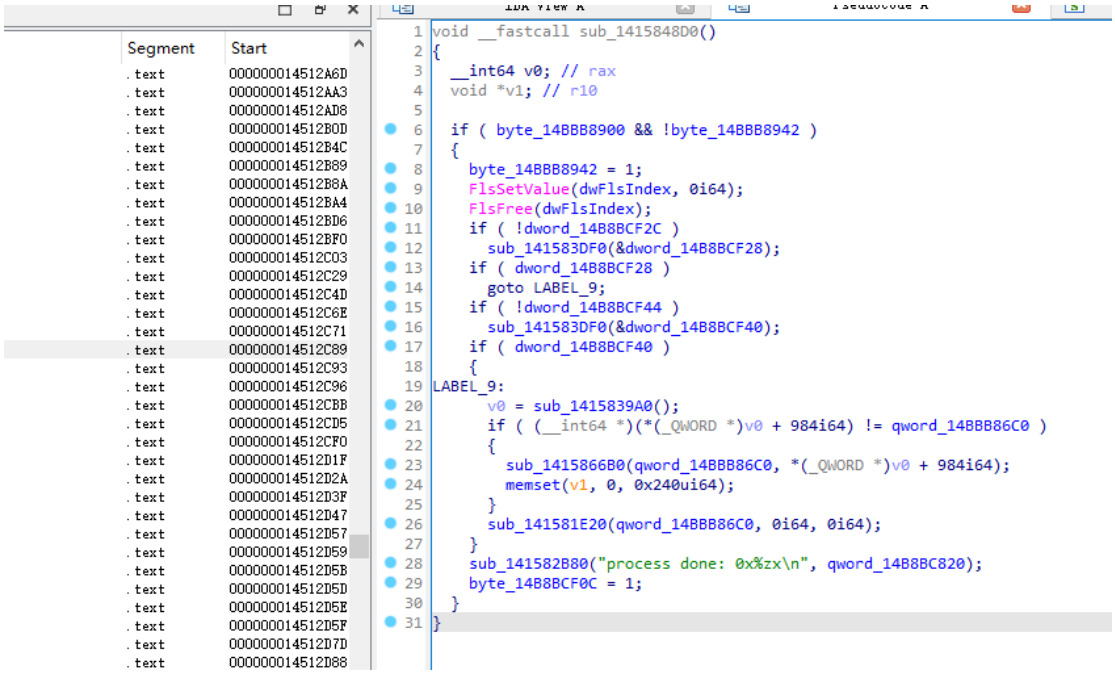
Game Code Protection：

Analysis Process：

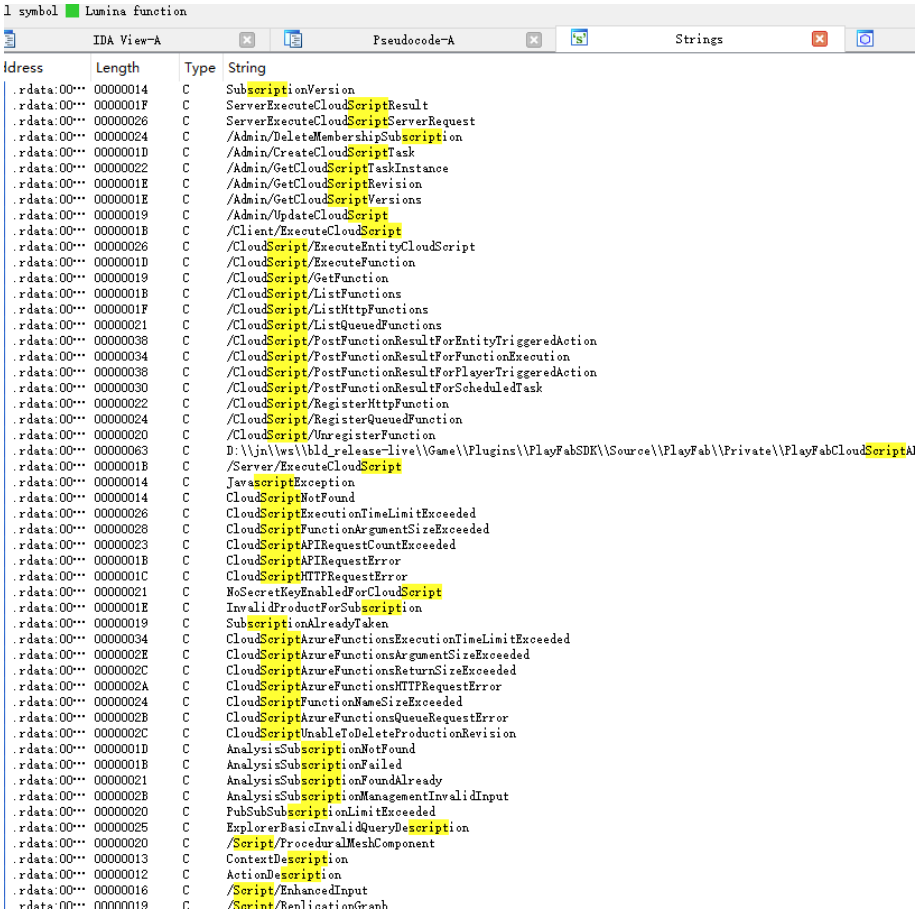
1. Determine the game engine by analyzing the game EXE since different engines have different analysis modes. Based on the identification of basic game information, we can confirm that Unity is used for game development.



2. Using IDA for decompilation, we found that the code was not encrypted and the strings were not encrypted.



```
void __fastcall sub_1415848D0()
{
  __int64 v0; // rax
  void *v1; // r10

  if ( byte_14BBB8900 && !byte_14BBB8942 )
  {
    byte_14BBB8942 = 1;
    FlsSetValue(dwFlsIndex, 0i64);
    FlsFree(dwFlsIndex);
    if ( !dword_14B8BCF2C )
      sub_141583DF0(&dword_14B8BCF28);
    if ( dword_14B8BCF28 )
      goto LABEL_9;
    if ( !dword_14B8BCF44 )
      sub_141583DF0(&dword_14B8BCF40);
    if ( dword_14B8BCF40 )
    {
LABEL_9:
      v0 = sub_1415839A0();
      if ( (__int64 *)(*(_QWORD *)v0 + 984i64) != qword_14BBB86C0 )
      {
        sub_1415866B0(qword_14BBB86C0, *(_QWORD *)v0 + 984i64);
        memset(v1, 0, 0x240ui64);
      }
      sub_141581E20(qword_14BBB86C0, 0i64, 0i64);
    }
    sub_141582B80("process done: 0x%zx\n", qword_14B8BC820);
    byte_14B8BCF0C = 1;
  }
}
```



```
.rdata:00··· 00000014 C SubscriptionVersion
.rdata:00··· 0000001F C ServerExecuteCloudScriptResult
.rdata:00··· 00000026 C ServerExecuteCloudScriptServerRequest
.rdata:00··· 00000024 C /Admin/DeleteMembershipSubscription
.rdata:00··· 0000001D C /Admin/CreateCloudScriptTask
.rdata:00··· 00000022 C /Admin/GetCloudScriptTaskInstance
.rdata:00··· 0000001E C /Admin/GetCloudScriptRevision
.rdata:00··· 0000001E C /Admin/GetCloudScriptVersions
.rdata:00··· 00000019 C /Admin/UpdateCloudScript
.rdata:00··· 0000001B C /Client/ExecuteCloudScript
.rdata:00··· 00000026 C /CloudScript/ExecuteEntityCloudScript
.rdata:00··· 0000001D C /CloudScript/ExecuteFunction
.rdata:00··· 00000019 C /CloudScript/GetFunction
.rdata:00··· 0000001B C /CloudScript/ListFunctions
.rdata:00··· 0000001F C /CloudScript/ListHttpFunctions
.rdata:00··· 00000021 C /CloudScript/ListQueuedFunctions
.rdata:00··· 00000038 C /CloudScript/PostFunctionResultForEntityTriggeredAction
.rdata:00··· 00000034 C /CloudScript/PostFunctionResultForFunctionExecution
.rdata:00··· 00000038 C /CloudScript/PostFunctionResultForPlayerTriggeredAction
.rdata:00··· 00000030 C /CloudScript/PostFunctionResultForScheduledTask
.rdata:00··· 00000022 C /CloudScript/RegisterHttpFunction
.rdata:00··· 00000024 C /CloudScript/RegisterQueuedFunction
.rdata:00··· 00000020 C /CloudScript/UnregisterFunction
.rdata:00··· 00000063 C D:\\jn\\ws\\bld_release-live\\Game\\Plugins\\PlayFabSDK\\Source\\PlayFab\\Private\\PlayFabCloudScriptAl
.rdata:00··· 0000001B C /Server/ExecuteCloudScript
.rdata:00··· 00000014 C JavascriptException
.rdata:00··· 00000014 C CloudScriptNotFound
.rdata:00··· 00000026 C CloudScriptExecutionTimeLimitExceeded
.rdata:00··· 00000028 C CloudScriptFunctionArgumentSizeExceeded
.rdata:00··· 00000023 C CloudScriptAPIRequestCountExceeded
.rdata:00··· 0000001B C CloudScriptAPIRequestError
.rdata:00··· 0000001C C CloudScriptHTTPRequestError
.rdata:00··· 00000021 C NoSecretKeyEnabledForCloudScript
.rdata:00··· 0000001E C InvalidProductForSubscription
.rdata:00··· 00000019 C SubscriptionAlreadyTaken
.rdata:00··· 00000034 C CloudScriptAzureFunctionsExecutionTimeLimitExceeded
.rdata:00··· 0000002E C CloudScriptAzureFunctionsArgumentSizeExceeded
.rdata:00··· 0000002C C CloudScriptAzureFunctionsReturnSizeExceeded
.rdata:00··· 0000002A C CloudScriptAzureFunctionsHTTPRequestError
.rdata:00··· 00000024 C CloudScriptFunctionNameSizeExceeded
.rdata:00··· 0000002B C CloudScriptAzureFunctionsQueueRequestError
.rdata:00··· 0000002C C CloudScriptUnableToDeleteProductionRevision
.rdata:00··· 0000001D C AnalysisSubscriptionNotFound
.rdata:00··· 0000001B C AnalysisSubscriptionFailed
.rdata:00··· 00000021 C AnalysisSubscriptionFoundAlready
.rdata:00··· 0000002B C AnalysisSubscriptionManagementInvalidInput
.rdata:00··· 00000020 C PubSubSubscriptionLimitExceeded
.rdata:00··· 00000025 C ExplorerBasicInvalidQueryDescription
.rdata:00··· 00000020 C /Script/ProceduralMeshComponent
.rdata:00··· 00000013 C ContextDescription
.rdata:00··· 00000012 C ActionDescription
.rdata:00··· 00000016 C /Script/EnhancedInput
```

We can also use UE Dumper to dump data structures for quick analysis understanding of the game logic.

```
// Class G01.GzBaseCharacter
// 0x0150 (0x07D0 - 0x0680)
class AGzBaseCharacter : public ACharacter
{
public:
    uint8                                          Pad_2D2A[0x38];                                  // 0x0680(0x0038)(
    class UGzDamageableComponent*                  DamageableComponent;                             // 0x06B8(0x0008)(
    class UGzAbilitySystemComponent*               AbilitySystemComponent;                          // 0x06C0(0x0008)(
    class UDataTable*                              DefaultAttributesDT;                             // 0x06C8(0x0008)(
    TArray<TSubclassOf<class UGameplayEffect>>     StartupEffects;                                  // 0x06D0(0x0010)(
    class FName                                    CapsuleCollisionProfileName;                     // 0x06E0(0x0008)(
    class APlayerState*                            PersistentPlayerState;                           // 0x06E8(0x0008)(
    struct FGzNativeCharacterComponentSpec         AkComponentSpec;                                 // 0x06F0(0x0080)(
    class UGzCharacterAkComponent*                 AkComponent;                                     // 0x0770(0x0008)(
    class UGzZoneTrackingComponent*                ZoneTrackingComponent;                           // 0x0778(0x0008)(
    TArray<class UGzBPOnlyCharacterComponentSpec*> BPOnlyComponentSpecs;                             // 0x0780(0x0010)
    TSubclassOf<class UGzAITokenComponent>         AITokenComponentClass;                           // 0x0790(0x0008)(
    class UGzCombatComponent*                      CachedCombatComponent;                           // 0x0798(0x0008)(
    class UGzEnvironmentZoneManagerComponent*      EnvironmentZoneManager;                          // 0x07A0(0x0008)(
    class UGzInvComponent*                         InvComponent;                                    // 0x07A8(0x0008)(
    class UGzAITokenComponent*                     AITokenComponent;                                // 0x07B0(0x0008)(
    struct FLyraReplicatedAcceleration             ReplicatedAcceleration;                          // 0x07B8(0x0003)(
    uint8                                          Pad_2D2B[0x5];                                   // 0x07BB(0x0005)(
    FMulticastInlineDelegateProperty_              OnRepPlayerState;                                // 0x07C0(0x0010)(

public:
    class USceneComponent* GetBPComponent(TSubclassOf<class USceneComponent> ComponentClass, const class FName CompName
    void OnRep_PersistentPlayerState();
    void OnRep_ReplicatedAcceleration();
```

Thus, understanding the game logic can be expedited through data structures and code analysis.

Analysis Conclusion:

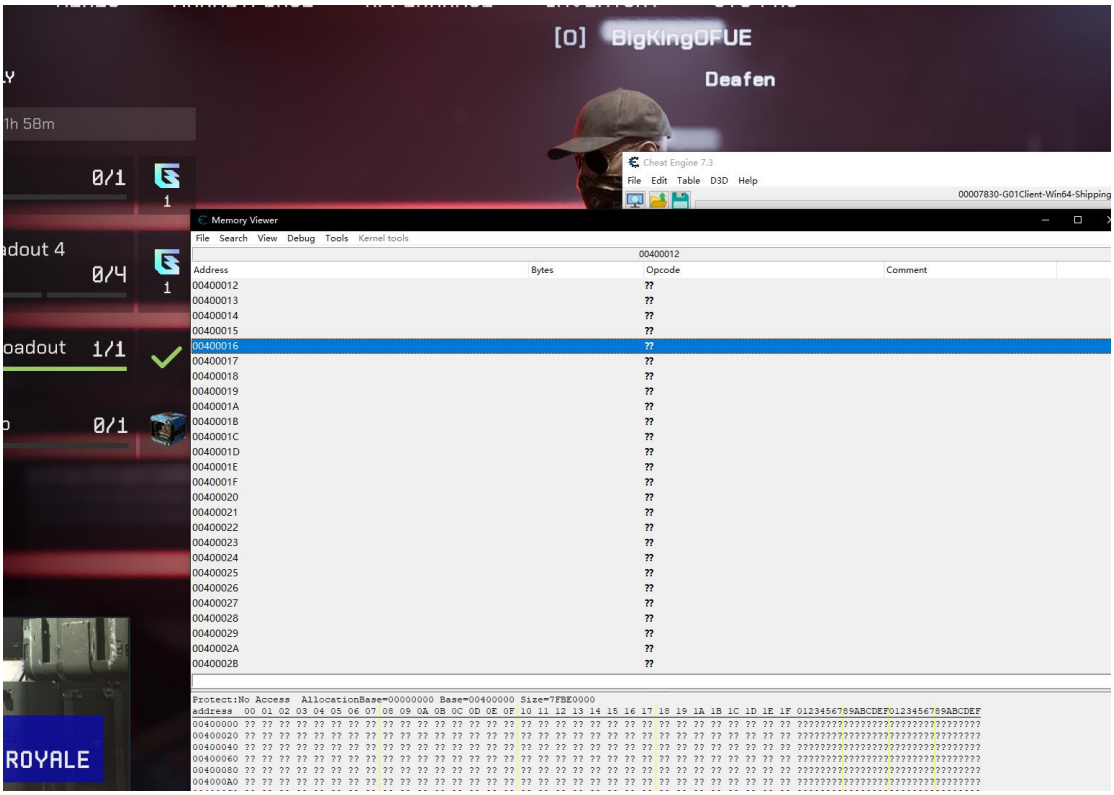
Conclusion: OTG scores 0 in game code protection as the client code and strings are not encrypted, enabling users to easily dump game data structures for quick analysis.

Fix Recommendations: Add local encryption for code and local protection for strings.

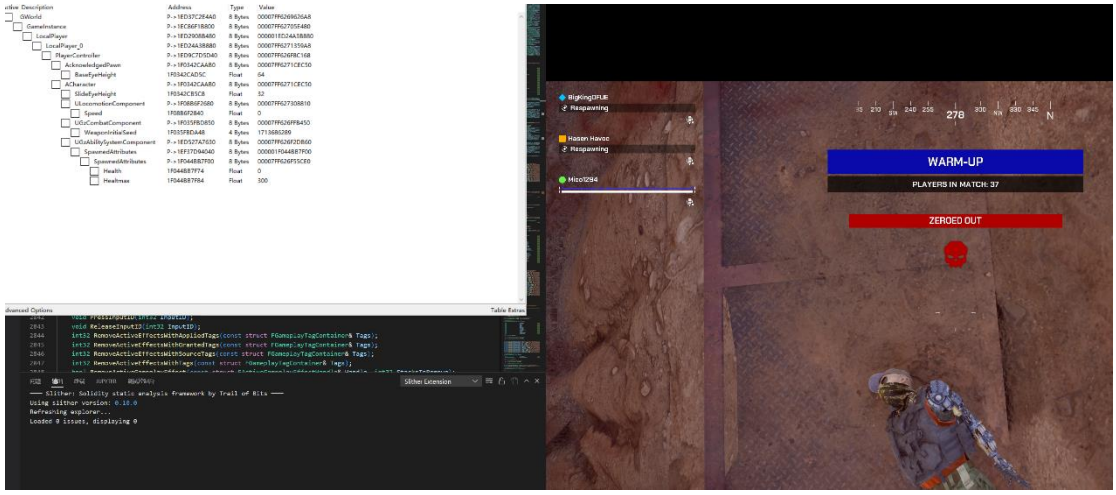
Game Basic Anti-Cheat:

Analysis Process:

1. In basic anti-cheat detection, testing primarily focused on whether the game has anti-debugging mechanisms and read/write protection..
2. While using CE for memory viewing in-game, it was noted that memory scanning was ineffective, indicating EAC functionality.



3. With specialized CE attachment, memory reads and writes were possible, allowing for code analysis in conjunction with dumped structures and IDA.



Analysis Conclusion:

1. OTG's basic anti-cheat measures score 3. By integrating EAC for game protection, some level of security enhancement is achieved, raising the technical threshold for analysis or attacks. However, experienced attackers can bypass EAC's process protection to modify game data undetected.
2. Testing focused on anti-debugging and read/write protection due to their critical role in cheat functionalities. Lack of these fundamental protections renders additional detections like injections and hooks ineffective.

Fix Recommendations: Implement functional measures and include sensitive data in synchronization frameworks.

Game Protocol & Logic Security Analysis

Analysis Process:

1. Analysis of game structures and code logic reveals that certain weapon attribute data is not synchronized to the server during weapon property synchronization, permitting local modification of attribute values for cheating purposes.

```
class AGzWeaponActor : public AActor
{
public:
    uint8                                          Pad_30B2[0x10];
    class AActor*                                  MagazineProp;
    struct FGzWeaponConstructionInfo               ConstructionInfo;
    class USkeletalMeshComponent*                  SkeletalMeshComponent;
    class UGzWeaponComponent*                      WeaponComponent;
    class UGzWeaponItemData*                       WeaponItemData;
    class UGzWeaponSkinItemData*                   WeaponSkinItemData;
    class UGzWeaponAttachmentComponent*            AttachmentComponents[0x8];
    class UAkComponent*                            AkComponent;
    TArray<struct FGzInventoryItemAttachmentContent> PendingAttachments;
    TArray<class UObject*>                         AttachmentModifierResources;
    TArray<class UGzWeaponBehaviorAttachment*>     BehaviorAttachments;
    uint8                                          Pad_30B3[0x20];

public:
    void OnInitBehaviorAttachments();
```

For instance, data like SpreadData, SwayData, RecoilData influences shot trajectory and hit detection logic

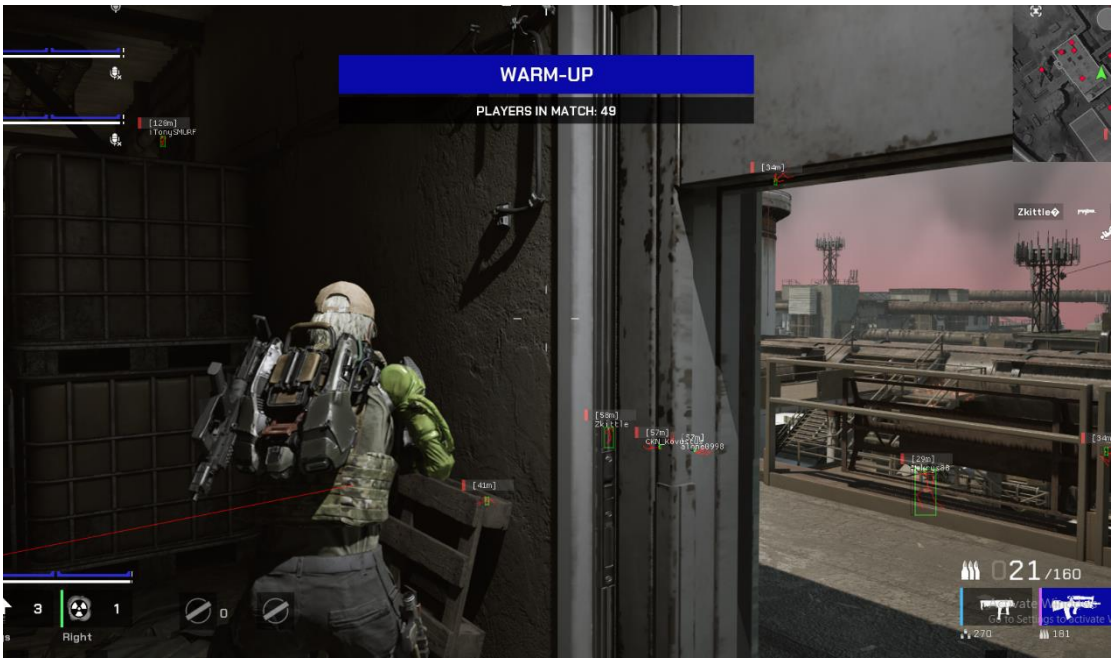
2. The game employs Azure PlayFab Game Server solution for game server operations, with on-chain operations including Decode Hex functionality carried out via Cloud Script in the GS. Due to PlayFab's reliance on RestAPI, project owners must securely manage APP:title permissions to prevent misuse or risks like game data deletion due to Secret Key breaches

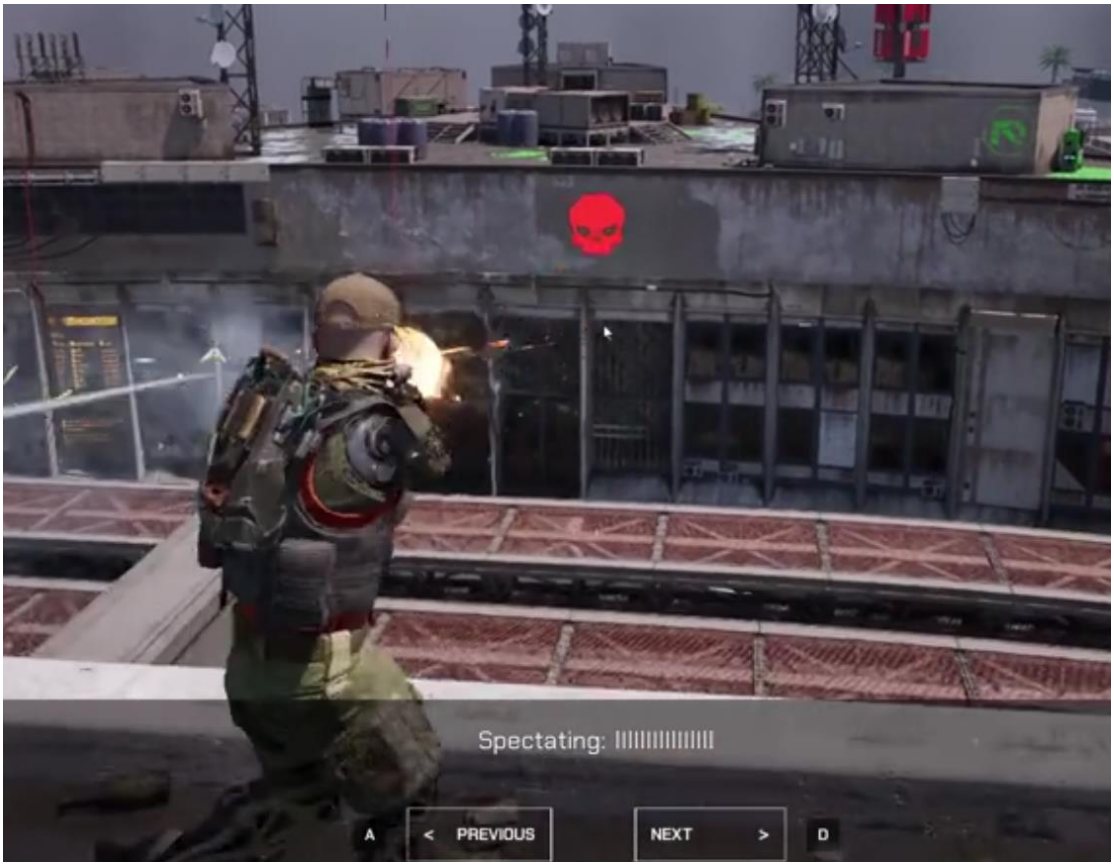
Damocles


```
POST https://1E5FA.playfabapi.com/CloudScript/ExecuteFunction?sdk=UE4MKPL-1.120.230707 HTTP/1.1
Accept: */*
Accept-Encoding: deflate, gzip
Content-Length: 141
Content-Type: application/json; charset=utf-8
Host: 1E5FA.playfabapi.com
User-Agent: GO1/++g01+release-live-CL-325613 (http-legacy) Windows/10.0.19045.1.256.64bit
X-EntityToken:
[illegible]
X-PlayFabSDK: UE4MKPL-1.120.230707

{"FunctionName":"GetPrimaryBalance","FunctionParameter":{"sessionId":"f0273993-57b4-4e9c-bc67-7502b1368a88","v":325613,"featureSwitches":{}}}
```

```
HTTP/1.1 200 OK
Cache-Control: no-cache, no-store, must-revalidate
Content-Length: 179
Content-Type: application/json
Expires: 0
Pragma: no-cache
access-control-allow-credentials: true
access-control-allow-headers: Content-Type, Content-Encoding, X-Authentication, X-Authorization, X-PlayFabSDK, X-ReportErrorAsSuccess, X-SecretKey, X-EntityToken, Authorization, x-ms-app,
x-ms-client-request-id, x-ms-user-id, traceparent, tracestate, Request-Id
access-control-allow-methods: GET, POST
access-control-allow-origin: *
date: Wed, 16 Oct 2024 14:47:31 GMT
server: istio-envoy
vary: Accept-Encoding
x-envoy-upstream-service-time: 56
x-requestid: 585ce7782aaf49bc8c8a9191f35fffd56
x-tracecontext-traceid: 6ff4caca506051ef7499bbeb95e4a69a

{"code":200,"status":"OK","data":{"ExecutionTimeMilliseconds":8,"FunctionName":"GetPrimaryBalance","FunctionResult":{"Code":1000,"BackendVersion":250797,"BalanceString":"10.26"}}}
```


3. Also found that there are already some cheating





Analysis Conclusion:

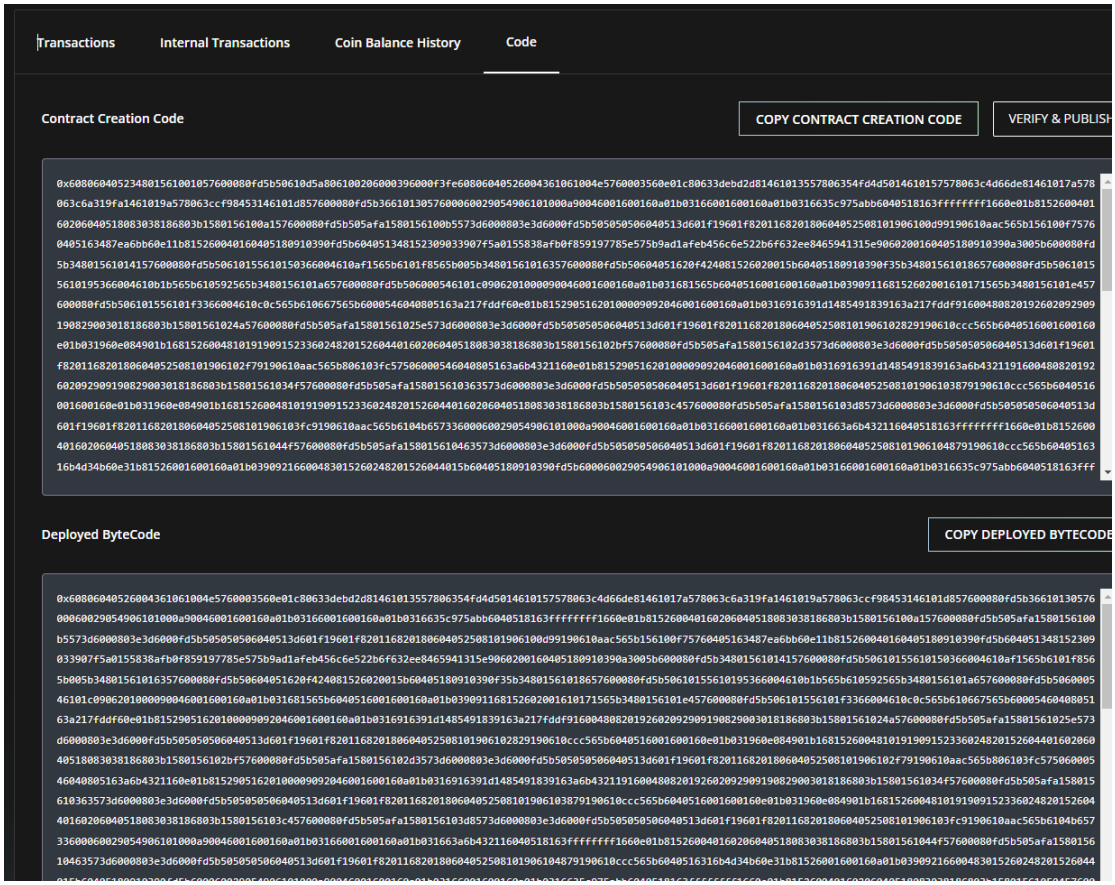
1. The current logic poses risks due to unsynchronized data rendering clients vulnerable to malicious cheat functionalities. Additionally, the opaque nature

of on-chain interaction scripts hinders risk assessment. It is recommended that project owners rigorously audit functional scripts and tightly control permission accounts.

Fix Recommendations: Enhance sensitive data synchronization, encrypt script interactions, and introduce functional measures.

WEB3 Security Analysis：

The current token contract code for OTG is not open-source, employing proxy contract methods.



The Decoder contract is open-source, and each call is proxied through Cloud

Script on the Game Server, mitigating risks. However, the primary asset risks lie in the security of proxy functions themselves

About Damocles

Damocles Labs is a security team established in 2023, specializing in security for the Web3 industry. Their services include contract code auditing, business code auditing, penetration testing, GameFi code auditing, GameFi vulnerability discovery, GameFi cheat analysis, and GameFi anti-cheat measures. They are committed to making continuous efforts in the Web3 security industry, producing as many analysis reports as possible, raising awareness among project owners and users about GameFi security, and promoting the overall security development of the industry.。

Twitter: https://twitter.com/DamoclesLabs

WebSite：http://damocleslabs.com/

Analysis Report repo：https://github.com/DamoclesLabs/GameFi-Analysis-Report/