软件安全课程实验一

实验一 Linux 平台漏洞攻防工具使用

实验目的

- 掌握 Linux 进程的原理;
- 了解 ELF 可执行文件格式与载入原理;
- 掌握知名反汇编工具的工作原理与使用方法;

实验要求

- 编写代码独立解析 ELF 文件，获取 ELF 文件元信息
- 需独立使用进程分析工具分析给定 ELF 的进程地址空间
- 以小组为单位完成对 ELF 程序的逆向分析后获取 flag

实验环境

- 实验平台：pwn.hust.college
- 操作系统：Ubuntu 20.04
- 分析工具：IDA free version

实验内容

- 实验内容一：使用 Rust，C，Python 等语言，不调用外界库，根据 PPT 和课上所学的知识，获取 ELF 的文件元信息。要求至少读取以下信息：段在文件中的偏移量、段虚拟地址、文件中段的长度、内存中段的长度、节名（.text, .data）、节在执行时的虚拟地址、节在文件中的偏移量、节长度。
- 实验内容二：根据课堂内容，使用 gdb 命令 info proc mappings 或者 /proc/{进程 pid}/maps 查看给定 ELF 可执行文件（easy_re）的进程地址空间。
- 实验内容三：以小组为单位针对给定 ELF 可执行文件（easy_re）进行合作逆向并获取程序中的 license：

- 题目的 license 格式为 XXXX-XXXX-XXXX-XXXX 共四段 license 字符串的限定范围是 0-9,a-z,A-Z，在题目中每段密钥均有一个小关卡，解出小关卡内容，即可获得对应片段的密钥。

- 第一个密钥片段我们使用明文进行对比，使用 IDA 尝试逆向，或者通过 gdb 进行动态调试，即可获得第一个密钥片段。参考视频链接 https://www.bilibili.com/video/BV1oG411o7A4

- 第二个密钥片段我们使用了异或移位算法，因为异或移位均为可逆，因此根据逆向的结果，在脚本中通过简单的^ 和 << 即可获得第二个密钥片段。

- 第三个密钥片段我们使用了较为复杂的异或移位算法，推荐使用 z3 求解，不推荐使用手工计算。

    - z3 是一个约束求解器，在 python3 中的库为 z3-solver。
    - z3 中有 3 种类型的变量，分别是整型(Int)，实型(Real)和向量(BitVec)。在本次实验中，我们使用 BitVec 函数。
    - 使用 Bitvec(name,bv,ctx=None)，可以创建一个位向量，name 是他的名字，bv 表示大小
    - 定义约束条件时，我们将逆向代码的逻辑作为未知方程进行写入，并约束字符范围为可见字符。
    - solver.check()使用 check 函数查看是否有解，如果有解返回值为 sat，没有解则为 unsat。
    - 有解的情况下，使用 solver.model()即可获得一组正确的解。
    - 有关 z3 详细参考内容，如下链接：
    - https://firmianay.gitbooks.io/ctf-all-in-one/content/doc/5.8.1_z3.html
    - https://ericpony.github.io/z3py-tutorial/guide-examples.htm
    - https://github.com/Z3Prover/z3/wiki

- 第四个密钥片段我们将该四字节进行 md5 加密，密文通过 IDA 或者动态调试获取，随后我们可以通过脚本爆破，即可获得第四个密钥片段。

- 我们针对前八个字节，前十二个字节，分别做了检查，当输入正确的片段后，会获得对应的响应，当获得四组密钥片段后，输入我们得到的 license 即可获得 flag。

实验总结

在本次试验中，要求给出自己逆向与分析的体会。包括难点，关键技术以及遇到

的问题解决方法。

实验代码

根据课堂讲解内容给出 `python` 利用脚本相关代码。编写所需的库介绍