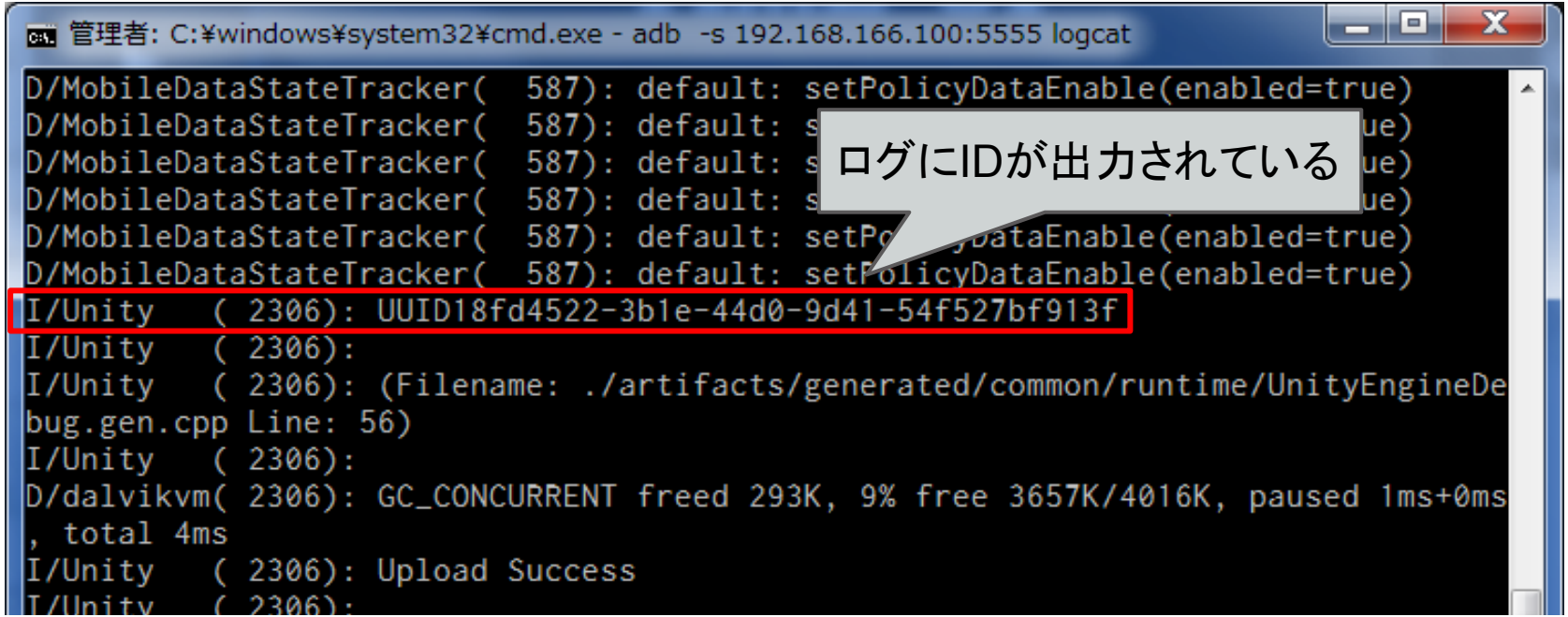
SECCON 2015 x CEDEC CHALLENGE
ゲームクラッキング＆チートチャレンジ
調査結果

vulscryptos

[1] sandbagの問題点

1. 不要なログ出力
2. アイテム不正入手
3. 任意のスコアデータの登録

[1]-1. 不要なログ出力　概要



[1]-1. 不要なログ出力　再現手法

難易度：簡単

Android SDKの入ったPCと、
USBデバッグモードが有効になったAndroidを
USBケーブルで繋ぎ、
PCで adb logcat を実行

[1]-1. 不要なログ出力　影響度

今回の"UUID"は個人情報とは紐付かない、
スコア送信用のユーザ識別IDなので、特に影響はない

しかし、
ログ出力から個人情報・パスワードがダダ漏れになっていた
……というケースも過去にあったため、細心の注意が必要

[1]-1. 不要なログ出力　対策

「ログ出力しても問題ない情報」と、
「ログ出力してはならない情報」とを吟味する

リリース時によく確認を！

[1] sandbagの問題点

1. 不要なログ出力
2. アイテム不正入手
   本来はプレイ回数に応じて入手できるボールが、
   プレイ回数に関係なく入手できてしまう
3. 任意のスコアデータの登録

[1]-2. アイテムの不正入手　問題点

```
# cat /data/data/net.sandbag2015/shared_prefs/net.sandbag2015.xml
<?xml version='1.0' encoding='utf-8' standalone='yes
<map>
    <int name="Screenmanager Is Fullscre
    <int name="Screenmanager Resolution
    <int name="scorePoint" value="305" />
    <string name="playerName">Charo</st
    <int name="Screenmanager Resolution Height" value="1920" />
    <int name="item" value="0" />
    <int name="playcount" value="1" />
    <string name="5.0.2">5bdc25aea3242a84c567ddb45a63b91e</string>
    <string name="uuid">18fd4522-3b1e-44d0-9d41-54f527bf913f</string>
</map>
```

アプリの設定値等を保存するファイル。
本来は自由にアクセスできないファイルだが、
改造(root化)した端末だと自由にアクセス可能

[1]-2. アイテムの不正入手　問題点

```
# cat /data/data/net.sandbag2015/shared_prefs/net.sandbag2015.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <int name="Screenmanager Is Fullscreen mode" value="0" />
    <int name="Screenmanager Resolution Width" va
    <int name="scorePoint" value="305" />
    <string name="playerName">Charo</string>
    <int name="Screenmanager Resolution Height" value="
    <int name="item" value="0" />
    <int name="playcount" value="1" />
    <string name="5.0.2">5bdc25aea3242a84c567ddb45a63b91e</string>
    <string name="uuid">18fd4522-3b1e-44d0-9d41-54f527bf913f</string>
</map>
```

プレイ回数・入手アイテムが
平文でファイルに書いてある

[1]-2. アイテムの不正入手　再現手法

難易度：root化さえできれば簡単

1. 端末をroot化する("端末の型番 root"でGoogle検索)
2. 以下のファイルを編集する
   /data/data/net.sandbag2015/shared_prefs/net.sandbag2015.xml

[1]-2. アイテムの不正入手　影響度

- アイテムの入手難易度が変わるため、
  ゲームバランスが崩壊
- ゲームの寿命を著しく縮める

- 仮に、個人情報が平文で保存されていた場合、
  悪意のあるアプリに読み取られて
  外部に送信される恐れがある

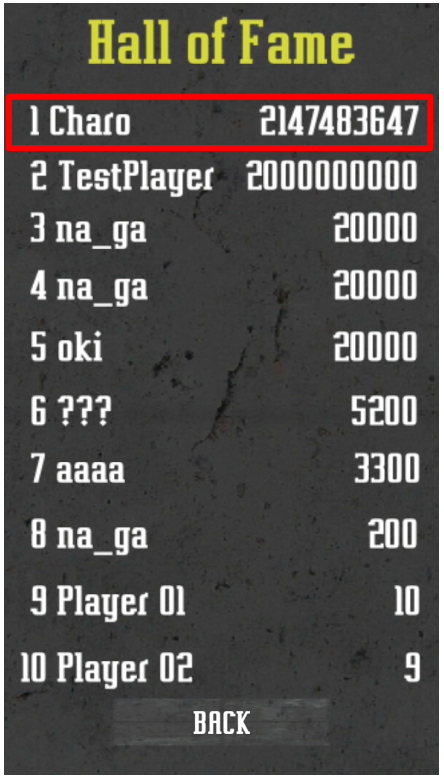
[1]-2. アイテムの不正入手　対策

- プレイ回数・入手アイテムはサーバ側で管理
  - 見せたくない・改ざんされたくない情報を
    端末に置かない
  - 効果は大きい一方で、

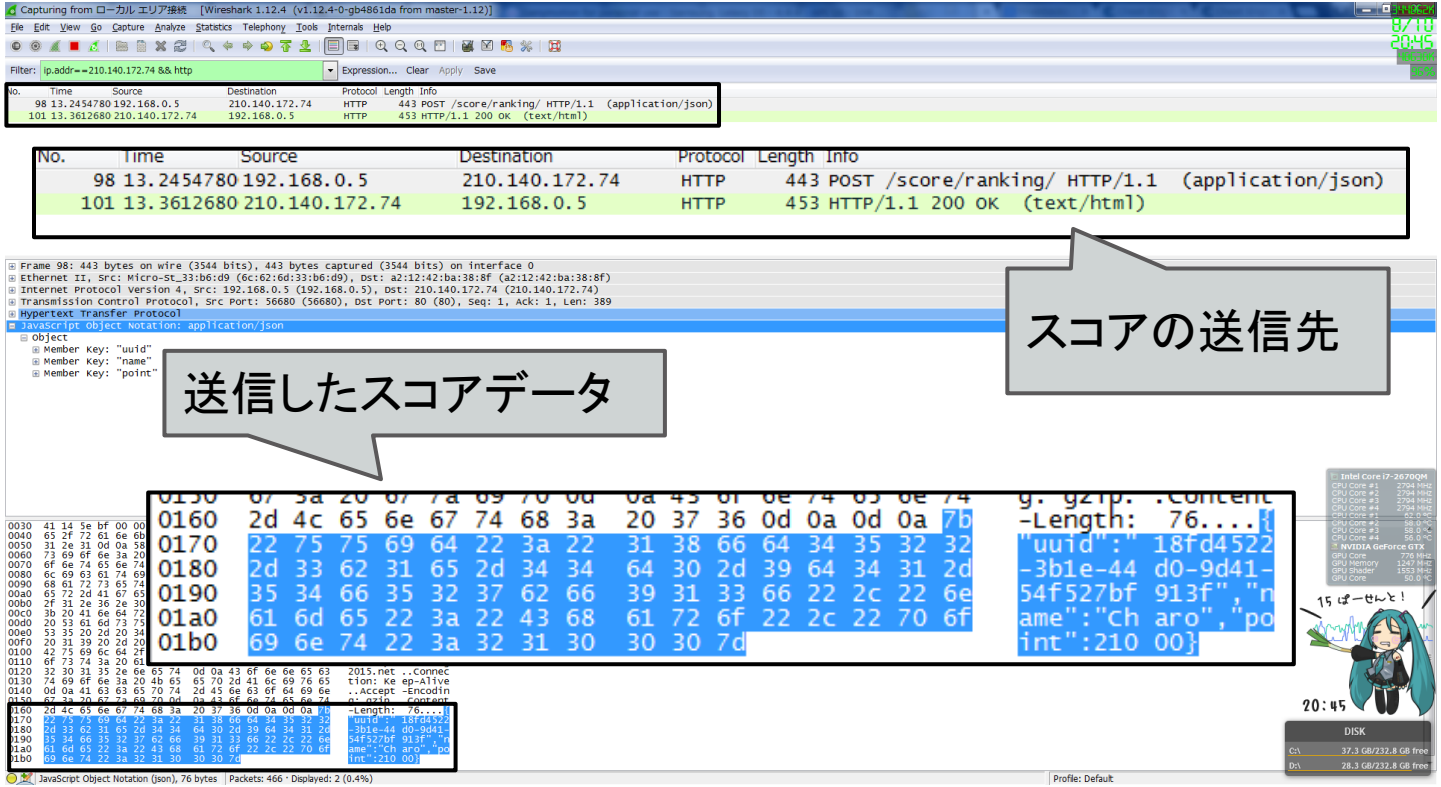
    ゲームプレイにインターネット環境が必須になる、
    といったデメリットも
- 端末に置く場合、暗号化して保存する

[1] sandbagの問題点

1. 不要なログ出力
2. アイテム不正入手
3. 任意のスコアデータの登録
   実際にはプレイしていないのに、
   任意のスコアデータを登録できる



[1]-3. 任意のスコアデータの登録　問題点



[1]-3. 任意のスコアデータの登録　再現手法

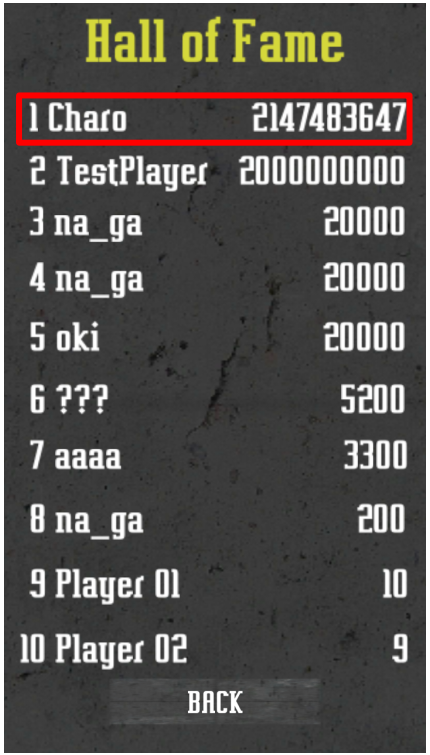
http://api.sandbag2015.net/score/ranking/ に

{"uuid":"任意の文字列", "name":"任意の名前", "point":整数}

をPOSTするだけ

[1]-3. 任意のスコアデータの登録　影響度

ランキングに不正なスコアが登録される



[1]-3. 任意のスコアデータの登録　対策

- httpsにする
  - 「通信内容の取得」を難しくする
  - が、のぞき見用の環境を整えればhttpsも取得可能ということで効果はそこそこ

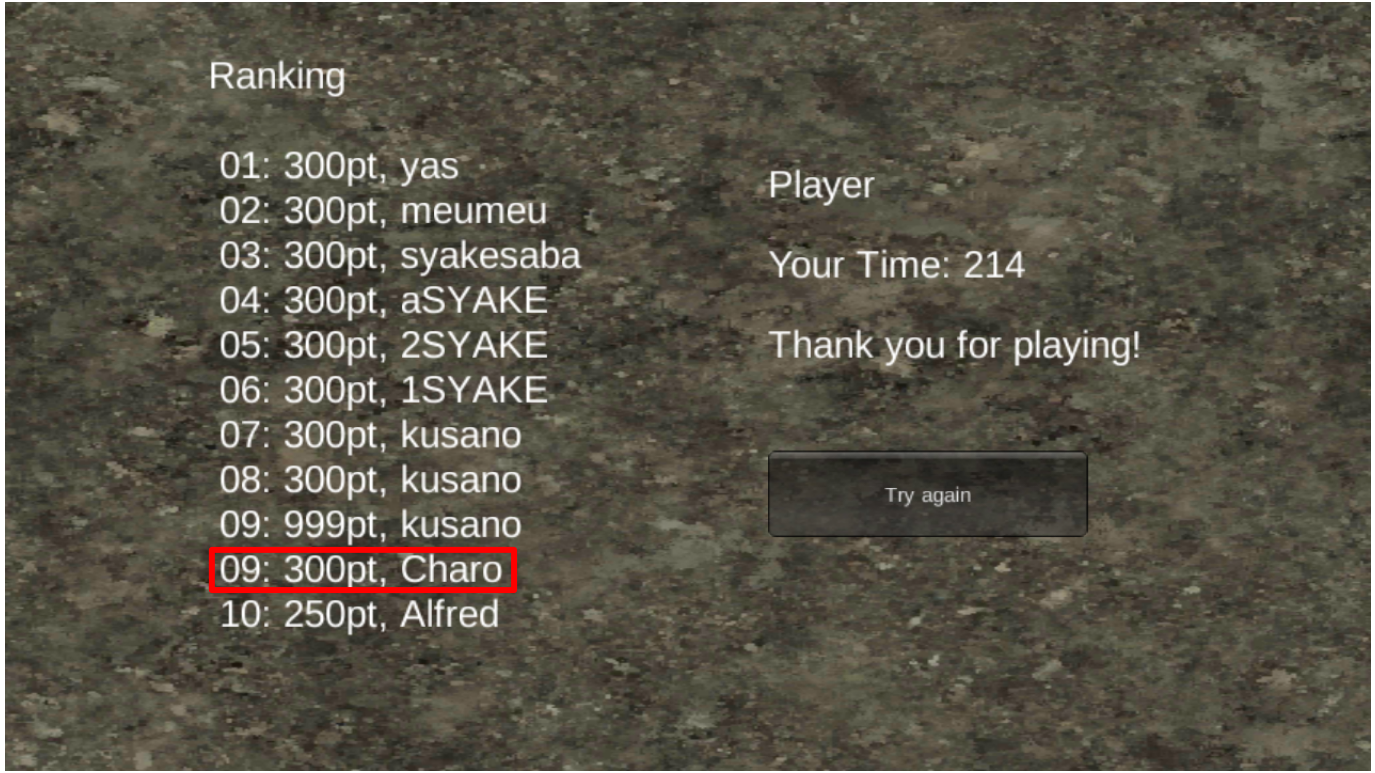
[1]-3. 任意のスコアデータの登録　対策

- スコアデータを暗号化して送信する
  - のぞき見されても、「どんなデータを送ればいいか」がわからないようにする
  - 突破にはアプリの解析が必要になるため、コードの難読化と併せると効果大

[2] SUNIDRAの問題点

1. 任意のスコアデータの登録
2. メモリの書き換えによるチート
3. サーバ側のデータ検証不備

[2]-1. 任意のスコアデータの登録　概要



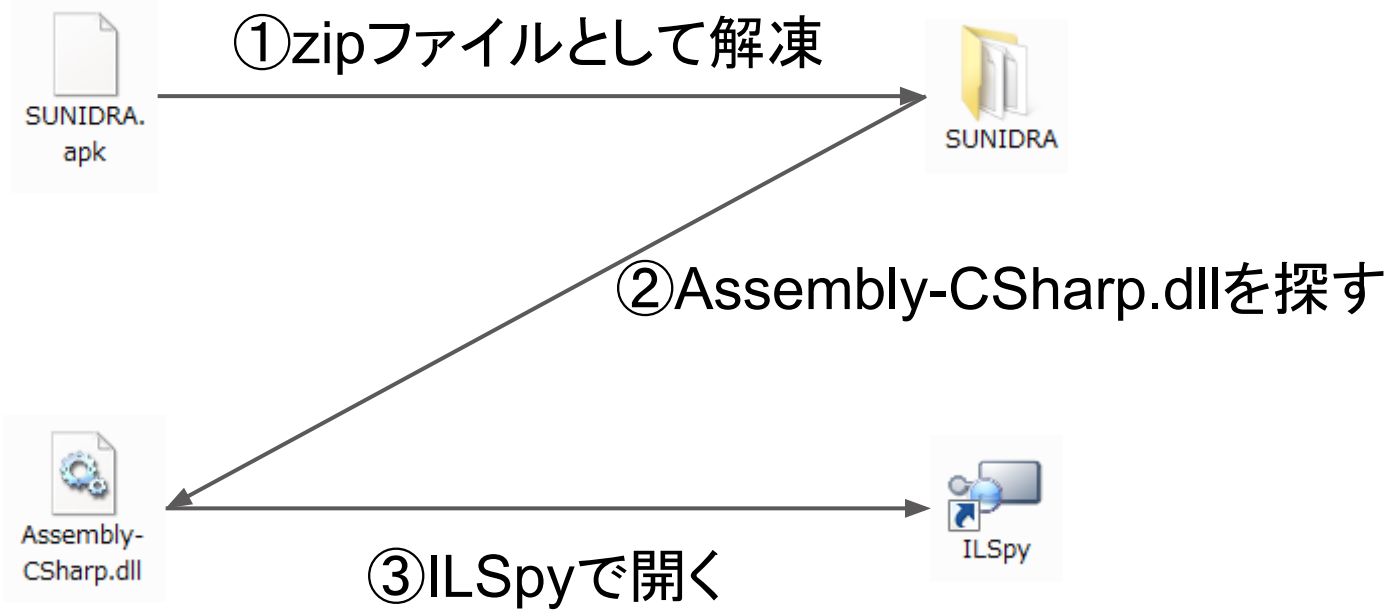
[2]-1. 任意のスコアデータの登録　影響度

ランキングに不正なスコアが登録される

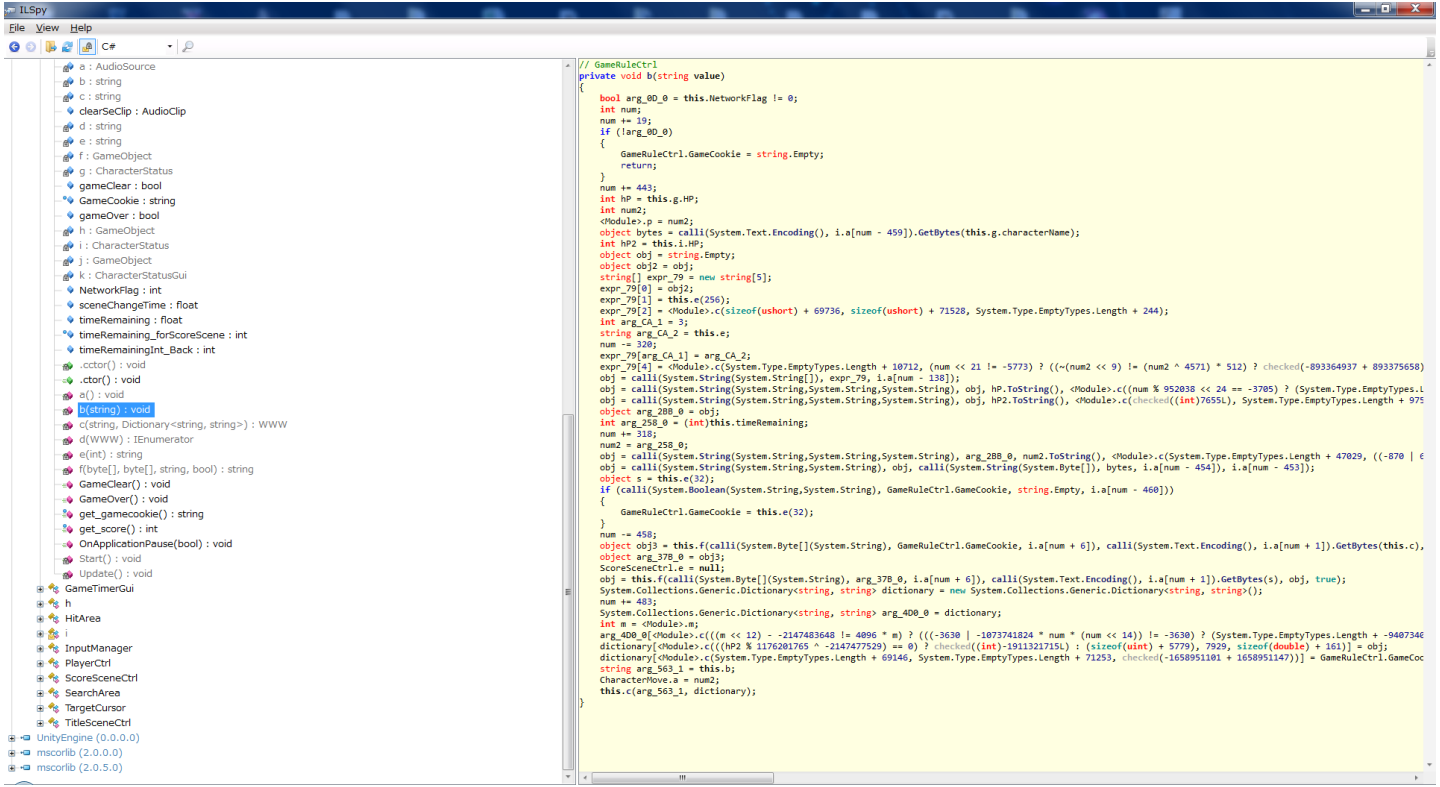
[2]-1. 任意のスコアデータの登録　再現手法

アプリがどうやって送信データを作っているかを
リバースエンジニアリングで解析し、
アプリと同じ手順で送信データを作って送る

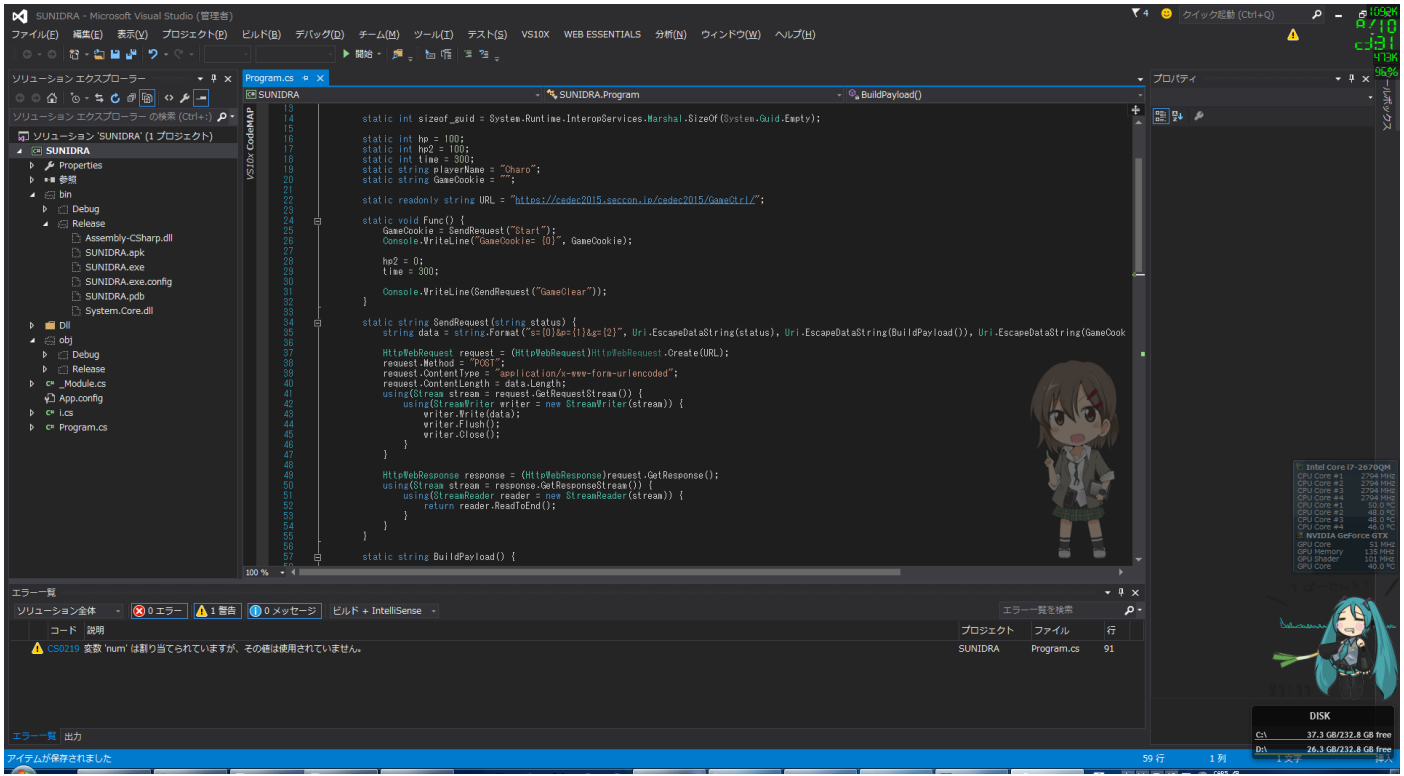
[2]-1. 任意のスコアデータの登録　再現手法



[2]-1. 任意のスコアデータの登録　再現手法



[2]-1. 任意のスコアデータの登録　再現手法



[2]-1. 任意のスコアデータの登録　問題点

httpsが使用され、送信データは暗号化され、
コードは難読化されていた

が……
解析者が根負けするレベルの難読化ではなかった

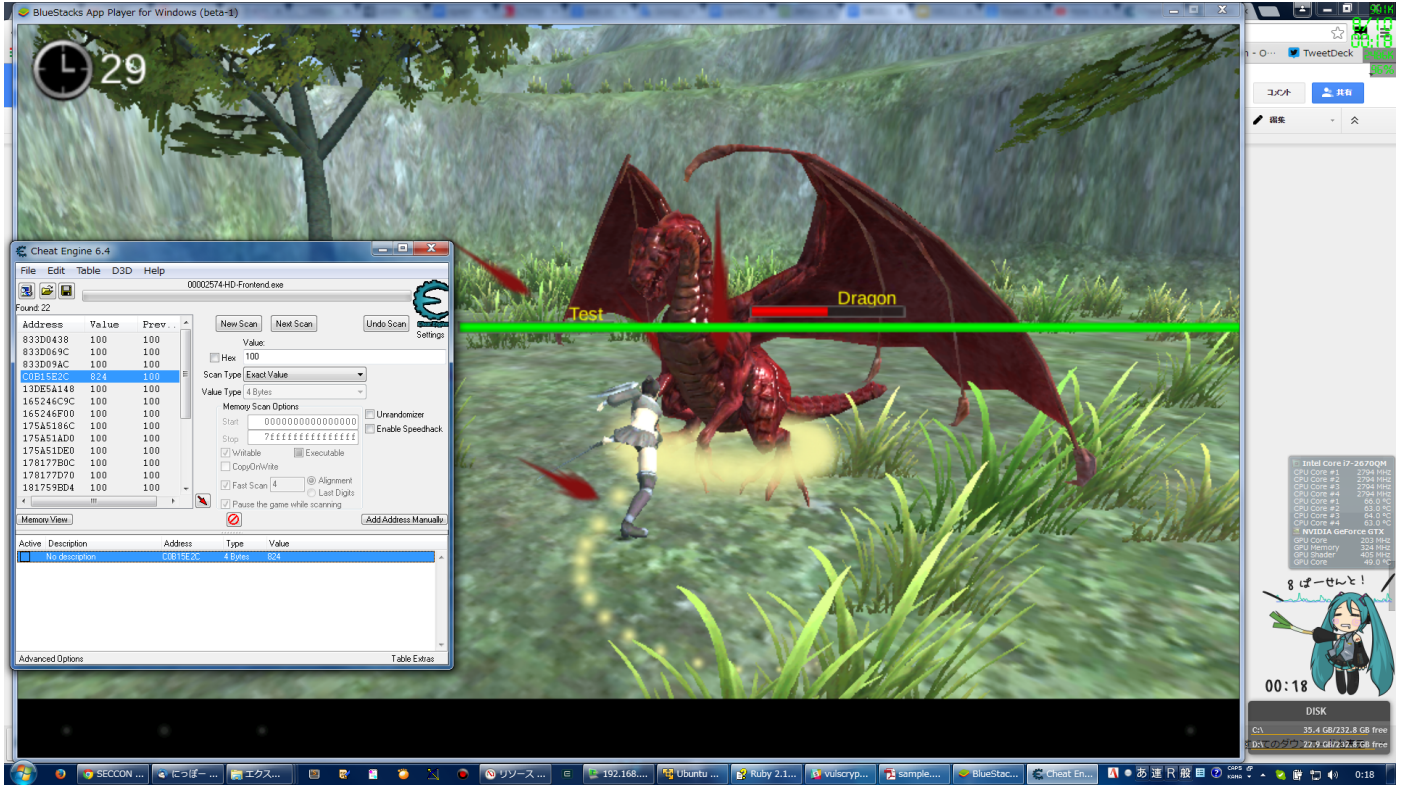
[2]-1. 任意のスコアデータの登録　対策

- 解析の手間を増やす
  - イタチごっこなのでキリが無いが、
    「チートの手間 ＞ チートの恩恵」になれば開発側の勝ち
  - 個人的には……
    送信データの生成処理が
    Javaプラグイン・Android NDKプラグインで
    実装されていたら、解析を諦めていたかも？

[2] SUNIDRAの問題点

1. 任意のスコアデータの登録
2. メモリ書き換えによるチート
   フラグや変数、さらにはコードの内容も書き換え、
   アプリを自分仕様にする
3. サーバ側のデータ検証不備

[2]-2. メモリ書き換えによるチート　概要



[2]-2. メモリ書き換えによるチート　再現手法

難易度：中

1. メモリ書き換えできる環境を作る
   root化したAndroidにプロセスメモリエディタを導入
   または、PCにAndroidエミュレータとプロセスメモリエディタを導入
   (今回はBlueStacksとCheat Engine 6.4を使用)
2. ゲーム中の出来事(被弾・回復等)に応じて変数の値が変動(被弾なら減少、回復なら増加)することを利用し、
   目当ての変数がメモリ上のどこにあるのかを絞り込む
3. メモリを任意の値で書き換える

[2]-2. メモリ書き換えによるチート　影響度

- 根気さえあれば何でもできる
  - 攻撃されてもHPが減らない
  - 一撃で敵を倒せる
  - キャラクターの移動速度を速くする　etc.
- ゲームバランスの崩壊

[2]-2. メモリ書き換えによるチート　対策

- root化された端末・エミュレータ環境では
  プレイできないようにする
  - メモリ書き換えできる環境を排除する
  - エミュレータ環境かどうかは、

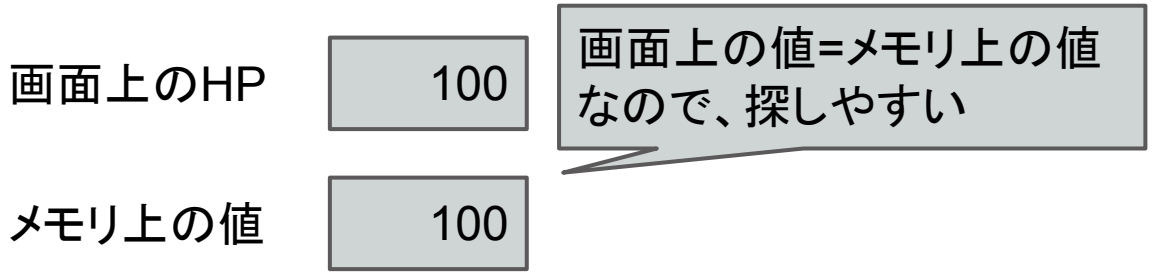
    特定のパッケージが入っているかどうかで検出可能
    (例) com.genymotion, com.bluestacks.appfinder

[2]-2. メモリ書き換えによるチート　対策

- 値をそのまま保持するのではなく、
  ビット反転する等加工して保持する
  - 目当ての変数がメモリ上のどこにあるのかを
    探しにくくする（かなり効果的）
  - getter/setterをうまく使えば実装も簡単

[2]-2. メモリ書き換えによるチート　対策

```
//Before:
public int HP;
```




```
void Attacked(int damage){
    this.HP -= damage;
}
```

[2]-2. メモリ書き換えによるチート　対策

```
//After:
private int _HP;
public int HP{
    get{return ~_HP;}
    set{_HP = ~value;}
};

void Attacked(int damage){
    this.HP -= damage;
}
```

画面上のHP　100

メモリ上の値　-101

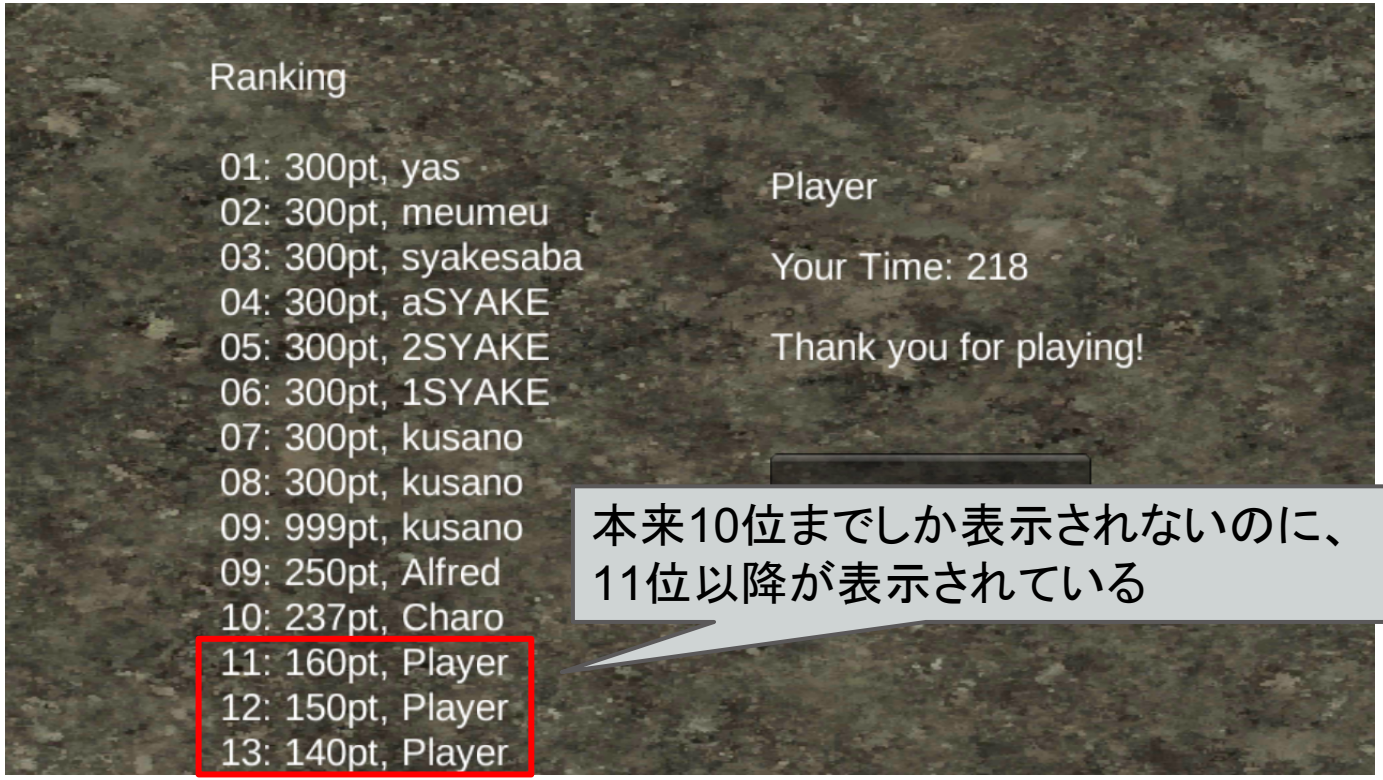
画面上の値≠メモリ上の値
なので、かなり探しにくい

ロジック部分はそのままでOKなので、
ソースコードのメンテナンス性を損なわない

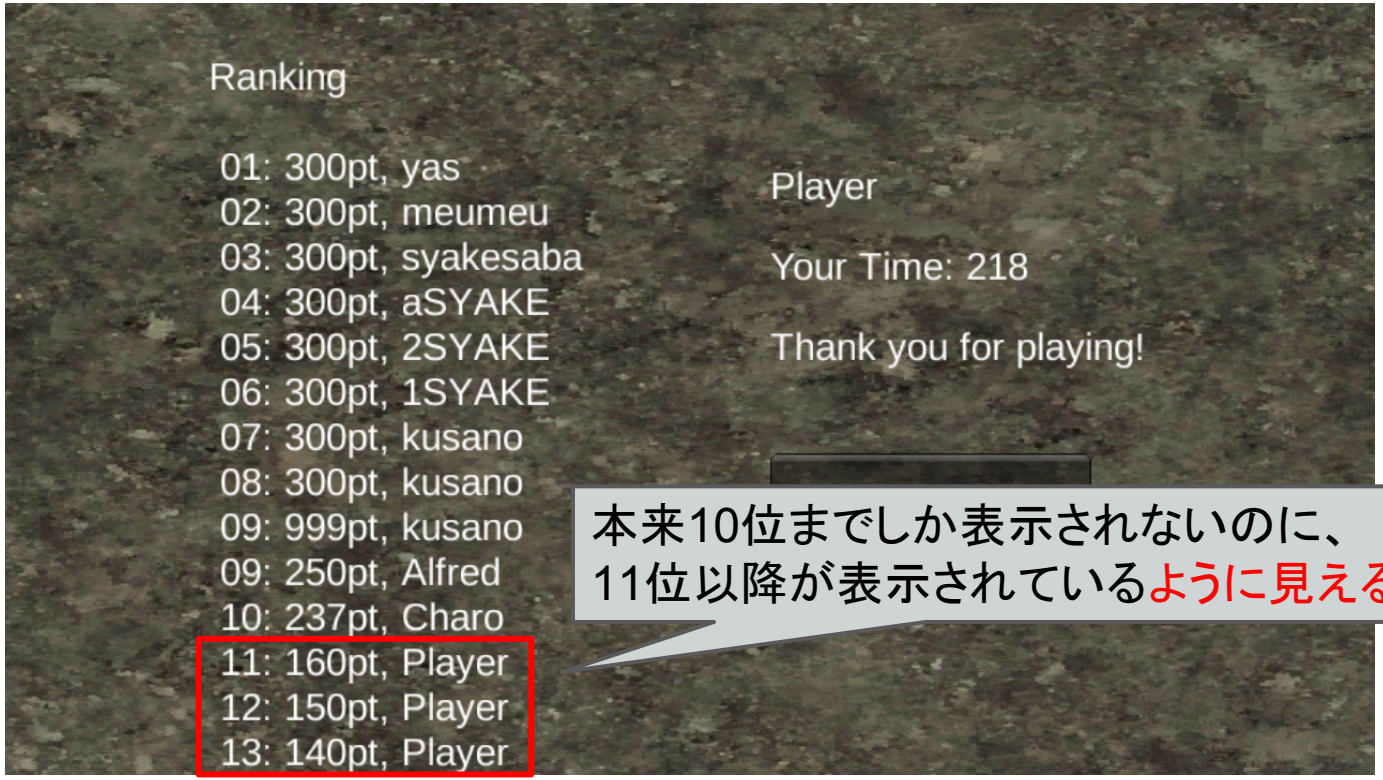
[2] SUNIDRAの問題点

1. 任意のスコアデータの登録
2. メモリの書き換えによるチート
3. サーバ側のデータ検証不備

[2]-3. サーバ側のデータ検証不備　概要



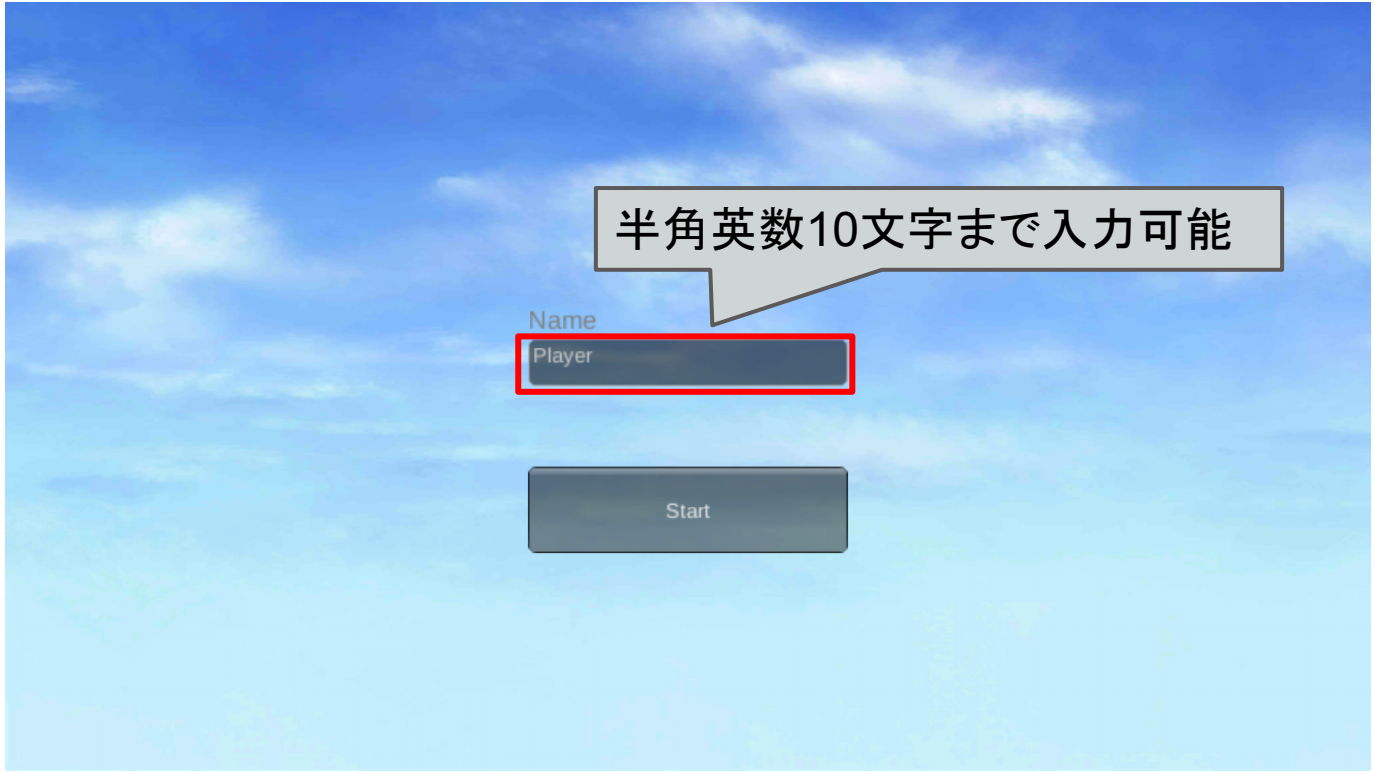
[2]-3. サーバ側のデータ検証不備　概要



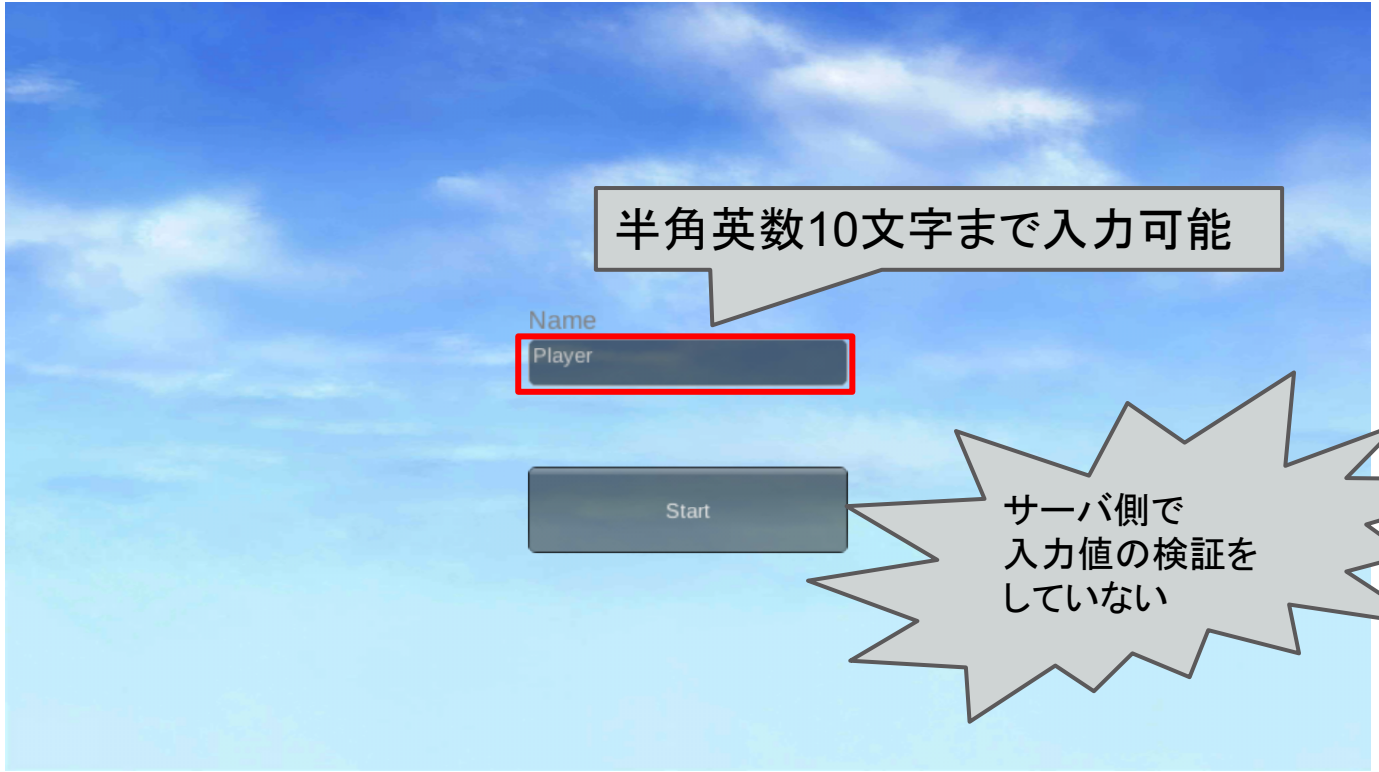
[2]-3. サーバ側のデータ検証不備　再現手法

[2]-1. 任意のスコアデータの登録 のテクニックで、
"Charo\n 11: 160pt, Player\n 12: 150pt,..."
というプレーヤー名でランクインさせる

[2]-3. サーバ側のデータ検証不備　問題点



[2]-3. サーバ側のデータ検証不備　問題点



[2]-3. サーバ側のデータ検証不備　影響度

- 偽のランキングが作れる
- サーバ側のプログラムに脆弱性があった場合、
  サーバを乗っ取られたり、
  サーバからデータを盗まれたりする恐れがある

[2]-3. サーバ側のデータ検証不備　対策

- 送られてきたデータを無条件に信用しない
  - クライアント側で検証するだけでは不十分
  - データ長・文字種等、サーバ側でも検証をした上で登録