Damocles

Cradles 游戏分析报告

2023.11.22

Senna

DAMOCLES LABS

Damocles

目录

Damocles

一、 概要

Cradles 于 11.15 日开放下载，Damocles 团队于 11.16 日对该游戏进行深入的安全分析，通过分析发现该游戏有大量 Debug 信息未删去，从 Debug 日志推断该游戏开发团队为中国团队。并且在测试过程中发现，该游戏并未进行任何的安全保护，且游戏通信协议部分使用开源引擎，并且对于有些逻辑判断过于不严谨，不推荐用户去游玩体体验。

安全评分：★ ☆ ☆ ☆ ☆

二、 游戏背景

- 进行评估的游戏版本：20231115
- 游戏类型&游戏引擎：MMORPG，Unity2021.3.x
- 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 框架下的时间函数）
  - 自瞄/自动锁定
  - 无敌
  - 无限体力
  - 挖矿加速

三、 游戏安全性分析

游戏代码保护：

分析过程：

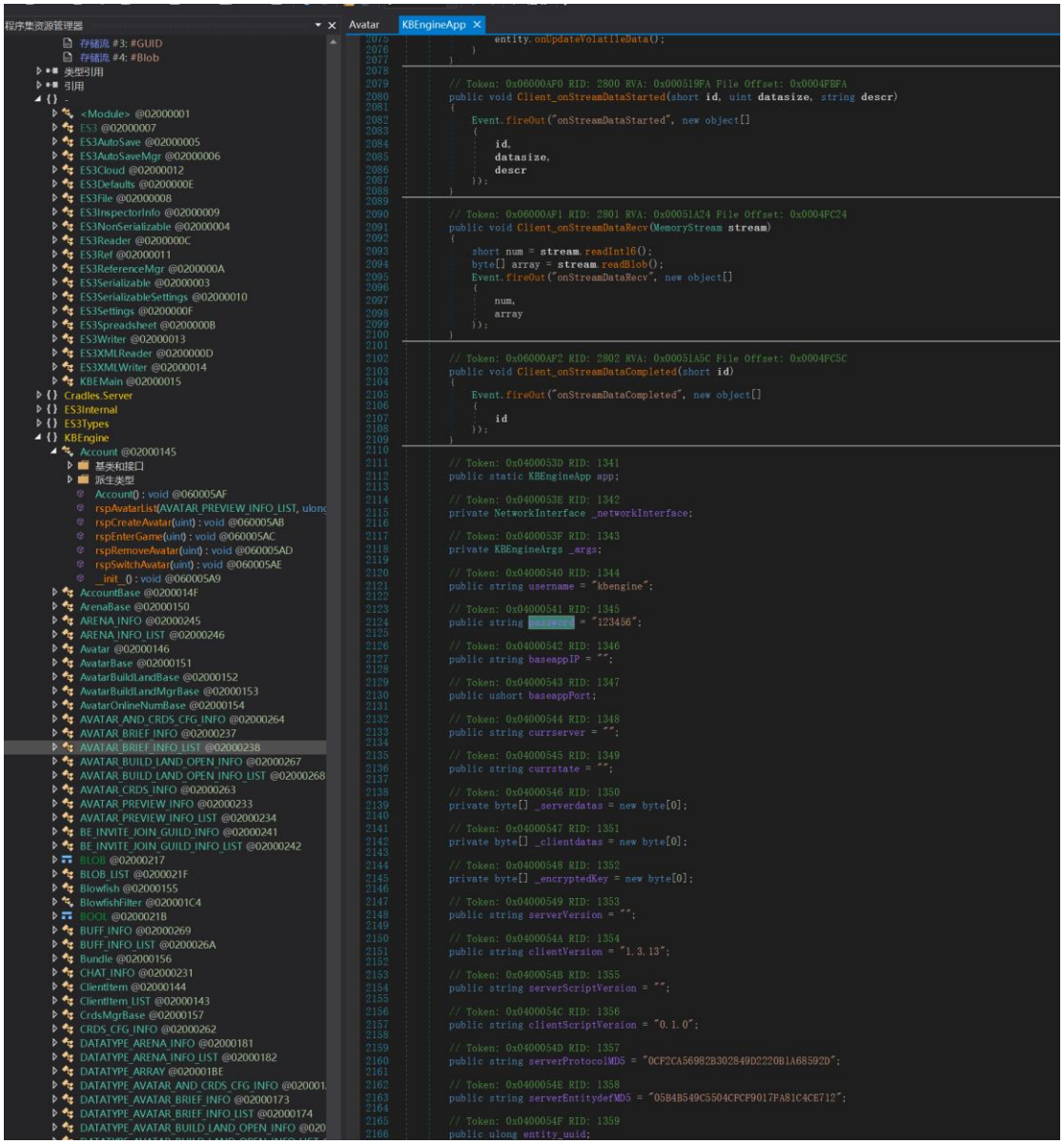
1. 由于不同的引擎有不同的分析模式，所以在获取到游戏 EXE 后首先需要确定游戏使用的引擎，通过对游戏基础信息识别我们可以确定该游戏是使用 Unity21.3.x 进行开发。

2. 通过浏览游戏释放的文件，可以确定游戏采用 Mono 机制，并不是采用 iL2Cpp 的模式进行开发。采用这种方式开发的游戏，整体的安全性会更差，分析更简单。

Damocles


stamina
return this.m_bag;
// Token: 0x1700001D RID: 29
// (get) Token: 0x060005B1 RID: 1457 RVA: 0x000264F1 File Offset: 0x000246F1
public Dictionary<int, ClientItem> WareHouse
get
return this.m_warehouse;
// Token: 0x060005B2 RID: 1458 RVA: 0x000264F9 File Offset: 0x000246F9
public override void __init__()
if (KBEngineApp.app.player().id != this.id)
return;
this._isPlayer = true;
// Token: 0x060005B3 RID: 1459 RVA: 0x0002651A File Offset: 0x0002471A
public int Currency(Avatar.CurrencyType type)
if (this.m_money.ContainsKey((int)type))
return (int)this.m_money[(int)type].num;
return 0;
// Token: 0x060005B4 RID: 1460 RVA: 0x0002653E File Offset: 0x0002473E
public MONEY_INFO CurrencyInfo(Avatar.CurrencyType type)
if (this.m_money.ContainsKey((int)type))
return this.m_money[(int)type];
return null;
// Token: 0x060005B5 RID: 1461 RVA: 0x0002655C File Offset: 0x0002475C
public override void ntfError(uint errcode)
Event.fireOut("OnServerErrcode", new object[]
errcode
});
Debug.Log(">>>>>>>>>OnServerErrcode!!!!! :" + errcode.ToString());
// Token: 0x060005B6 RID: 1462 RVA: 0x00026590 File Offset: 0x00024790
public override void ntfTeamMemberSyncInfo(TEAM_MEMBER_SYNC_INFO arg)
if (this.teamMemberState.Count == 0)
this.teamMemberState.Add(arg);
for (int i = 0; i < this.teamMemberState.Count; i++)
if (arg.dbid == this.teamMemberState[i].dbid)
this.teamMemberState[i] = arg;
else if (i >= this.teamMemberState.Count - 1)
this.teamMemberState.Add(arg);
Debug.Log("队伍成员状态返回");
Debug.Log(arg);
Event.fireOut("updateTeamMemberInfo", new object[]
""
});
// Token: 0x060005B7 RID: 1463 RVA: 0x00026638 File Offset: 0x00024838
public override void onNameChanged(string oldValue)
Event.fireOut("OnNameChanged", new object[]
this
});
// Token: 0x060005B8 RID: 1464 RVA: 0x0002664E File Offset: 0x0002484E
public override void rspExecDuelFieldNextState(uint arg)
Debug.Log("决斗场状态");
Debug.Log(arg);
// Token: 0x060005B9 RID: 1465 RVA: 0x00026665 File Offset: 0x00024865
public override void rspChangeGuildIconID(uint a)
00 %
分析器
KBEngine.KBEngineApp.password : string @04000541
KBEngine.KBEngineApp.createAccount(string, string, byte[]) : void @06000A99
ClientApp.Register(string, string, byte[]) : void @06000149
KBEngine.AvatarBase.hp : uint @0400044F
读取于
Cradles.Character.FolkControl.DoTakeDamage(int, Transform) : void @06001077
Cradles.Character.FolkControl.DoUpdate() : void @0600107A
Cradles.Character.HumanControl.DoSetAnimator() : void @06001095
Cradles.Character.HumanControl.DoStates() : void @06001093
Cradles.Character.HumanControl.SummonPetProcess(string) : IEnumerator @06001097
Cradles.Character.PlayerControl.DoTakeDamage(int, Transform) : void @060010D5
Cradles.Character.PlayerControl.DoUpdate() : void @060010D9
Cradles.Character.PlayerControl.StartCollecting() : IEnumerator @060010DE
Cradles.Character.PlayerControl.StartInteracting() : IEnumerator @060010E2
Cradles.Character.PlayerControl.StartLooting() : IEnumerator @060010E3

并且该游戏使用的协议框架是 KBEngine。



因此可以通过 Github 等开源库获取到 KBEngine 的源码，以及一些公开的资料，之后便可以加快游戏分析的速度。

Damocles

kbengine_unity3d_plugins / KBEngine.cs

Code Blame 3059 lines (2439 loc) · 84.9 KB Code 55% faster with GitHub Copilot

```
25          public class KBEngineApp
296             public virtual void process()
305                 // 向服务端发送心跳以及同步角色信息到服务端
306                 sendTick();
307             }
308
309             /*
310                 当前玩家entity
311             */
312             public Entity player()
313             {
314                 Entity e;
315                 if(entities.TryGetValue(entity_id, out e))
316                     return e;
317
318                 return null;
319             }
320
321             public void _closeNetwork(NetworkInterface networkInterface)
322             {
323                 networkInterface.close();
324             }
325
326             /*
327                 向服务端发送心跳以及同步角色信息到服务端
328             */
329             public void sendTick()
330             {
331                 if(_networkInterface == null || _networkInterface.connected == false)
332                     return;
333
334                 if(!loginappMessageImported_ && !baseappMessageImported_)
335                     return;
336
337                 TimeSpan span = DateTime.Now - _lastTickTime;
338
339                 // 更新玩家的位置与朝向到服务端
340                 updatePlayerToServer();
341
342                 if(_args.serverHeartbeatTick > 0 && span.Seconds > _args.serverHeartbeatTick)
343                 {
344                     span = _lastTickCBTime - _lastTickTime;
345
346                     // 如果心跳回调接收时间小于心跳发送时间，说明没有收到回调
347                     // 此时应该通知客户端掉线了
348                     if(span.Seconds < 0)
349                     {
350                         Dbg.ERROR_MSG("sendTick: Receive appTick timeout!");
351                         _networkInterface.close();
352                         return;
```

分析结论：

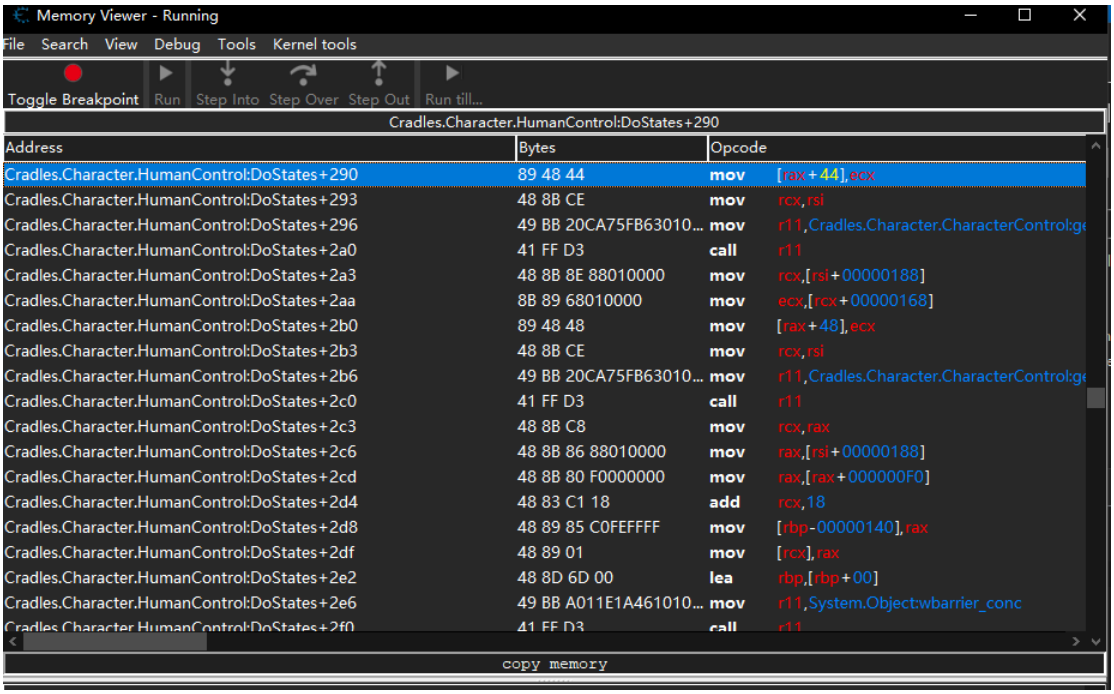
Cradles 在游戏代码保护方面得分为 0，毫无保护。在传统游戏中，往往会采用定制加密，加壳等方式对源码进行保护,并且传统游戏也很少会采用 mono 模式进行编译。由于 Cradles 并没有健全的游戏基础代码保护并且采用过时的编译技术，导致恶意玩家分析代码的门槛与成本都很低，如果有外挂出现，对正常玩家来是极度不公平，在玩家可以自由决斗的区域，作恶玩家更容易打败对手。

Damocles

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. 通过 CE 对修改游戏内的 stamina 和 HP 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 stamina 是实现无线体力/蓝量，HP 锁定可以在 10s 以内有效)

分析结论：

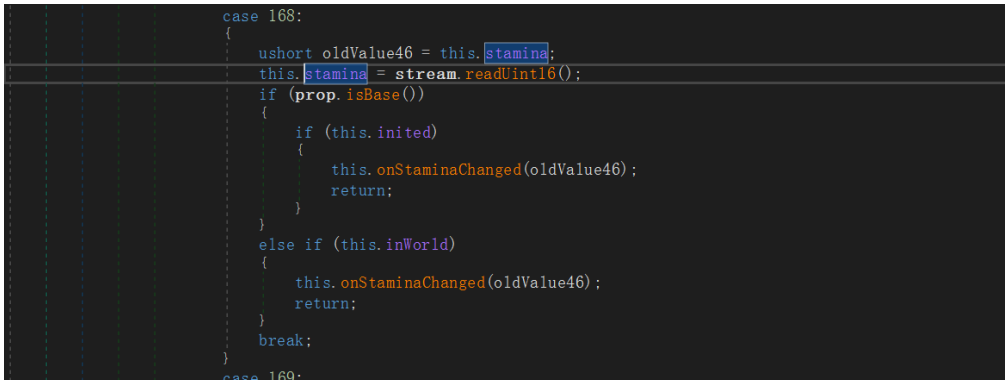
1. Cradles 在反作弊能力方面得分为 0，如果存在恶意用户可以任意作弊。
2. 只测试反调试和读写保护两个方面的原因是对于一块外挂来说，找数据与实现功能只需要通过调试和读写就可以实现。如果最基础的两个保护能力都缺失的话，那么一些注入、hook 等检测也毫无意义。

游戏逻辑问题

分析过程：

对于采用 mono 方式编译的 MMORPG 游戏来说，直接修改数据原则上是收益很低，但是在我们测试中发现，对于一些数据，如血量、体力等修改是可以生效的，其中血量修改后在 9s 内有效，超过时间则会无法攻击怪物，猜测该点在服务器有伤害时间限制。体力修改则可以长时间生效，猜测该点在服务器并未做任何判断，理由是：在本地角色体力耗尽时，可以通过暂停运动进行体力恢复操作，如果本地可以恢复，那么对于游戏来说则可以省去服务器校验的步骤。

体力更新逻辑：



```
case 168:
{
    ushort oldValue46 = this.stamina;
    this.stamina = stream.readUint16();
    if (prop.isBase())
    {
        if (this.inited)
        {
            this.onStaminaChanged(oldValue46);
            return;
        }
    }
    else if (this.inWorld)
    {
        this.onStaminaChanged(oldValue46);
        return;
    }
    break;
}
case 169:
```

血量更新逻辑：

```
case 132:
{
    uint oldValue33 = this.hp;
    this.hp = stream.readUint32();
    if (prop.isBase())
    {
        if (this.inited)
        {
            this.onHpChanged(oldValue33);
            return;
        }
    }
    else if (this.inWorld)
    {
        this.onHpChanged(oldValue33);
        return;
    }
    break;
}
```

并且在 Avatar 类中有很多与人物相关的属性，这部分属性中应该还存在其他可以操作的点。

Damocles

```
public ushort minPoisonDmg;

// Token: 0x04000467 RID: 1127
public sbyte moveH;

// Token: 0x04000468 RID: 1128
public sbyte moveLevel;

// Token: 0x04000469 RID: 1129
public sbyte moveV;

// Token: 0x0400046A RID: 1130
public string name = "";

// Token: 0x0400046B RID: 1131
public uint netDelayMs;

// Token: 0x0400046C RID: 1132
public uint nextSwtichRedNameTime;

// Token: 0x0400046D RID: 1133
public ushort perlOsRecoverStamina;

// Token: 0x0400046E RID: 1134
public ushort poisonResist;

// Token: 0x0400046F RID: 1135
public byte proficiencyTalentPoint;

// Token: 0x04000470 RID: 1136
public ushort robotUID;

// Token: 0x04000471 RID: 1137
public short sceneGroupID;

// Token: 0x04000472 RID: 1138
public uint sceneID;

// Token: 0x04000473 RID: 1139
public uint serverTime;

// Token: 0x04000474 RID: 1140
public byte sex;

// Token: 0x04000475 RID: 1141
public short speedIncPct;

// Token: 0x04000476 RID: 1142
public ushort stamina = 100;

// Token: 0x04000477 RID: 1143
public byte subState;

// Token: 0x04000478 RID: 1144
public int tavernRefTaskID;

// Token: 0x04000479 RID: 1145
public uint teamID;

// Token: 0x0400047A RID: 1146
public int vigor = -1;

// Token: 0x0400047B RID: 1147
public float walletHasCrdsNum;
```

分析结论：

1. Cradles 的整体游戏逻辑安全问题很严重，尤其是该游戏是涉及强制 PVP 模式的，外挂研发门槛低，收益高，在研发出成型外挂后，完全可以实现单方面虐杀。
2. 缺乏对游戏数据的感知，以及对游戏内其他易受攻击的点的检测，同时由于使用的是开源引擎，其协议属于完全开放状态，对于存在挖矿的游戏来说，这种行为的风险性极高。

Damocles

游戏协议分析

Cradles 采用的是 KBEngine 引擎作为协议基础，关于该引擎网络上有现成的资料可供参考。

参考资料：

1、KBEngine 技术总览

2、KBEngine MMORPG Demo

3、KBEngine unity3d plugins

Damocles

WEB3 安全分析：

由于目前 Cradles 的代币并未上线，所以 WEB3 方面的分析暂缓，并且由于挖矿相关的协议完全暴露，用户挖矿数量对于游戏来说只是暂存的数字而已，所以该部分并不展开分析

```
{
    // Token: 0x02000263 RID: 611
    public class AVATAR_CRDS_INFO
    {
        // Token: 0x04000652 RID: 1618
        public string name = "";

        // Token: 0x04000653 RID: 1619
        public string accountName = "";

        // Token: 0x04000654 RID: 1620
        public ulong avatarDBID;

        // Token: 0x04000655 RID: 1621
        public string crdsWalletAddr = "";

        // Token: 0x04000656 RID: 1622
        public byte level;

        // Token: 0x04000657 RID: 1623
        public float walletHasCrdsNum;

        // Token: 0x04000658 RID: 1624
        public float exchangeCrdsNum;

        // Token: 0x04000659 RID: 1625
        public uint crdsOreNum;

        // Token: 0x0400065A RID: 1626
        public uint cradlesPower;

        // Token: 0x0400065B RID: 1627
        public uint cpItemNum;

        // Token: 0x0400065C RID: 1628
        public string loginToken = "";
    }
}
```

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。