三叶草二进制第二次面试

首先恭喜各位同学通过了三叶草技术小组第一次面试，希望你们能在接下来的第二次面试中更加出色！

第二次面试以实际操作为主，大部分知识点你们可能从来没有接触过，但是请不要失去信心，我们重点考察学习能力，而非知识储备。基于这样的初衷，我们在这篇引导文章中准备了解题所需的所有学习资料，在这一周的时间里，希望大家使用利用网络，根据提示学习并解题。

解题时间:

2021 年 10 月 23 日18 时 00 分 —— 2021 年 10 月 31日 23 分 00 秒

请在截止时间之前提交解题报告

第二次面试内容

我们为你准备了七道题目，每一道题目都是标准的逆向题，你需要用 IDA 对题目进行逆向分析，并编写解题脚本以及解题报告。

每道题内部都内置了一个算法，你需要通过逆向工程技术，分析该算法，并求出逆算法。

举例来说，每一道题都类似数学表达式 `F(FLAG) == Y`

首先，你需要逆向分析程序，得出 F 函数的具体计算步骤以及计算结果值 Y，然后根据计算步骤以及 Y 的值反推 F 的参数 FLAG 的值。

例如通过逆向分析，你得出 F(FLAG) = 2 * FLAG + 1 == 5，目标值 Y = 5，则你可以解出 FLAG 为 2。

通过例子，相信你已经掌握了基本 CTF 逆向题的解题思路，我们的考核题目只是将 F 函数的功能写得更复杂了一点，其本质跟例子差不多。

七个关卡难度没有递增关系，如果你在某道题目卡住，可以换一道题目。若你没有获得某道题目的 Flag 且写了比较好的报告，仍然可以获取该题的大部分分数。

什么是好的报告？你若在报告中呈现了题目代码的大致逻辑、解题进度、脚本、遇到的困难、解决这些困难的思路、学习到的新知识等等，就算是好的报告。

我们将根据报告判断你在某题的解题进度，即使没有获得最终 Flag，依然可以获取题目 Flag 进度折合分值。

我们非常重视报告中所体现的学习能力以及解决问题的能力，请务必详细写报告。

报告请用 Markdown 完成，推荐使用 `Typora` 软件来编写 Markdown 文档。请合理安排标题层级、代码应该放在代码块中并选择相应的语言。提交报告时，请将 md 文件导出为 pdf 文件与 md 源文件、图片等资源一起打包发送至邮箱 chenqqcom162@qq.com

若你解出题目 Flag 且 没有在报告中体现该题的解题过程，你将无法获得该题的所有分数，甚至可能被质疑存在作弊行为。

题目获取

请二进制方向选手私聊发送学号获取题目压缩包（任何时间都行，最好在10月23日18时之前）

QQ 列表

SYJ: 2962816128

77: 2271897004

marginal: 1330728738

wlz: 2484805199

wmx: 1447380573

我们将在正式开始，即 10月23日18 时 00 分，在二面群公布压缩包解压密码，请提前联系上面的 QQ 获取题目压缩包。

题目运行环境

七道题目均在 Linux x64环境下编译，你可以安装 VMware 虚拟机软件，并在虚拟机软件中安装 Ubuntu 操作系统来运行题目程序。

环境配置步骤

1. 安装 VMware 虚拟机软件
2. 在 VMware 软件中安装 Ubuntu 操作系统
3. 将题目通过共享 文件/SSH/拖放 等方式传入虚拟机系统
4. 虚拟机中打开终端，用 cd 命令切换至题目所在的目录
5. 为题目设置可执行权限 (chmod 777 xxx)
6. 运行题目 ./xxx

调试配置步骤

Windows 上使用 IDA 调试 Linux 程序，只能采用远程调试的方式。

远程调试需要将 IDA 的调试服务端程序上传至目标系统并运行，最后在IDA中配置远程调试的 IP 和端口。

具体过程

1. 上传 dbgsrv/linux_server64 至目标系统（虚拟机中）
2. 使用 chmod 对其添加可执行权限
3. ./linux_server64 运行 IDA 调试服务端

在 IDA 中，配置过程

1. 打开 Debugger 菜单，并选择 Select Debugger
2. 在弹出的窗口中选择 Remote Linux Debugger
3. 再次打开 Debugger 菜单，选择 Process Options
4. 在 Process Options 中填写远程虚拟机的IP地址（ifconfig查看）

关于调试 Linux 程序的视频，可以参考入门教程

upx 脱壳 https://www.bilibili.com/video/BV1nL4y167Xw

环境配置有问题的同学，可以在面试群找逆向方向的管理员帮忙。

评分标准

- 七道题目 Flag (10 分/题)
    - 未成功获取 Flag，我们按解题报告中的进度折合计算
- 解题报告 (10 分/题)
    - 学号、姓名、班级
    - 已解出题目的 FLAG
    - 程序逻辑 (代码 + 文字描述)
    - 解题脚本
    - 遇到的困难（含未解决的困难）
    - 困难的解决方法

    - 学习到的知识
- 答辩(共计 15 分)
    - 我们会根据解题报告提 3 个问题

题目问题反馈 & 求助

你可以向逆向管理员求助以下内容

1. 一切环境搭建问题
2. IDA 操作问题，例如远程调试、如何复制数据、如何安装插件、如何获取伪代码等（我们很建议你先自己查阅资料尝试）
3. 如何运行题目

若你发现题目存在问题，例如无解、多解等问题，请及时反馈。

学术诚信

请各位同学一定要独立完成自己的面试题目，不与它人分享解题思路。

下列行为将视为作弊行为:

1. 与他人提供、交流解题思路
2. 相互抄袭解题报告
3. 相互抄袭解题脚本
4. 提交不属于自己的 Flag
5. 相互分享 idb 文件
6. 其它作弊行为

下列行为不视为作弊行为:

1. 帮助或寻求帮助环境配置方面的问题

若发现作弊行为，将永久失去三叶草小组面试资格。

题目知识点提示

若你在做题的过程中遇到困难，可以看看这些提示来获取一些灵感，不建议直接看，直接看很有可能看不懂！题目中遇到了再来看！注意这不是在劝退你，加油查阅资料，写代码尝试，调试，用上所有的手段完成题目！

异或运算

异或是一种位运算，编程语言中一般使用 ^ 运算符表示，与加法运算符一样，也属于二元运算符。

异或运算具有以下性质

若 C = A ^ B, 则有
A ^ C == B
A ^ B == C
C ^ B == A
A ^ A == 0
B ^ B == 0

异或一般用于加密

加密过程：明文 ^ 密钥 == 密文

解密过程：密文 ^ 密钥 == 明文

Base64 编码

Base64 是一种编码， 通过一张大小为 64 个可见字符的编码表，将所有数据都映射到这张表中的字符。

一个字节需要占用 8 个 bit 位，逢 256 进 1。 Base64 编码表大小为 64（占 6 bit 位），逢 64 进 1，我们可以将编码的过程理解为进制转换的过程。

Base64 编码经常被出题人修改，例如替换编码表为自定义的编码表，这可以防止选手用正常的 Base64 算法解密，解决思路参考 B站三叶草入门培训视频关于 Base64 的讲解。另外，Base64 编码表下标也可能被修改，例如在原算法计算的下标值基础上异或，得出新下标，再用新下标查表。

Z3 解方程组

Z3 是一个 python 库，可以用来解线性方程组（含位运算方程）

安装方法

```
pip install z3-solver
```

值得一提的是，Z3 重载了 python 中的运算符，你可以直接用 python 表达式描述方程组。

更多使用方法建议百度。

AES 加密

AES 是一个比较复杂的加密算法，我们在题目中简化了 AES 的计算过程，以大家的数学基础，完全可以通过逆向理解简化版的 AES 过程。我们去除了AES 中最复杂的列混合，选取了AES中部分子阶段，并修改了循环行为，使得难度更低。

另外，AES 中有个算法叫密钥生成，该算法将输入密钥，生成每一轮AES所需的子密钥，题目程序的密钥是固定的，理论上生成的子密钥也是固定，也许你可以利用这个性质少分析多代码。

二叉树

在尝试 level6 的时候，你将遇到二叉树这种数据结构。二叉树是一种数据结构，建议你通过网络，先快速学习什么是链表，再学习二叉树。

二叉树节点一般来说有三个域，分别是左子树指针、右子树指针、数据，二叉树的遍历算法有前序遍历、中序遍历、后序遍历，任选其一学习。

如何提取程序中的二叉树?

你可以通过调试时使用 IDAPython 提取二叉树数据，见 IDAPython 提示。

C++ 异常

在尝试 level7 的时候，你将遇到 C++ 异常处理机制，你需要先学习 C++ 异常处理机制，并尝试编写一些含有异常处理的 demo 程序，逆向这些 demo 程序来分析 throw 与 catch 的对应关系。

小提示1：C++ 的异常实现与编译器有很大的关系，建议在 Linux 环境下用 g++ 编译你的 demo 程序，这样可以保证与题目一致的环境。

小提示2: C++ 异常处理的 catch 代码，在IDA中伪代码不可见，请尝试切换到汇编模式寻找 catch 代码。

小提示3: 若不确定 catch 代码与 throw 的对应关系，可以在每个 catch 基本块的头部设置断点，然后观察 throw 之后哪个断点命中。

小提示4: 在 IDA 中，将 catch 代码创建成一个独立的函数，可以看其伪代码。

创建独立的函数的方法:

1. 要修改 catch 所在的函数边际使其不包含该 catch 代码，可以通过快捷键 E 实现或者在函数头部编辑函数。
2. 当 catch 代码不属于任何函数时，在 catch 代码头部按下 p 键创建函数
3. 创建函数之后可以用 F5 得到伪代码

随机数

C 语言中一般使用 srand(x) 函数设置随机数种子，rand() 获取随机数。若设置相同的随机数种子，则 rand() 获取到的随机数序列每次都相同，所以随机数并不一定真的随机。

注意，Windows 与 Linux 的随机数算法不一样，python 与 Linux 的随机数算法也不一样，即设置同样的种子，不同的随机数算法获得的结果不同。

IDA 脚本与调试提示

使用 IDAPython 脚本，可以帮助我们做一些自动化操作，例如数据提取。

调试时常用

- idc.read_dbg_qword(va) 读取 va 地址上的一个 int64 整数值
- idc.read_dbg_byte(va) 读取 va 地址上的一个 int8 整数值
- idc.read_dbg_memory(va, size) 读取 va 地址 size 字节数据，返回字节数组
- idc.patch_dbg_byte(va, value) 将 va 地址上的值修改成 value
- idc.get_reg_value("rsp") 读取 rsp 寄存器值， rsp 可以换成其它寄存器
- idaapi.set_reg_val("rax", 123) 将 rax 寄存器的值设置为 123

对于提取运行时数据，你可以在想要的数据生成完成之后设置断点，当断点命中时，将脚本复制到 IDA 底部的命令窗口，按下回车即可执行脚本。

若你在脚本运行时遇到问题，可以向逆向方向管理员寻求帮助。

我们可以帮助你解决脚本运行操作及环境问题，但是不会为你解答脚本的代码问题。