ハッキングの実態と対策

- 実践編 -

株式会社エージーテック

Contents

1.　本資料について

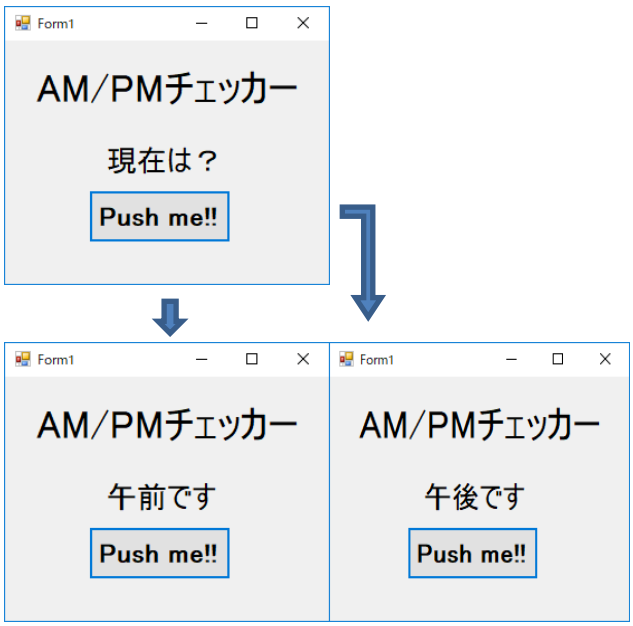
本資料は、無料ツールを使用することによって、簡単にハッキングできる手順をご紹介します。具体的には、実行中のプロセスにアタッチして変数の値を変更したり、復元したソースコードを修正後、再コンパイルをします。

2.　もう「知らない」で済まされない、ハッキング対策

「ハッカー」「クラッカー」「情報漏洩」など、セキュリティに関連する多くの問題をニュースで見かけるようになりました。技術力のないハッカーでも、レベルの高い攻撃を展開できるような時代になったといえるのではないでしょうか。常にセキュリティリスクを考慮する必要がありますが、対策を練る上で大切なのは「弊社のシステムは規模が小さいから、そのようなリスクは起こらないだろう」という考えを捨て、「対策をしていなければ、リスクは必ずある。」としっかりと認識することが大事だと感じます。セキュリティに関する知識の向上、投資をおこない、問題を起こさないための対策をしていきましょう。

3.　ハッキング実践

ツール名称や具体的な手順はお見せしませんが、ハッキング作業の一部を紹介します。今回は無料ツールを使用して、デコンパイル、デバッグ、アセンブリの編集を試します。今回サンプルプログラムとして、「 AM/PM チェッカー」というサンプルプログラムを用意しました。ボタンを押下した後に、現在の時刻から「午前」か「午後」かを表示します。



＜ハッキング実践＞

1) 今回使用したツールでは、対象のアセンブリをドラッグアンドドロップをするだけで、逆コンパイルが可能です。

2) 逆コンパイルしたソースを確認します。ボタン押下時の処理(button1_Click)が確認できます。

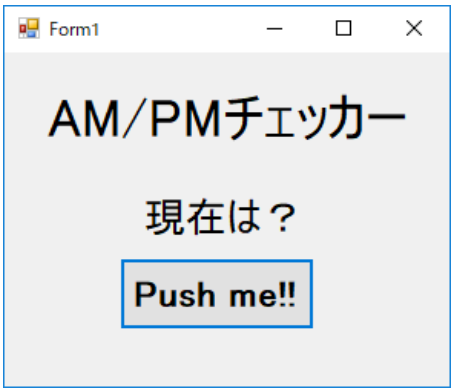
```
    }

    // Token: 0x06000002 RID: 2 RVA: 0x00002060 File Offset: 0x00000260
    private void button1_Click(object sender, EventArgs e)
    {
        if (TimeSpan.Compare(DateTime.Now.TimeOfDay, new TimeSpan(12, 0, 0)) == 1)
        {
            this.label1.Text = "午前です";
            return;
        }
        this.label1.Text = "午後です";
    }

    // Token: 0x06000003 RID: 3 RVA: 0x000020AC File Offset: 0x000002AC
    private bool CheckLicence()
    {
        return true;
    }

    // Token: 0x06000004 RID: 4 RVA: 0x000020AF File Offset: 0x000002AF
    private void Form1_Load(object sender, EventArgs e)
    {
    }

    // Token: 0x06000005 RID: 5 RVA: 0x000020B1 File Offset: 0x000002B1
```

3) 通常デバッグをするために、シンボル（ .pdb ）ファイルが必要となりますが、このツールでは、シンボルファイルが不要でもデバッグが可能です。分岐点（ If(Timespan….) ）にブレークポイントをつけます。

```
17          // Token: 0x06000002 RID: 2
18          private void button1_Click(object sender, EventArgs e)
19          {
20              if (TimeSpan.Compare(DateTime.Now.TimeOfDay, new TimeSpan(12, 0, 0)) == 1)
21              {
22                  this.label1.Text = "午前です";
23                  return;
24              }
25              this.label1.Text = "午後です";
26          }
```

4) デバッグを開始します。

5) プログラムが立ち上がります。



6) ボタンを押下後、デバッグポイントでブレークすることを確認します。

```
// Token: 0x06000002 RID: 2 RVA: 0x00002060 File Offset: 0x00000260
private void button1_Click(object sender, EventArgs e)
{
    if (TimeSpan.Compare(DateTime.Now.TimeOfDay, new TimeSpan(12, 0, 0)) == 1)
    {
        this.label1.Text = "午前です";
        return;
    }
    this.label1.Text = "午後です";
}
```

7) ボタン押下時に出力される label1.text の内容を確認します。

Name	Value	Type
Tag	null	object
Text	"現在は？"	string
TextAlign	TopCenter	System.Drawing.Content

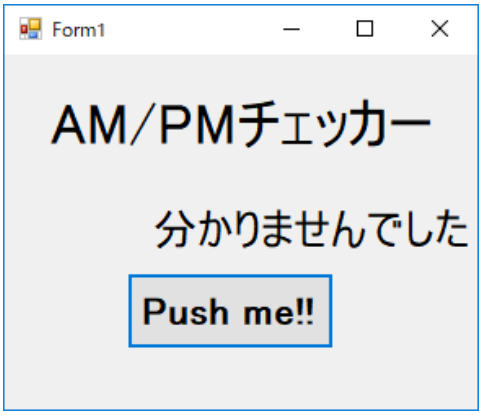
8) ステップオーバーし、「午前です」に変更されたことを確認します。

Name	Value	Type
Tag	null	object
Text	"午前です"	string

9) 「午前です」から「分かりませんでした」に変更します。

Name	Value	Type
Tag	null	object
Text	"分かりませんでした"	string

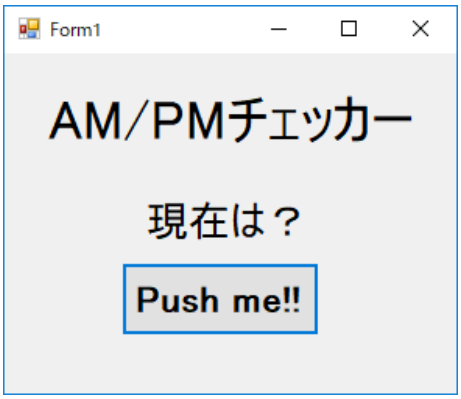
10) 処理を続行させると、ラベルの内容が 9) で変更した通りに変わり、 label1.text の書き換えに成功しました。



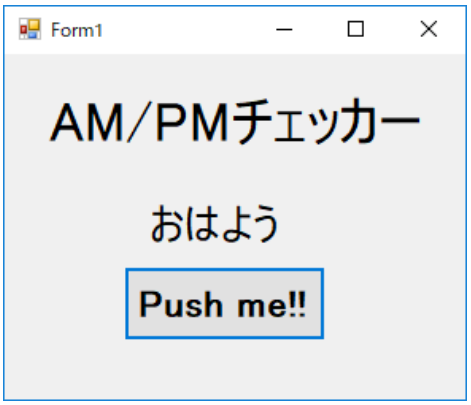
11) 次に、アセンブリを修正します。※実際のソースコードは必要ありません。
分岐のメッセージをそれぞれ「おはよう」と「こんにちは」に変更します。

```
// Token: 0x06000002 RID: 2
private void button1_Click(object sender, EventArgs e)
{
    if (TimeSpan.Compare(DateTime.Now.TimeOfDay, new TimeSpan(12, 0, 0)) == 1)
    {
        this.label1.Text = "おはよう";
        return;
    }
    this.label1.Text = "こんにちは";
}
```

12) そのツールにて、バイナリの再作成が完了したので、改造したバイナリを実行します。



13) 下記の通り、ボタン押下後のメッセージが「おはよう」に変わっていることを確認します。



4.　弊社の Dotfuscator にて難読化を実施した場合

先ほどのサンプルアプリに対して、弊社の Dotfuscator にて難読化を実施した結果が下記になります。

※分岐した多数のモジュールのうち、一部を紹介します。

```
int a_ = 5;
short num = (short)1692998431;
short num2 = num;
num = (short)1860377375;
switch (num2 == num)
{
}
num = (short)1648951296;
if (num != 0)
{
}
num = (short)1540096000;
short num3 = num;
num = (short)1747648513;
if (num != 0)
{
}
return global::a.d("[illegible]", a_);
```

元々のソースコード量が少ないにもかかわらず、ほとんどが意味不明な文字に置き換わっており、少々時間を掛けたところで、ハッキングのきっかけになる部分が見当たりません。文字列の暗号化もしていますので、文字列ベースのヒントもありません。さらに Dotfuscator では、できる限り多くのメソッド名を同じ文字に変換していますので（「1 対多」の名前変換）、解析がほぼ不可能な状態になっているのが分かります。

デバッグチェックの検知も差し込んでいますので、デバッグを試みてもエラーが表示されます。

Dotfuscator の機能概要については、下記 URL をご参照ください。

「.NET アプリケーション 開発向け難読化ツール – 機能概要」

https://www.agtech.co.jp/products/preemptive/dotfuscator/featuresinfo.html

5.　まとめ

このようにハッキングの作業はツールを使用することで、簡単にできることが分かりました。この方法を用いることで、知的財産の流出は勿論ですが、そのアプリに関する不正ツールが作成された場合、御社の収益機会や顧客に対する信頼が損なわれる可能性があります。このようなハッキング事情を理解していただければ、アプリケーションを「強化する」ことを強く検討する必要があるのではないでしょうか。