

スマホゲームセキュリティ

IL2CPPは本当に安全なの？

自己紹介

- nevermoe
 - 趣味：水泳、スキー、引きこもり
 - 履歴：

目次

- スマホゲームの攻撃と防御
- IL2CPP紹介
- IDA Plugin
- まとめ

スマホゲームの攻撃と防御 (Cheating)

- ネット通信改ざん
 - Replay
 - Request or Response改ざん
- ローカル改ざん
 - メモリチート
 - スピードハック
 - ファイル改ざん
 - バイナリ改ざん
 - フック



通信改ざん



ローカル改ざん

スマホゲームの攻撃と防御 (Anti-cheating)

- ネット通信改ざん
 - Replay → Token追加
 - Request or Response改ざん → 通信暗号化/SSL Pinning
- ローカル改ざん
 - メモリチート → メモリ暗号化
 - スピードハック → ローカル検知 / サーバertimestamp
 - ファイル改ざん → ローカルDB暗号化 / 改ざん検知
 - バイナリ改ざん → パッキング/難読化 / 改ざん検知
 - フック → アンチデバッグ / フックフックフレームワーク
検知 / インジェクション検知

スマホゲームの攻撃と防御 (ゲームエンジン)

- **Unity3D**

- 2Dでも、3Dでも
- Cross Platform
- sourceの一部公開



圧倒的にシェアが高い

- **Cocos2d-x**

- 2Dゲームに特化
- Cross Platform
- Open Source

- **FlashAIR/Unreal/Corona等**

- **自作**

スマホゲームの攻撃と防御

(Unity3D Anti-cheating現状)

- Unity3D Anti-cheating Toolkit
 - メモリ暗号化
 - スピードハック検知
 - 等

=>バイナリ自体リバーシングされたら終わり

スマホゲームの攻撃と防御 (目標)

* バイナリ自体リバーシングされたら終わり

- iOS -> IL2CPP

- 取得は簡単

- パッキング、難読化一切なし

=> シンボルのないアセンブラをリバーシング？

- Android -> Mono

- パッキング

- 難読化

=> 難読かされたコードをリバーシング？

=>アセンブラを読もう！

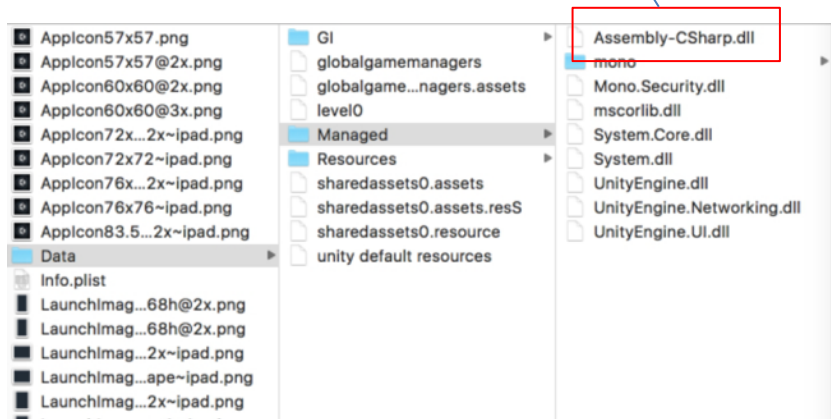
IL2CPP紹介

(Without IL2CPP / Scripting Backend: Mono)

- Android



- iOS



```
namespace MiniJSON
{
    public static class Json
    {
        public static object Deserialize(string json)
        {
            if (json == null)
            {
                return null;
            }
            return Json.Parser.Parse(json);
        }

        public static string Serialize(object obj)
        {
            return Json.Serializer.Serialize(obj);
        }
    }

    private sealed class Parser : IDisposable
    {
        private const string WORD_BREAK = "{}[],:~\\";

        private StringReader json;

        private char NextChar
        {
            get
            {
                return Convert.ToChar(this.json.Read());
            }
        }

        private Json.Parser.TOKEN NextToken
        {
            get
            {
                string nextWord;
                Dictionary<string, int> str;
                int num;
                this.EatWhitespace();
                if (this.json.Peek() == -1)
                {
                    return TOKEN.None;
                }
                char ch = NextChar;
                if (ch == '{')
                {
                    return TOKEN.Object;
                }
                if (ch == '[')
                {
                    return TOKEN.Array;
                }
                if (ch == ',')
                {
                    return TOKEN.Comma;
                }
                if (ch == ':')
                {
                    return TOKEN.Colon;
                }
                if (ch == '~')
                {
                    return TOKEN.Null;
                }
                if (ch == '\\')
                {
                    return TOKEN.Escape;
                }
                if (char.IsDigit(ch))
                {
                    return TOKEN.Number;
                }
                if (char.IsLetter(ch))
                {
                    return TOKEN.String;
                }
                return TOKEN.None;
            }
        }

        private void EatWhitespace()
        {
            while (NextChar == ' ')
            {
                json.Read();
            }
        }

        private void ParseObject()
        {
            TOKEN token = NextToken;
            if (token == TOKEN.None)
            {
                return;
            }
            if (token == TOKEN.Object)
            {
                object obj = new Dictionary<string, object>();
                while (true)
                {
                    token = NextToken;
                    if (token == TOKEN.None)
                    {
                        return;
                    }
                    if (token == TOKEN.String)
                    {
                        string key = ParseString();
                        token = NextToken;
                        if (token == TOKEN.Colon)
                        {
                            token = NextToken;
                            obj[key] = Parse();
                        }
                    }
                    else if (token == TOKEN.Comma)
                    {
                        token = NextToken;
                    }
                    else if (token == TOKEN.None)
                    {
                        return;
                    }
                }
            }
            else if (token == TOKEN.Array)
            {
                object obj = new List<object>();
                while (true)
                {
                    token = NextToken;
                    if (token == TOKEN.None)
                    {
                        return;
                    }
                    if (token == TOKEN.String)
                    {
                        string key = ParseString();
                        token = NextToken;
                        if (token == TOKEN.Colon)
                        {
                            token = NextToken;
                            obj[key] = Parse();
                        }
                    }
                    else if (token == TOKEN.Comma)
                    {
                        token = NextToken;
                    }
                    else if (token == TOKEN.None)
                    {
                        return;
                    }
                }
            }
        }

        private string ParseString()
        {
            string str = "";
            while (true)
            {
                char ch = NextChar;
                if (ch == '\\')
                {
                    char next = NextChar;
                    if (next == 'n')
                    {
                        str += "\n";
                    }
                    else if (next == 'r')
                    {
                        str += "\r";
                    }
                    else if (next == 't')
                    {
                        str += "\t";
                    }
                    else if (next == 'b')
                    {
                        str += "\b";
                    }
                    else if (next == 'f')
                    {
                        str += "\f";
                    }
                    else if (next == '"')
                    {
                        str += "\"";
                    }
                    else if (next == '\\')
                    {
                        str += "\\";
                    }
                    else
                    {
                        str += next;
                    }
                }
                else if (ch == '"')
                {
                    return str;
                }
                else
                {
                    str += ch;
                }
            }
        }

        private object Parse()
        {
            TOKEN token = NextToken;
            if (token == TOKEN.None)
            {
                return null;
            }
            if (token == TOKEN.String)
            {
                return ParseString();
            }
            if (token == TOKEN.Number)
            {
                return ParseNumber();
            }
            if (token == TOKEN.Object)
            {
                return ParseObject();
            }
            if (token == TOKEN.Array)
            {
                return ParseArray();
            }
            if (token == TOKEN.Null)
            {
                return null;
            }
            if (token == TOKEN.Escape)
            {
                char ch = NextChar;
                if (ch == 'n')
                {
                    return "\n";
                }
                if (ch == 'r')
                {
                    return "\r";
                }
                if (ch == 't')
                {
                    return "\t";
                }
                if (ch == 'b')
                {
                    return "\b";
                }
                if (ch == 'f')
                {
                    return "\f";
                }
                if (ch == '"')
                {
                    return "\"";
                }
                if (ch == '\\')
                {
                    return "\\";
                }
                return ch;
            }
            return null;
        }

        private object ParseArray()
        {
            object obj = new List<object>();
            while (true)
            {
                token = NextToken;
                if (token == TOKEN.None)
                {
                    return null;
                }
                if (token == TOKEN.String)
                {
                    string key = ParseString();
                    token = NextToken;
                    if (token == TOKEN.Colon)
                    {
                        token = NextToken;
                        obj[key] = Parse();
                    }
                }
                else if (token == TOKEN.Comma)
                {
                    token = NextToken;
                }
                else if (token == TOKEN.None)
                {
                    return null;
                }
            }
            return obj;
        }

        private object ParseNumber()
        {
            string str = "";
            while (true)
            {
                char ch = NextChar;
                if (char.IsDigit(ch))
                {
                    str += ch;
                }
                else if (ch == '.')
                {
                    str += ch;
                }
                else if (ch == '-')
                {
                    str += ch;
                }
                else if (ch == 'e')
                {
                    str += ch;
                }
                else if (ch == 'E')
                {
                    str += ch;
                }
                else if (ch == ' ')
                {
                    return ParseNumber();
                }
                else
                {
                    return null;
                }
            }
            return str;
        }

        private void Dispose()
        {
            json.Dispose();
        }
    }
}

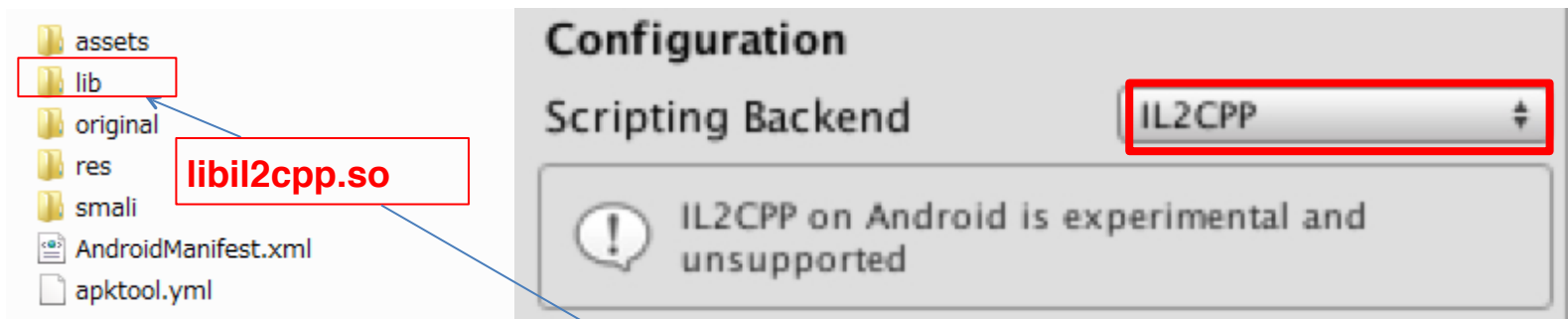
private static class Serializer
{
    private static string Serialize(object obj)
    {
        return Json.Serializer.Serialize(obj);
    }
}

private static class Parser
{
    private static object Parse(string json)
    {
        return Json.Parser.Parse(json);
    }
}
```

IL2CPP紹介

(With IL2CPP / Scripting Backend: IL2CPP)

- Android



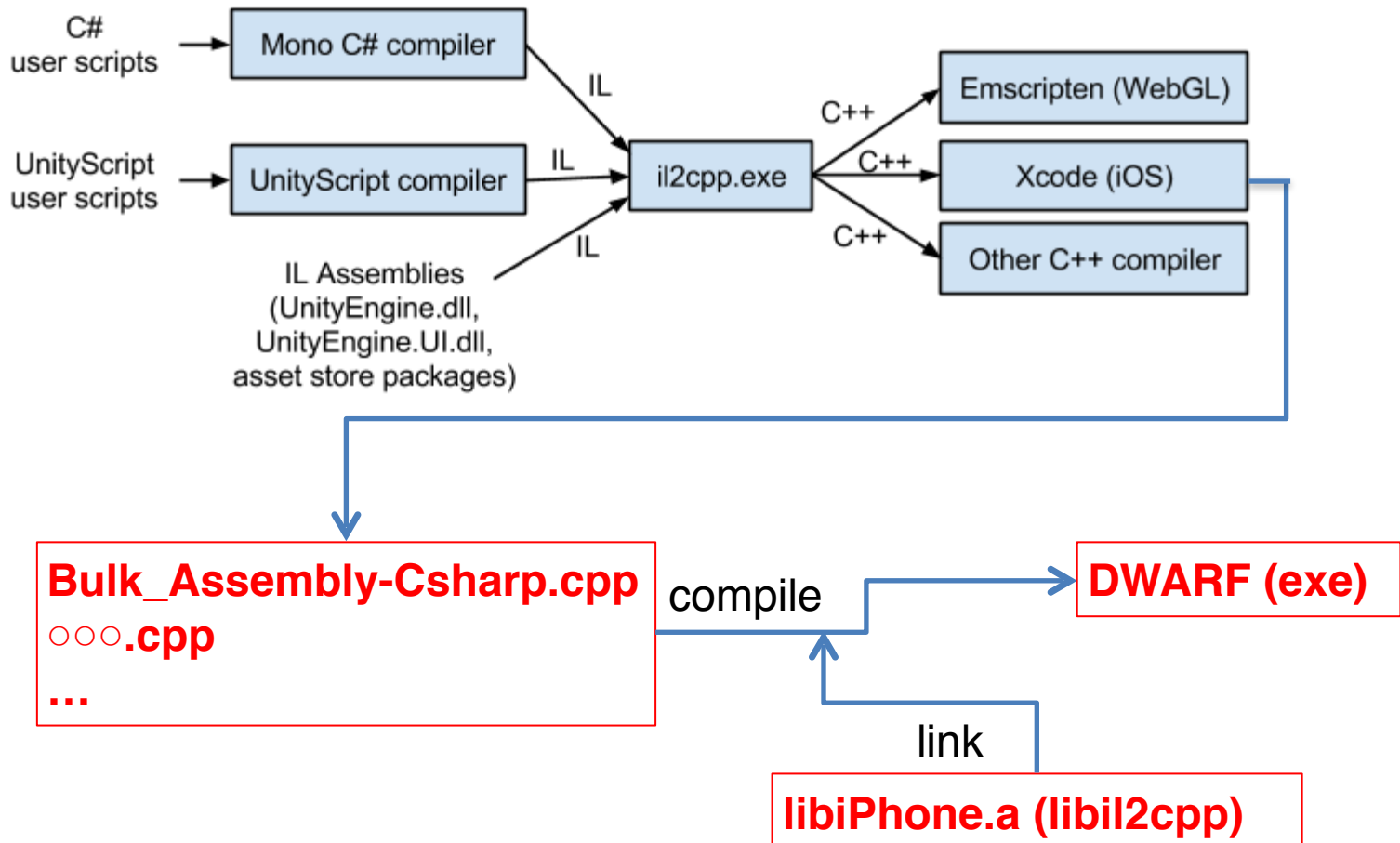
- iOS



```
LDR      X8, [SP,#arg_10]
STR      X8, [SP,#arg_A8]
ADRP    X8, #dword_103753684@PAGE
NOP
LDR      W9, [SP,#arg_1C]
STR      W9, [SP,#arg_B0]
LDR      W8, [X8,#dword_103753684@PAGEOFF]
MOV      W9, #1
MADD    W8, W8, W8, W9
MOV      W9, #0x24920000
MOVK    W9, #0x4925
MADD    X9, X8, X9, XZR
UBFM    X9, X9, #0x20, #0x3F
SUB     W10, W8, W9
ADD     W9, W9, W10,LSR#1
UBFM    W10, W9, #2, #0x1F
UBFM    W10, W10, #0x1D, #0x1C
SUB     W9, W10, W9,LSR#2
CMP     W8, W9
MOV     W8, #0x16
MOV     W9, #0x19
```

IL2CPP紹介

(流れ)



IL2CPP紹介

(libil2cppの役割)

- libil2cpp (libiPhone.a等)
 - Support VM
 - Garbage Collection
 - Metadata Loader

IL2CPP紹介 (Metadata)

void EnemyAttack(...)

```
IL_0045:  
{  
    PlayerHealth_t1138339563 * L_7 = __this->get_playerHealth_6();  
    NullCheck(L_7);  
    int32_t L_8 = L_7->get_currentHealth_3();  
    if (((int32_t)L_8) > ((int32_t)0))  
    {  
        goto IL_0066;  
    }  
}  
  
Animator_t2776330603 * L_9 = __this->get_anim_4();  
NullCheck(L_9);  
Animator_SetTrigger_m514363822(L_9, _stringLiteral4088827653, /*hidden argument*/NULL);  
}
```

```
tDistance.m_EffectColor.  
m_EffectDistance.m_UseGr  
aphicAlpha.effectColor.e  
ffectDistance.useGraphic  
Alpha.Assembly-CSharp.As  
sembly-CSharp.dll.EnemyA  
ttack.OnTriggerEnter.OnT  
riggerExit.Attack.timeBe  
tweenAttacks.attackDamag  
e.anim.playerHealth.play  
erInRange.EnemyHealth.am  
ount.hitPoint.TakeDamage  
.Death.StartSinking.star
```

```
ordering.Internal error.  
Trying to destroy objec  
t that is already releas  
ed to pool.Mesh can not  
have more than 65000 ver  
ticesPlayerPlayerDeadwhe  
reisthelog?DeadSpawnGame  
OverScore: DieShootableF  
ire1_name This is not po  
ssible to be called for  
standalone input. Please
```

Data/Managed/Metadata/global-metadata.dat

IL2CPP紹介

(解析手法)

- 解析手法
 - libiPhone.aを分析
 - 必ずついてる
 - しかもsymbolつき
 - libil2cppのsourceを分析
 - Windows Store Appをビルドすればsourceがついてくる

IL2CPP紹介

(Metadataローディング)

- Global-metadata.datをメモリに読み込み、メモリごとに構造体に変換する

```
//MetadataCache.cpp
void MetadataCache::Initialize()
{
    s_GlobalMetadata = vm::MetadataLoader::LoadMetadataFile ("global-metadata.dat");
    s_GlobalMetadataHeader = (const Il2CppGlobalMetadataHeader*)s_GlobalMetadata;
    assert (s_GlobalMetadataHeader->sanity == 0xFAB11BAF);
    assert (s_GlobalMetadataHeader->version == 21);
    ...
}
```

IL2CPP紹介

(StringLiteralローディング)

//MetadataCache.cpp

```
void MetadataCache::InitializeMethodMetadata (uint32_t
index)
{
    .....
    for (uint32_t i = 0; i < count; i++)
    {
        .....
        switch (usage)
        {
            .....
            case kIl2CppMetadataUsageStringLiteral:
                *s Il2CppMetadataRegistration-
>metadataUsages[destinationIndex] =
GetStringLiteralFromIndex (decodedIndex);
                break;
            default:
                ....
        }
    }
}
```

//Il2CppMetadataUsage.cpp

```
extern void** const g_MetadataUsages[7877] =
{
    (void**)&Contraction_t1673853792_0_0_0_var,
    (void**)&Level2Map_t3322505726_0_0_0_var,
    (void**)&String_t_0_0_0_var,
    (void**)&TypedReference_t1025199857_0_0_0_var,
    (void**)&ArgIterator_t2628088752_0_0_0_var,
    (void**)&Void_t1841601450_0_0_0_var,
    ...
    ...
    ...
    (void**)&_stringLiteral2004437333,
    (void**)&_stringLiteral3025533088,
    (void**)&_stringLiteral3687436746,
    (void**)&_stringLiteral2779811765,
    (void**)&_stringLiteral273729679,
};
```



IL2CPP紹介

(MethodInfoローディング)

// Class.cpp

```
void SetupMethodsLocked (Il2CppClass *klass, const FastAutoLock& lock)
{
    ...
    for (MethodIndex index = start; index < end; ++index) {
        ...
        newMethod->name = MetadataCache::GetStringFromIndex (methodDefinition->nameIndex);
        ...
        newMethod->methodPointer = MetadataCache::GetMethodPointerFromIndex (methodDefinition->methodIndex);
        ...
    }
    ...
}
```

// MetadataCache.cpp

```
Il2CppMethodPointer
MetadataCache::GetMethodPointerFromIndex
(MethodIndex index)
{
    ...
    return s_Il2CppClassCodeRegistration-
    >methodPointers[index];
}
```

```
extern const Il2CppMethodPointer g_MethodPointers[16812] =
{
    Locale_GetText_m1894433032,
    Locale_GetText_m2553164138,
    SafeHandleZeroOrMinusOneIsInvalid_ctor_m3340306667,
    SafeHandleZeroOrMinusOneIsInvalid_get_IsInvalid_m2033528032,
    SafeWaitHandle_ctor_m1710231470,
    SafeWaitHandle_ReleaseHandle_m634725016,
    ...
    VignetteAndChromaticAberration_ctor_m3270745889,
    VolumeHandler_Start_m3226079559,
    VolumeHandler_SetVolume_m3613034220,
    VolumeHandler_OnDestroy_m1170460248,
    VolumeHandler_ctor_m818831955,
};
```



IDA Plugin (Mapping to IDA)

- iOS

__const:0000000101A73F00		DCQ qword_101D36DB0
__const:0000000101A73F08		DCQ qword_101D36DB8
__const:0000000101A73F10		DCQ qword_101D36DC0
__const:0000000101A73F18		DCQ qword_101D36DC8
__const:0000000101A73F20		DCQ qword_101D36DD0
__const:0000000101A73F28		DCQ qword_101D36DD8
__const:0000000101A73F30		DCQ qword_101D36DE0
__const:0000000101A73F38		DCQ qword_101D36DE8
__const:0000000101A73F40		DCQ qword_101D36DF0
__const:0000000101A73F48		DCQ qword_101D36DF8
__const:0000000101A73F50		DCQ qword_101D36E00
__const:0000000101A73F58		DCQ qword_101D36E08
__const:0000000101A73F60	off_101A73F60	DCQ sub_1007AB700
__const:0000000101A73F68		DCQ sub_1007AB708
__const:0000000101A73F70		DCQ sub_1007AB784
__const:0000000101A73F78		DCQ sub_1007AB814
__const:0000000101A73F80		DCQ sub_1007AB854
__const:0000000101A73F88		DCQ sub_1007AB8CC
__const:0000000101A73F90		DCQ sub_1007AB928
__const:0000000101A73F98		DCQ sub_1007AB94C
__const:0000000101A73FA0		DCQ sub_1007ABBA4
__const:0000000101A73FA8		DCQ sub_1007ABCC4
__const:0000000101A73FB0		DCQ sub_1007ABD28
__const:0000000101A73FB8		DCQ sub_1007ABD68
__const:0000000101A73FC0		DCQ sub_1007ABD70
__const:0000000101A73FC8		DCQ sub_1007ABDD4

String

Method

```
//Il2CppMetadataUsage.cpp
extern void** const g_MetadataUsages[7877] =
{
    (void*)&Contraction_t1673853792_0_0_0_var,
    (void*)&Level2Map_t3322505726_0_0_0_var,
    (void*)&String_t_0_0_0_var,
    (void*)&TypedReference_t1025199857_0_0_0_var,
    (void*)&ArgIterator_t2628088752_0_0_0_var,
    (void*)&Void_t1841601450_0_0_0_var,
    ...
    ...
    ...
    (void*)&_stringLiteral2004437333,
    (void*)&_stringLiteral3025533088,
    (void*)&_stringLiteral3687436746,
    (void*)&_stringLiteral2779811765,
    (void*)&_stringLiteral273729679,
};
; extern const Il2CppMethodPointer g_MethodPointers[16812] =
{
    Locale_GetText_m1954433032,
    Locale_GetText_m2553164138,
    SafeHandleZeroOrMinusOneIsInvalid__ctor_m3340306667,
    SafeHandleZeroOrMinusOneIsInvalid_get_IsInvalid_m2033528032,
    SafeWaitHandle__ctor_m1710231470,
    SafeWaitHandle_ReleaseHandle_m634725016,
    ...
    VignetteAndChromaticAberration__ctor_m3270745889,
    VolumeHandler_Start_m3226079559,
    VolumeHandler_SetVolume_m3613034220,
    VolumeHandler_OnDestroy_m1170460248,
    VolumeHandler__ctor_m818831955,
};
```

IDA plugin (Demo)

1. Plugin使用方法
2. Cheating Demo

IDA Plugin (注意事項)

- Unity Version 5.3.6以上
 - Metadata version 21であればOK
- iOS, Android : 自動ロード
- 他のplatform : 手動ロード
- AndroidのbinaryではIDAがうまく関数、xref認識してくれない場合が多い (iOSのバイナリでおすすめ)

まとめ

- CheatingとAnti-cheatingはdynamic
- IL2CPPだけに頼るAnti-cheatingはいけ
ない
- 対策：metadataの中の文字列を難読化する
- Source Code: [https://github.com/
nevermoe/unity_metadata_loader](https://github.com/nevermoe/unity_metadata_loader)

Thank you!