# 某模拟经营游戏认证分析

## 0x0 前言

最近有朋友推荐了一个旧瓶（旧图像引擎）装新酒（新数据，新DLC）的某经典模拟经营类游戏的全新版本。其本体+DLC价格在大概300-400RMB左右，这本无伤大雅。但现在制作组把原版的纯本地运行改成了启动器启动游戏主程序+联网验证模式，属实丑陋无比。现在让我们一步步尝试分析出它的网络验证协议，写出程序的本地私有验证服务器。



Freedom

--《勇敢的心》威廉-华莱士

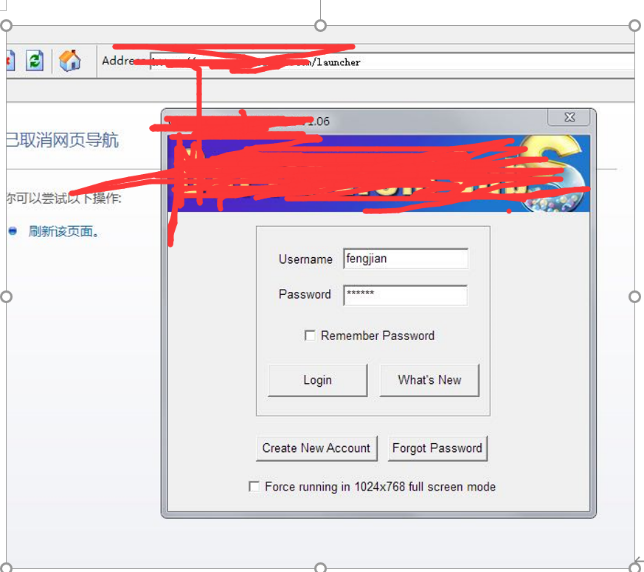
Ps：其实勇敢的心不是指华莱士，而是指后来的苏格兰国王罗伯特·布鲁斯，就是电影里[福尔柯克战役](https://baike.baidu.com/item/%E7%A6%8F%E5%B0%94%E6%9F%AF%E5%85%8B%E6%88%98%E5%BD%B9/22851525)关键时刻背叛华莱士，投靠长腿爱德华的那个苏格兰贵族，但电影仅仅是电源，真实历史并不是这样。



### 班诺克本战役-布鲁斯在一次单挑中一斧头劈死了英国骑士亨利-德博亨

## 0x1 去掉启动器

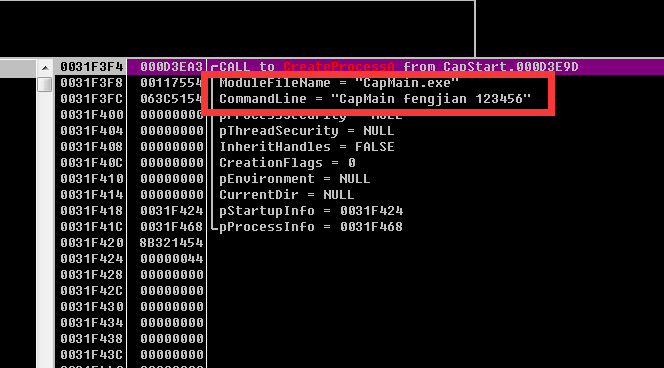
我们直接双击主程序，主程序会直接退出，看来还是要先从启动器开始。打开启动器会弹出一个登录窗口，输入用户名和密码会打开主程序后再告诉你用户名或密码错误。这里就可以推测这个启动器并不负责具体验证，而是仅仅作为接受用户名和密码输入控件，典型的银弹中间层假肢式开发思路。



这里我们随便输入用户名fengjian 密码123456

我想让主程序可以直接单独运行。这里直接挂上调试器，对API CreateProcessA下个断点，然后Run起来，点击登录按钮。





成功断下来后可以看到用户名和密码就是主程序的运行参数，通过此法就可以脱离启动器直接启动主程序了。

## 0x2 分析验证协议

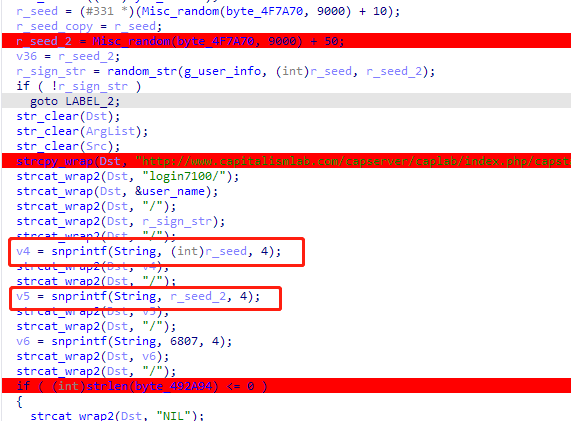
静态分析发现其网络请求主要分三部分

1. 登录验证
2. 补丁下载
3. 游戏得分信息

这里我们主要看关键的登录验证部分

### 登录请求url

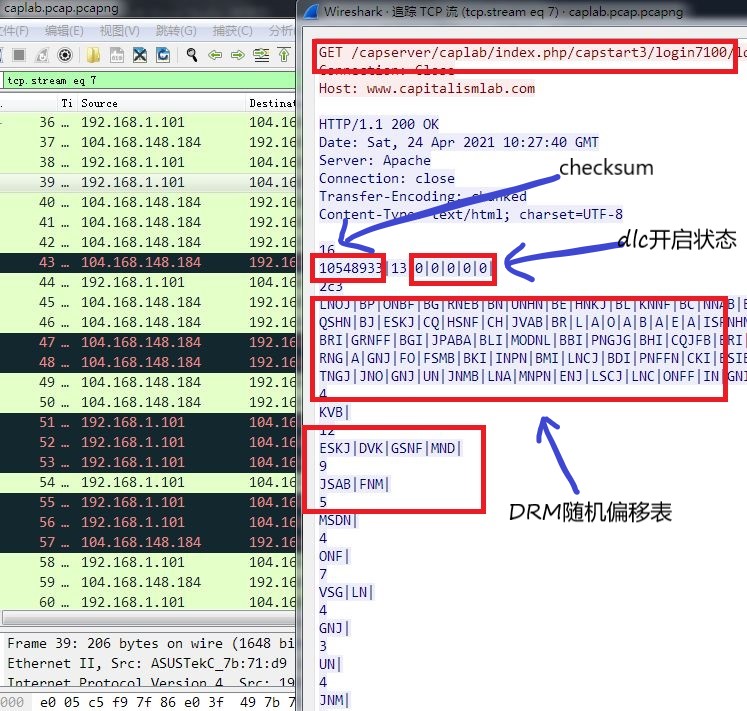
认证协议线程中会用时间当种子生成2个随机数，随后会通过http的uri参数发给服务器，从请求Uri可以看出是其主要构成是xxxxx/login7100/{用户名}/{密码签名}/{随机数1}/{随机数2}



### 认证协议

这里首先通过逆向分析基本流程，以及请购买了完整游戏的朋友帮忙抓包以进一步确定认证过程。

服务器会先通过用户名和密码的hash签名检测用户名和密码正确性，验证通过后生成本次启动的checksum，DLC开启状态，drm随机偏移表返回给客户端。



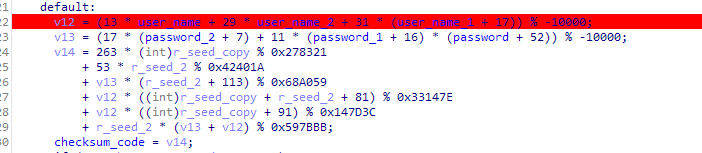
其认证过程如下：

1. 假如用户名和密码不正确，服务器端会直接返回错误码。假如正确则返回一个包含checksum + dlc开启状态 + 加后的DRM随机偏移码表。
2. 计算在内存中的用户名+密码和服务端返回的checksum比较，假如不正确程序会提示效验错误。
3. 假如前面步骤正确或者patch后跳过，服务器端返回的加密随机偏移表假如不是当局随机数生成的，解码后会产生一个不正确的数值表。既程序能进去，但数值是完全错乱的，这防止你抓包后重放请求。

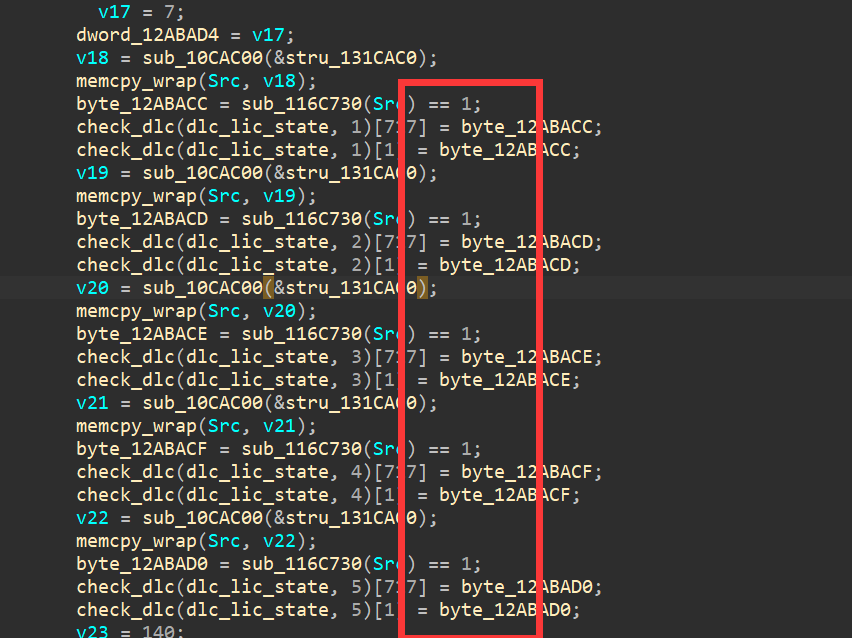
接下来是正餐，有三段校验：

1. Checksum检测

通过用户名和密码的前3个字节和启动时生成的2个随机数做运算得到checksum数字。



1. DLC开启状态



没什么可说的，对应的5个标志位为1即为开启，为0则是关闭。

1. 随机DRM偏移表



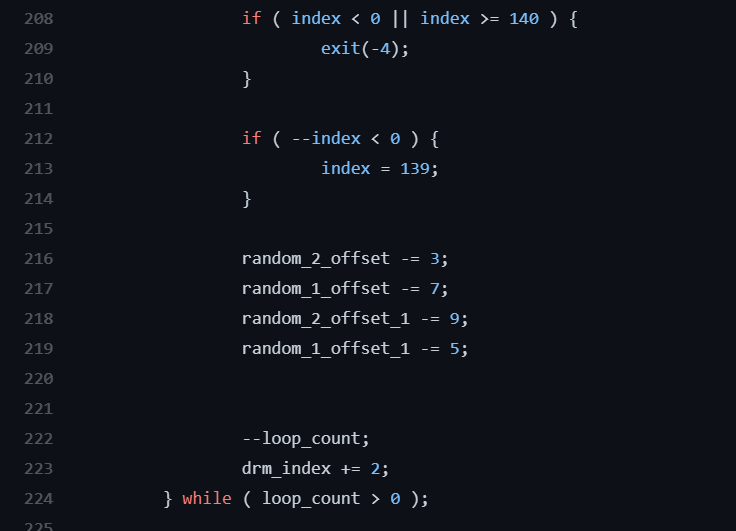
通过先前生成的2个随机数为因子，循环累计并调用解密算法解密DRM偏移。

可以看出这里其实需要知道服务器保存的原始码表才能写出整个验证逻辑。

为了优雅以及后面服务器修改后的扩展性，我们不考虑通过调试和内存dump等方式获取码表。只通过先前朋友正版登录过程中产生的网络抓包逆转算法，反推出保存在服务器中的未加密未随机打乱的原始整型码表。

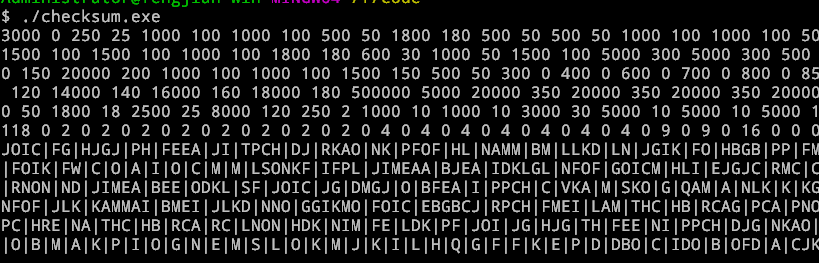
我们可以写下如下码表还原算法：







得到到原始码表

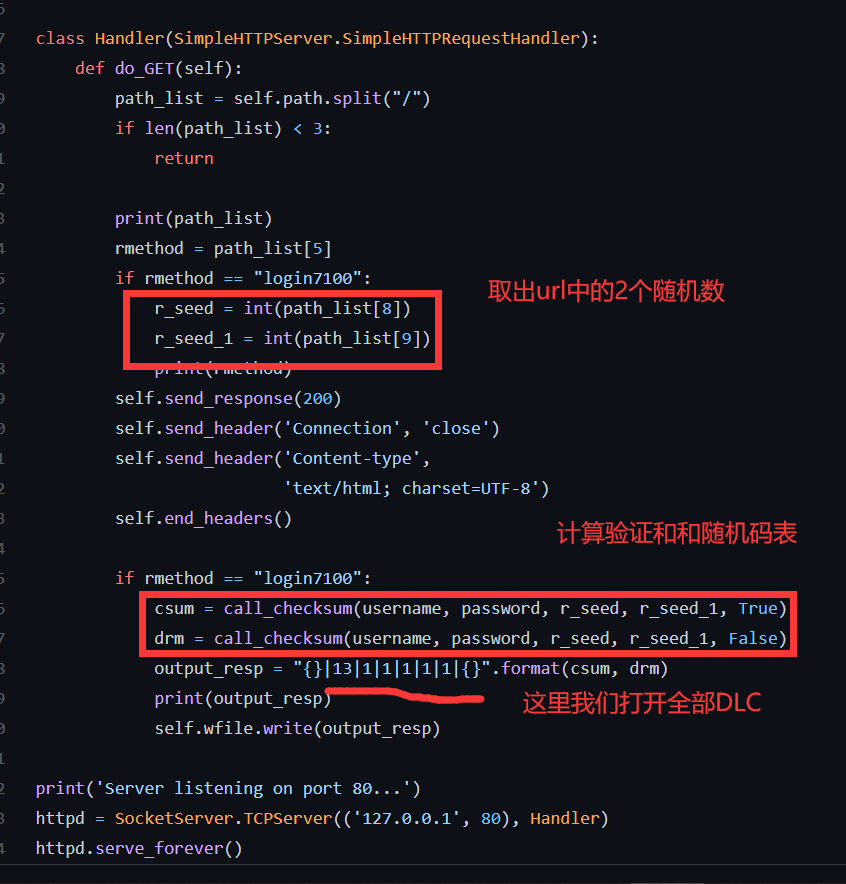


然后就可以根据这个码表写出一个生成DRM随机偏移字符串表的函数



## 0x3 搭建认证服务器

1. 修改本机host 将认证服务器域名指向本地服务器地址，比如127.0.0.1
2. 启动如下服务



## 0x4 结果

完全实现了游戏的网络认证过程，可以开心的离线玩了，不过别忘记补票。

