MoeCTF 2020

逆向工程入门指北, Copyright @ 2020 Reverier, XDSEC

逆向工程是什么?

简而言之, 你可以把逆向工程理解为从产品倒推设计方案的一门技术. 这是一门比较黑客向的东西, 想尽办法恢复别人深藏在产品中的设计思想, 并收归己用or进行更深层次的破解与利用.

CTF中的逆向工程指的都是软件逆向工程, 逆向的对象不仅包括你最常见的Windows或者Linux操作系统上的那些二进制文件, 也包括安卓的安装包, 物联网设备的固件, 形形色色的中间语言(例如Java的class文件, 既不属于机器代码, 也不属于人类可读的代码, 而是一种能够运行在Java Runtime, 也成为Java虚拟机JVM的东西上的字节码), 总之一切能隐藏逻辑, 并具有执行特定功能的文件, 都是软件逆向工程的目标.

我要从哪里开始呢?

就从最简单的Hello World开始吧.

逆向工程是一个逆向的过程, 那正向是什么呢? 没错, 是编程. 也许你不会编程, 但是歪打正着做对了一些题, 这是可以的, 但你越往下走, 就越艰难. 你要熟悉正向过程, 才能在逆向分析中游刃有余.

所以, 如果你还不会C语言, 请现在开始学习编程吧. 入门编程请从C语言开始.

为什么选择C语言

> *可是为什么不用人生苦短的Python呢? 为什么不从C++开始呢? 为什么不从其他更加高级/现代的语言开始呢?*

你可能会有这样的疑问, 没错, 编程考验的是逻辑处理能力, 但是逆向并不只是逻辑处理. 你需要更贴近计算机的运行原理. C语言在保证贴近高级语言的同时, 提供了足够底层的支持, 并且C语言的结构相对简单, 没有类与对象等各种复杂特性, 因此更适合拿来理解计算机执行程序的原理. 更高级的语言你学着挺舒服, 学完了一问运行原理你还是啥都不知道, 即使你想去了解也很难, 高级语言的实现过程一般都是你想象不到的复杂.

同时C语言具有设计上的缺陷, 但绝大部分的其他高级语言的解释器/编译器/运行时还是使用C语言编写出来的, 学会C语言之后再去入门其他的语言门槛会低很多, 但是你先学Python再学C, 你会发现C比你想象中的要难, 所以入门编程请从C语言开始.

如果上面说了这么多都无法打动你的话, 那么接下来的话你听好了: C语言是大一上学期必修课, 并且在你整个大学过程中都阴魂不散, 这下你该去学了吧?

学了一点C语言之后呢?

接下来你就可以尝试着上手逆向工程题目了. 有个叫做IDA的工具可以帮你把神必的机器代码变成汇编语言, 还能帮你把让人头晕的汇编语言变成类似C语言的代码. 接下来你就可以安心的分析程序逻辑了, 这就像是在看别人的代码, 然后揣度别人的意思一样, 通过分析程序内部的算法, 再运用一点逻辑和数学的知识, 你就能成功的把隐藏的信息给挖出来.

关于IDA如何使用, 我们就不再介绍了, 篇幅写不下, 本文只是给你指明一个方向, 具体使用方法还需要你自己通过谷歌或者百度一步步探索, 作为学长我可以为你引路, 但不能当你的老师/保姆呀. 至于资源去哪里找, 下一题有IDA的资源链接.

逆向题一般是什么套路?

我们不从逆向工程师的角度来看, 我们从一个软件开发者的角度来看.

比如让你写一个密码验证器, 你会怎么做?

```
strcmp(input, "this_is_password");
```

你可能会这么写, 但是这样写毫无逻辑可言, 就是简单的字符串比对. 你编译一下扔进IDA, 然后按下F5键, 你就能发现你的密码就躺在程序里, 可以被轻松破解掉.

那安全一点的写法呢?

你可以用各种算法来保证你的数据没有那么容易被别人给搞出来, 比如你可以使用异或:

```
char password_enc[] = {98, 126, 127, 101, 73, 127, 101, 73, 102, 119, 101,
101, 97, 121, 100, 114};
// password_enc的每一位和22进行异或, 就能得到真实的密码"this_is_password"
for (int i = 0; i < 16; i++) {
    if (input[i] ^ 22 != password_enc[i]) {
        printf("Password is wrong!\n");
        exit(0);
    }
}
printf("Password is right!\n");
```

虽然这个算法也很简单, 但编译之后的破译难度是不是增加了?

那么作为一个逆向手, 你的主要工作就是:

- 找到目标数据, 无论是加密过的还是没有加密过的
- 从你的输入开始, 逐步分析他的代码都干了些什么, 一直到输出的位置
- 把算法抽象出来, 理解他究竟对目标数据做了什么
- 尝试通过写出逆算法, 把加密的目标数据成功还原为加密前的数据.

例如上面那个异或数据的例子, 我们就很容易破解:

```
// 因为a^b=c时, b^c=a, 所以我们可以这样还原数据:
char password_enc[] = {98, 126, 127, 101, 73, 127, 101, 73, 102, 119, 101,
101, 97, 121, 100, 114};
char password[17];
for (int i = 0; i < 16; i++) {
    password[i] = password_enc[i] ^ 22;
}
password[16] = 0;
printf("%s\n", password);
```

简单吧.

真实做题中你会逐渐遇到更难的加密方式与算法, 这不仅要求你的编程能力要逐步提高, 还要求你要有耐心, 逆向的过程一般都是冗长而无聊的, 但是解出题目那一瞬的快感可以让你高兴一整天(我是这样的).

你说的好像挺简单, 但是好像不是所有题目都长这样啊

当然. 我前面介绍过, 任何能够执行的文件都是我们逆向的目标, 所以针对不同语言, 我们有不同的工具. 遇到新的逆向题目不要害怕, 越是古怪的题目可能越简单. 这个时候搜索引擎就派上大用场了, 你可以找到各种形形色色的工具, 但最终还是要回归到上面讲的逻辑分析上.

逆向只有这些嘛?

并不, 随着软件保护技术的发展, 先后出现了加壳, 花指令, 指令虚拟化, 虚拟机等等等各种更加高级的保护手法, 有时候可能IDA也对这些保护束手无策. 你可能还需要动态调试工具, 通过程序运行一步一步分析在保护之下的真实逻辑, 你可能也需要脱壳工具来帮助你脱掉程序的保护壳, 有时候脱壳工具也不管用, 你还需要了解手动脱壳.....你以为我会在这里给你全部说一遍嘛? 不可能的, 说完我都可以出书了.

我的目标已经达成了,万事开头难, 我只想帮你迈出第一步, 后面的探索过程就留给你自己啦. 学会用搜索引擎, 并在搜索引擎无能为力时懂得找学长寻求帮助, 你的进步一定会非常快的. 但学习逆向工程的路不会很顺利, 请做好准备哦~

加油, 祝你在逆向工程领域能取得好成绩!

本题flag: `moectf{0hhhhhhh_I_knOw_hoW_tO_R3v3rs3!}`