什么是逆向 （⊙ˍ⊙） ？

CTF比赛中的逆向工程题目就像是解开一个秘密宝藏的地图一样有趣。在这样的比赛中，我们通常会得到一些程序或者文件，而我们的任务就是去揭开它们隐藏的秘密。首先，我们会用到各种工具，比如反汇编工具像IDA Pro或者是开源的Ghidra，还有调试器如GDB或者Windbg，通过这些工具我们可以看到程序内部的样子。然后呢，我们要仔细观察程序的行为，找出那些加密算法或者逻辑谜题。

在这个过程中，我们需要动动小脑筋，像猫咪捕猎一样小心翼翼地跟踪程序的每一个动作～。有时候，我们需要一点点耐心，因为可能会遇到复杂的逻辑或者是加密算法，这就像是在迷宫里寻找出口一样。最后，当我们成功解开谜题的时候，那种成就感就像捕捉到了一只特别狡猾的老鼠一样让人开心！而且，逆向工程不仅能锻炼我们的逻辑思维能力，还能让我们更好地理解软件是如何工作的。

工具篇

工欲善其事，必先利其器，使用正确的工具是成功的第一步喵～。让我来分享一下常用的工具库，希望能帮到你 （//▽//）：

首先是 **IDA Pro** ，这是一款超级厉害的递归下降反汇编器，简直是逆向分析的神器之一！它可以分析x86、x64、ARM、MIPS、Java、.NET等众多平台的程序代码，是安全分析人士不可缺少的利器！

接下来是 **x64dbg+x32dbg**，这其实是指x64dbg和x32dbg两个调试器。它们都是免费且开源的，专为Windows操作系统设计，非常适合用来进行逆向工程、漏洞研究或是软件调试喵～。

还有 **吾爱破解专用 OllyDbg** ，OllyDbg 和 x64Dbg 大致属于同一类别的调试工具。它们的优势在于即使在没有符号信息的情况下也能进行调试（虽然也可以使用符号信息）。OllyDbg已经有一段时间没有更新了，而这个“吾爱破解专用OllyDbg”是带有大量插件的民间版本。

再来是 **Detect it Easy** ，这是一款轻松检测PE文件的程序。它的主要作用是查壳，并将PE文件的内容解析出来，包括导入函数、导出函数的名称及地址，以及入口函数地址等，是技术人员分析软件时的好帮手！

JADX 是一个 Dex 到 Java 的反编译器，用于从 Android Dex 和 Apk 文件生成 Java 源代码，非常适合做Android应用的逆向分析。

Dnspy 是一款开源的 .NET 反编译器和调试器，它允许你查看、编辑、反编译和调试 .NET 应用程序，还包含一个反汇编器，可以将汇编代码转换成更易于阅读的格式喵～

对于 **Linux环境（VMware/WSL）**，有时候我们需要在Linux环境下动态调试ELF文件，这时就可以用到大名鼎鼎的虚拟机 VMware 或者是微软的 WSL（Windows Subsystem for Linux），它们都能帮助我们在Windows系统中运行Linux环境。

Pwndbg 是一个用于GDB的插件，它提供了一系列功能和命令，帮助我们更方便地进行动态调试，特别是在做漏洞利用和CTF挑战时。

最后是 **Cheat Engine**，这是一款专注于游戏修改的工具，可以用来扫描游戏内存并允许修改它们。它还附带了调试器、反汇编器、汇编器、变速器、作弊器生成、Direct3D操作工具、系统检查工具等。

以上就是我常用的工具库，希望这些工具能够帮助到你，让我们的逆向工程之旅更加顺利喵 (ฅ´ω`ฅ) ！

语言篇

下面并不一定是按照语言进行说明，请见谅。(>__<。)人言道：一个人的正向水平决定了他的逆向水平。在了解对应的语言逆向前，最好还是先学习该语言的基础知识。下面就是简单的思路分享：

对于经典的C/C++逆向，当我们拿到一个可执行文件的时候，首先要做的是用一些小工具，比如DIE或者exeinfo来看看它有没有穿上“壳”的外套。如果有，我们得先帮它脱掉这层壳哦。然后就可以开始我们的静态分析之旅啦，在IDA的帮助下，我们可以看到程序内部的一些结构。如果这时候觉得有点难，可以使用x64dbg或者x32dbg（如果是Linux环境的话，就用dbg）来进行动态调试，这样就能更直观地看到程序运行时的情况了，喵！

对于Python的逆向嘛，有时候我们会遇到.py文件被打包成了exe或者是pyc文件。这时候，可以用pycdc这样的工具来让pyc文件变回.py文件的样子。如果遇到一些调皮的开发者修改了魔法数字（Magic Number），可以需要我们手动调整，如果反编译还是无法成功，那可能就需要我们一点点地去看字节码了，虽然稍微有点麻烦，但是也很有趣呢，喵！

.NET和C#的逆向相对来说就简单多了，因为用dnspy这样的工具可以直接把它们变成很

接近源代码的样子，就像剥开一层糖纸一样简单，喵！

Rust和Go语言的逆向嘛，可以把它想象成是比C++还要复杂一些的大迷宫，需要更多的耐心和技巧去解开谜题，喵！

对于Android应用的逆向，jadx是个好帮手，它可以让我们很容易地看到APK里的Java代码。如果涉及到native层，那就需要用到IDA来查看了。有时候，为了更好地理解程序的行为，还可以使用hook技术，比如用frida这样的工具来深入探究，喵！

至于游戏的逆向，通常有两种情况：Mono和IL2CPP。Mono的话，我们可以用dnspy来轻松反编译；而IL2CPP则需要一点额外的帮助，比如Il2CppDumper这样的工具。如果是要做一个~~外挂~~插件，那么 Cheat Engine （简称CE）就是个不错的选择，喵！

当然啦，逆向的世界还有很多很多种不同的类型，这里说的只是冰山一角。随着时间的推移，通过不断地实践，相信我们一定能成为逆向工程的小能手的。 ¤_¤✧

智慧的对抗

逆向工程的世界里，自然会有各种各样的对抗手段来保护程序的秘密不被轻易揭开。一般来说，这些手段包括加壳、加解密以及混淆哦。而在CTF比赛里面，出题人常用的这些小技巧啊，就像是给我们的小挑战加上了额外的乐趣呢：

首先是花指令，这就像是一场游戏中的烟雾弹，虽然不会干扰程序本身的运行，但却会让IDA这样的分析工具感到困惑。对于简单的花指令，我们可以手动调整一下反编译结果，但如果遇到复杂的花指令，那就要借助动态调试，追踪程序的执行流程来调整了，喵！

接着是壳，加壳的程序就像是穿上了隐身斗篷，它的执行流程通常是先加壳压缩，然后壳先执行，接着还原源程序并执行。对于常见的壳，我们可以找现成的工具来对付，但如果遇到魔改的壳，可能就需要手动脱壳了。这时候，掌握一些技巧，比如ESP定律来寻找原始入口点（OEP）就非常重要啦，喵！

再来是SMC，也就是自我修改代码，这种技术通过在运行时动态解密代码或数据来防止静态分析。破解SMC通常有两种方式：一是找到加密函数并通过idapython写脚本来解密；二是动态调试直到解密完成后再dump出来！

Anti-debugging，也就是反调试技术，这是出题人用来检测是否正在被调试的小手段，常见的有Windows API检查、断点检测、时间间隔检测等等。不过，每种反调试都有相应的对策，比如可以使用Hook或patch来绕过Windows API检查！

VM虚拟机保护，这可不是我们平时说的操作系统虚拟化，而是将代码转化为难以识别的伪代码字节流，在执行时再逐条解释还原成原始代码，这样就增加了逆向的难度。

最后说到Hook，有时候静态分析看到的东西和实际运行时的情况不太一样，这是因为出题人可能用了Hook技术。这时，我们需要找到这些“钩子”，揭示程序的真实执行逻辑，然后再进行分析，喵！

逆向与反逆向就像是猫捉老鼠的游戏，双方都在不断进化中。对抗逆向的技术多种多样，每一种都有其独特之处，想要深入了解的话，就一起探索这个奇妙的世界吧，喵！^_^

算法篇

在CTF的世界里，算法逆向可是个大热门呢！让我来给你讲讲那些CTF经常出现的算法吧，喵！

首先是**Xor**，这是一种非常常见又简单的加密手段哦！异或操作有个特点，就是a^a=0，a^0=a，还有b^a^a=b。掌握了这些特性，逆向**Xor**加密就变得容易多了。

Base家族也是常客，我们在CTF里经常会碰到Base算法的实现，其中最常见的就是Base64了，有时还会遇到它的变种，像是Base64的换表版本。熟悉Base家族的各种特性，可以让我们解题时更加得心应手。

提到**Tea家族**，可能听起来像是各种茶叶的名字，但实际上是指TEA算法，这是CTF中非常常见的加密算法。我们还需要了解它的变种，比如XTEA和XXTEA。在处理TEA家族的算法时，记得检查它们有没有被魔改过，同时注意round和delta的数值。

RC4是一种面向字节的流密码，因为它是对称密码，所以只要有key，通常就能恢复数据。不过也要小心，出题人可能会对它进行一些改动。

AES和DES作为经典的对称加密算法，只要有了key，通常也能恢复数据，这些都是逆向工程中不可或缺的好伙伴。

迷宫题也是一个有趣的类别，通常会给出一个迷宫的地图，让你找出从入口到出口的路径。这时，掌握一些经典的搜索策略，比如深度优先搜索（DFS），就非常重要了，喵！

Z3不是一个算法，而是Python的一个库，它是解决复杂方程组的神器。当面对大量复杂的方程时，用Z3来解方程就再合适不过了。

当然啦，加密解密的算法远远不止上面提到的这些，还有国密算法如SM1、SM2、SM3、

SM4、SM7、SM9、ZUC，以及离散余弦变换（DCT）、salsa20、ChaCha20等等，等待着我们去探索。

技巧篇

要注意技巧！技巧啊！！章鱼哥！！—— 海绵宝宝

首先是爆破，这是一种很有用的技巧哦！当你觉得逆向工作有些吃力，但又知道你需要的结果所在的范围不是很大的时候，爆破可能是个不错的选择。虽然普通爆破会比较耗时，但如果学会了使用多进程来提高爆破的速度，效率就会大大提升啦，喵！

再来是二进制插桩，这是一种动态的分析技术，可以在不干扰程序正常执行的情况下，在程序执行过程中插入特定的分析代码，帮助我们监控和分析程序的动态执行过程。现在，Pin和Frida都是非常流行的动态二进制分析平台，可以帮助我们更好地进行这项工作。

最后要说的是模拟执行，这里不得不提的就是angr啦！angr是一个强大的多架构二进制分析平台，它支持动态符号执行以及多种静态分析能力。不仅可以用来进行爆破，如果你足够熟练的话，甚至可以用它来实现llvm的平坦化，喵！

总结

逆向工程是解开程序秘密的过程，从使用IDA Pro进行反汇编，到用x64dbg/x32dbg进行调试，再到用jadx分析Android应用，每一步都需要细致入微的操作。面对各种保护手段，如花指令、加壳、自我修改代码等，我们既要学会静态分析又要懂得动态调试。算法方面，从简单的Xor加密到复杂的迷宫题，再到解密方程组的Z3库，掌握这些工具和技巧可以让逆向之路更加顺畅。(..•˘_˘•..)

当然，不用急于求成，将上面的内容一口气消化，一步一个脚印，结合练习题，相信迟早有一天你可以成为一个逆向高手！

最后，来一个简单的题目练习一下吧！

```c
#include <stdio.h>
int main() {
 int enc[] = {44, 39, 58, 57, 21, 39, 46, 33, 45, 47, 39, 29, 54, 45, 29, 54, 42, 39, 29, 53, 114, 48, 46, 38, 29, 45, 36, 29, 48, 113, 52, 113, 48, 49, 113, 99, 63};
 char *user_input = malloc(0x100);
 printf("Enter the flag: ");
 scanf("%s", user_input);
 for (int i = 0; i < 37; i++) {
  if (enc[i] != (user_input[i] ^ 0x42)) {
   printf("Wrong!\n");
   return 0;
  }
 }
 printf("Correct!\n");
 return 0;
}
```

喵喵喵？什么，这个题目看起来是一个简单的异或解密题目，要解决这个问题，我们需要找到与enc数组中每个元素异或0x42后的值，这个值应该就是正确的输入字符。那我就好心的用python解决一下吧，以后就要靠你自己了（傲娇）：

```python
enc = [44, 39, 58, 57, 21, 39, 46, 33, 45, 47, 39, 29, 54, 45, 29, 54, 42, 39, 29, 53, 114, 48, 46, 38, 29, 45, 36, 29, 48, 113, 52, 113, 48, 49, 113, 99, 63]
key = 0x42
flag = ""
for i in range(37):
    flag += chr(enc[i] ^ key)
```

```
6. print(flag)
```