reverse专题 - 1

f0rm2l1n

闲(尬)聊一下

- 一杯茶、一包~~烟~~薯片、一个逆向做一天
- 密码学、数学知识；学无止境
- 大量可用工具；学无止境 plus
- 与开发联系紧密
  - 语言
  - 平台
  - 架构

上周基础讲了的一些 + 没讲的一些

- 预处理、编译、汇编、链接细节
- x32dbg/x64dbg
  - Ollydbg
- self modifier code (smc)
  - 这样异或代码不会出事么 ☹
  - 编译器的区别，symbol resolving and relocation

C program is so native

有趣的例子 1 – symbol confliction

what will main print

```c
#include <stdio.h>

void func();

char x = 'a';
char y = 'b';

int main()
{
    func();
    printf("%c%c\n", x, y);
}
```

main.c

```c
int x;

void func()
{
    x = 0x293b;
}
```

func.c

OKAY 开始逆向专题

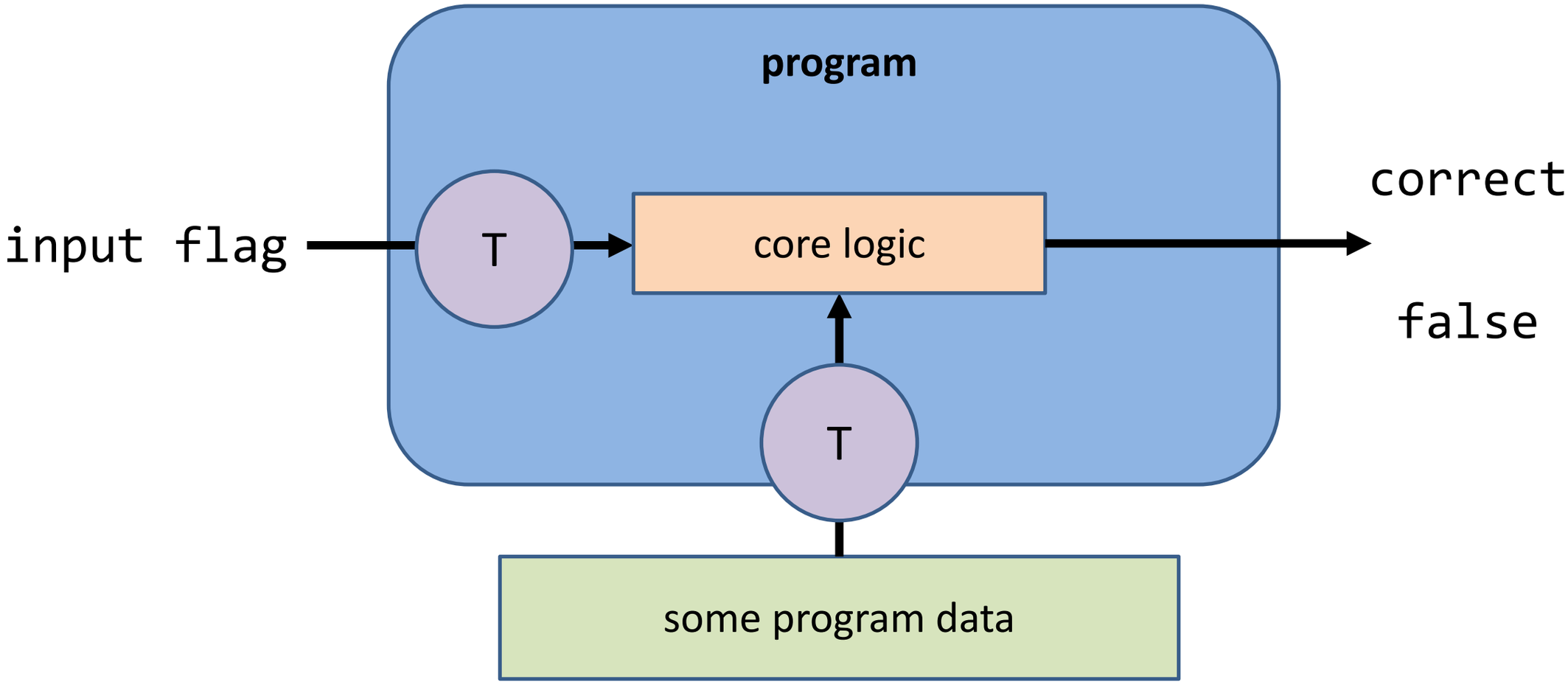
- ~~破解真实软件~~
  - 看雪论坛 https://www.kanxue.com/
  - 吾爱破解 https://www.52pojie.cn/
  - IOSRE https://iosre.com/

- CTF解题

好的目标 + 足够的投入以及耐心

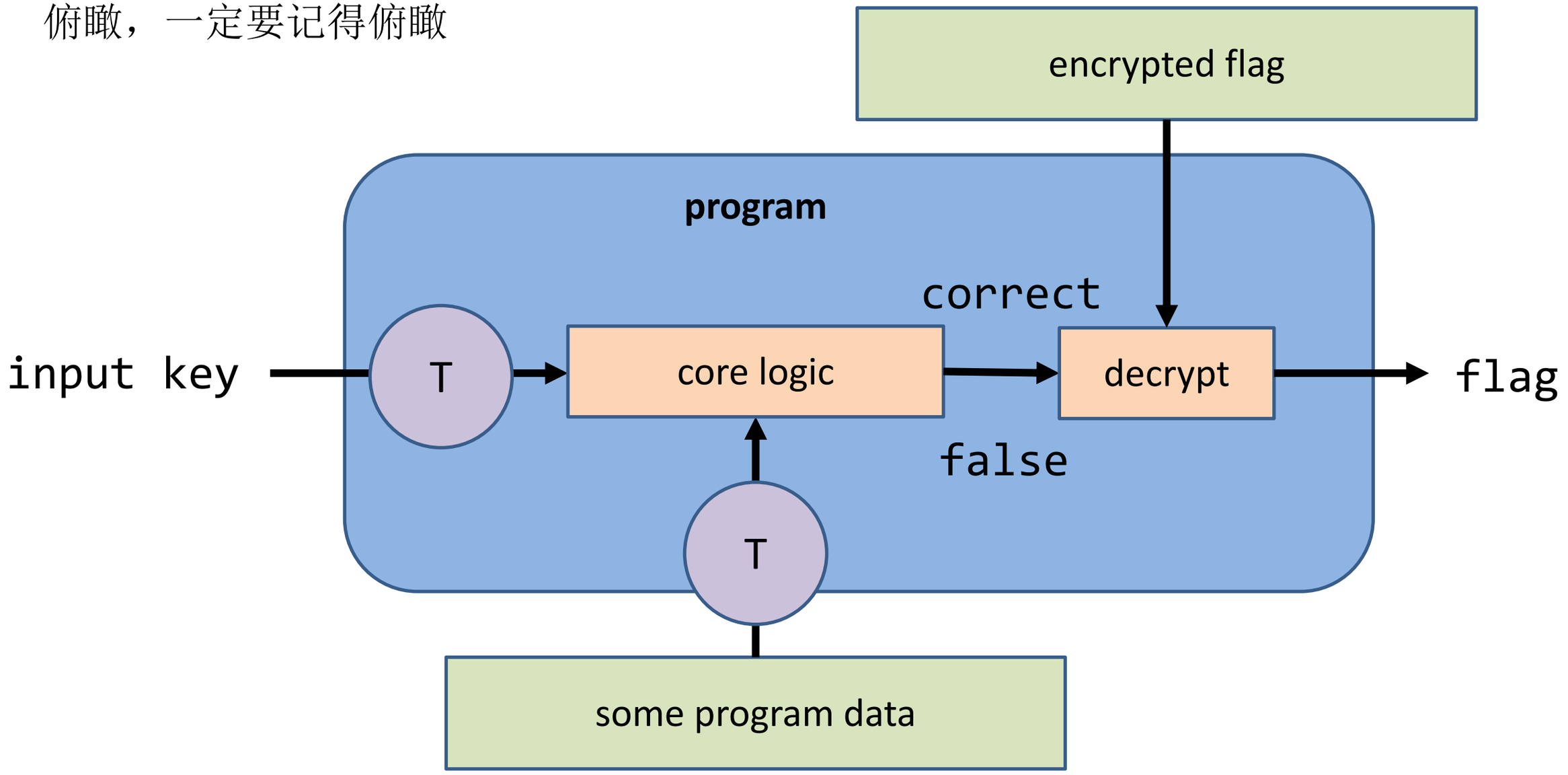
俯瞰一下逆向的赛题 - 1

俯瞰，一定要记得俯瞰



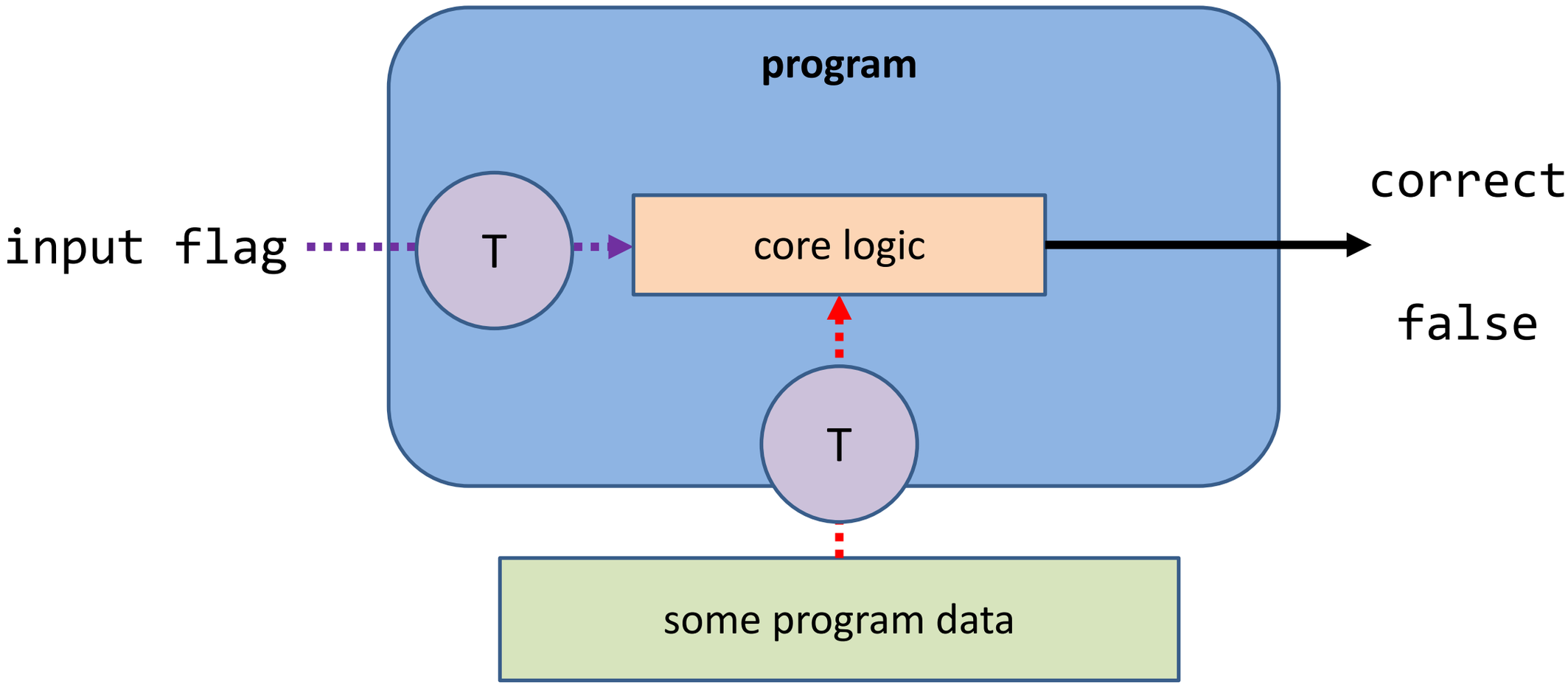
俯瞰一下逆向的赛题 - 2

俯瞰，一定要记得俯瞰



core chain and useless chain

俯瞰，一定要记得俯瞰



Outline

针对可执行程序的“综合”保护策略

- 壳
- 虚拟机

成熟的加壳软件和虚拟机保护软件可以接受一个 可执行程序 作为输入，并将保护过的可执行程序作为输出

依赖少，使用方便

加壳和脱壳 (packing and unpacking)

- 什么是壳

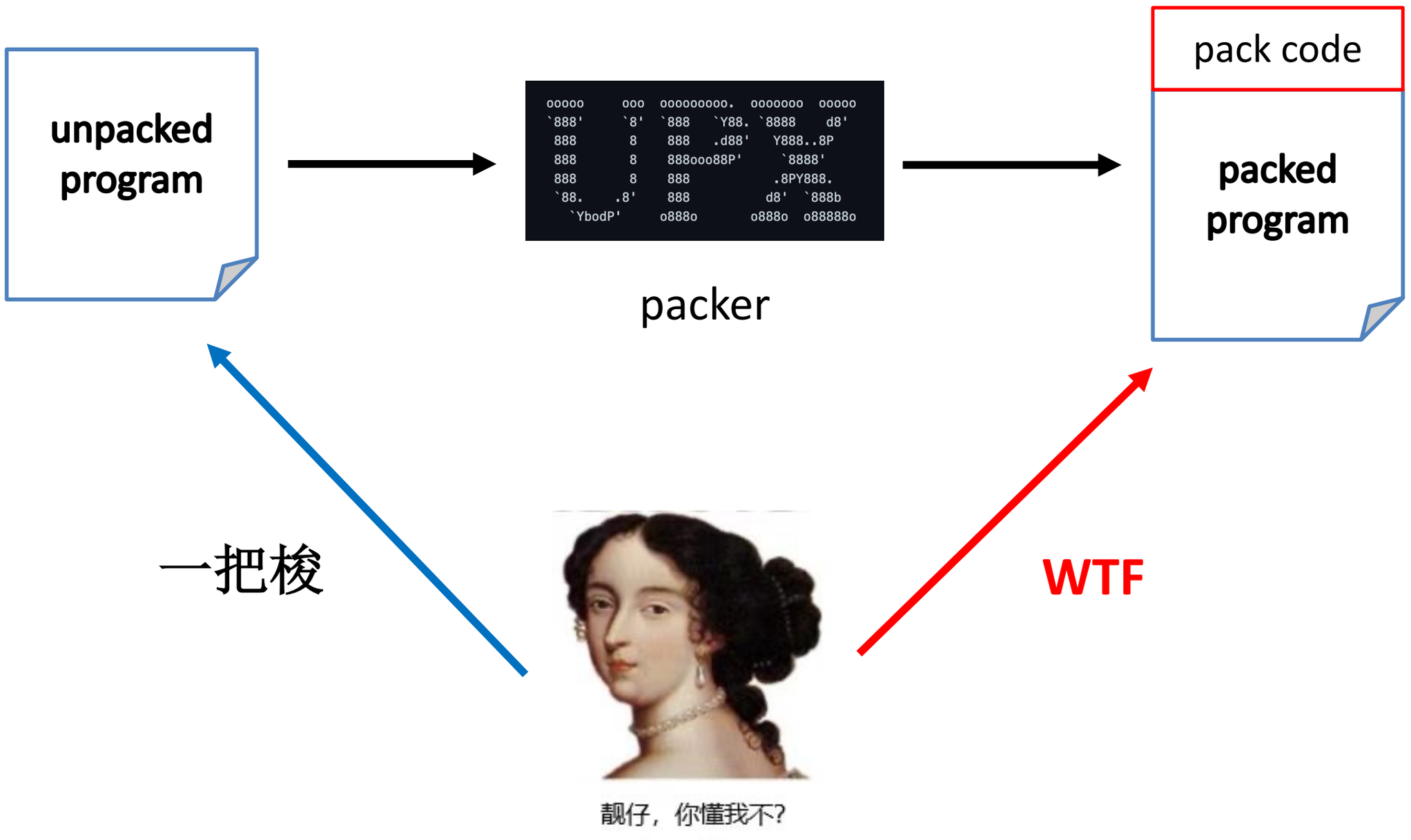
在自然界中，我想大家对壳这个东西应该都不会陌生了。植物用它保护种子、动物用它保护身体等等。同样，在一些计算机软件里也**有一段专门负责保护软件不被非法修改或者反编译的程序。它们一般先于（被保护的）程序运行，拿到控制权，然后完成它们保护软件的任务**。就像动植物的壳一般都是在身体外面一样理所当然。由于这段程序和自然界中的壳在功能上有很多相同的地方，基于命名的规则，大家把这样的程序称为“壳”了。就像计算机的病毒和自然界中的病毒，其实都是命名的方法罢了。

——《一切从“壳”说起》

blackwhite prefers packing

这道题外面有一层强壳，虽然不是我原创，但是这层壳本身是我从一个商用软件中提取并改造过的，所以并非完全借用。这个强壳的作者是**杨道沅教授**，他是中国加密界的元老，也是**Lock89**的作者。我用了他的壳主要是为了向他致敬，像**雷军的bitlok**之流跟杨先生的作品比较那是差了至少10条街的距离。

简单来说 和 一些个人看法



one tiny program

```c
#include <stdio.h>
#include <stdlib.h>

#include <unistd.h>

char password[] = "VeRY_VERy_h4RD_aNd_S3curE_pA5SWoRD";

int main(int argc, char* argv[])
{
    char input[64] = {0};

    read(STDIN_FILENO, input, 34);

    if (strcmp(password, input)) {
        printf("error\n");
    } else {
        printf("ok\n");
    }

    return 0;
}
```

壳介绍和分析1 – dropper

https://github.com/marcusbotacin/Dropper

- 或许应该叫，dump 类型的壳，“保留”了被保护可执行文件的完整性

1. demo 使用
2. 逆向比较
3. 源码分析
4. 加强版
   ACTF Dropper
   https://gist.github.com/valinet/e27e64927db330b808c3a714c5165b0a
5. 脱壳，很好脱吧

壳介绍和分析2 – UPX

UPX is a free, portable, extendable, high-performance **executable packer** for several executable formats.

- 压缩壳的代表

https://upx.github.io/

https://en.wikipedia.org/wiki/UPX

1. 运行 demo (linux)
2. UPX 原理，动态调试
3. upx壳的识别和脱壳

休息一下吧

思考有没有可能修改 UPX 成更恶心的壳

加密壳的简单实现

目标，自行实现一个简单加密壳，完成对 tiny program 的保护

加密壳的简单实现

有如下 concerns

1. 如何在进入 main 之前拿到代码执行能力
2. 如何在代码执行能力中完成去壳
3. 如何 patch 编译好的程序

加密壳的简单实现

有如下 concerns

1. 如何在进入 main 之前拿到代码执行能力

2. 如何在代码执行能力中完成去壳

~~3. 如何 patch 编译好的程序~~

对于一个已经编译好的binary
添加 section (壳代码)
添加 segment
更改 ELF entry-point address
.....

加密壳的简单实现

有如下 concerns

- 如何在进入 main 之前拿到代码执行能力
  - 完全自己写 _start，接管程序开始的执行部分 .....
  - 在链接前准备好 object file，这样子至少不用自己 patch section 和 segment；链接完成后在修改 entrypoint 并 fix 一些跳转 .....
  - .init_array 黑魔法

ELF中的 .init_array

This section holds an array of **function pointers** that contributes to a single **initialization array for the executable or shared object containing the section.**

.init_array .fini_array demo

ctor.cpp

简单加密壳伪代码 （作业会用的哦

tiny_pack.cpp

调试时候的有趣现象 ☺

b main
结果到 main 函数后发现指令乱了 ???

why? 再次理解软断点原理

通用方法 – 通过动态调试进行脱壳

壳本身可能会异常复杂，完全理解其中的解密逻辑需要时间

why bothers

- 既然这个程序能跑，而且壳代码在运行时会完成自解密，那就让它解密后，我们再获取runtime程序映像就好～

For Linux

1. gef 插件支持 dump 命令，可以将解密部分字节转储

2. 直接利用 core 转储整个程序 `generate-core-file`

example: iOS 砸壳

- 完成越狱的 iPhone (checkra1n)
- 配置 Frida server
- 手机运行程序后 host 端通过 Frida 进行 dump

路给堵死

1. 反调试手段
   ptrace自己、查看栈、checksum检查软断点、ps 查看进程、查看 /proc/self/cmd … 数不胜数
   demo: anti_trace

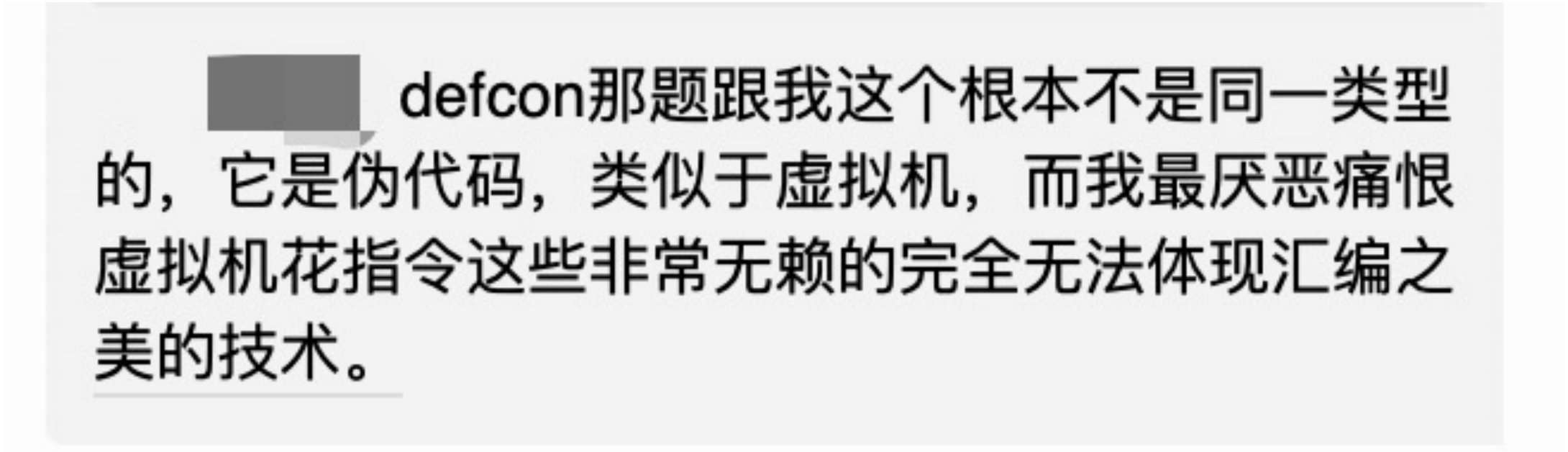
2. 细粒度壳代码
   每次只解密一个函数，而不是一口气全部完成解密

   demo: Defcon QUALS Secrecy

休息一下吧

虚拟机保护

P.S. blackwhite dislike VM protection



代表软件 VMProtect (VMP)

personal versions. For the more detailed comparison see the feature matrix, the full price list is here.

Ultimate - all-in-one software protection solution. Has the full set of features and supports serial numbers.

$499 buy now

Professional - for those, who doesn't need serial numbers. An ideal solution for analysis prevention.

$249 buy now

Lite - limited, yet functional entry-level edition. Allows to protect software at bargain price.

$149 buy now

虚拟机保护原理

- 重要操作虚拟化

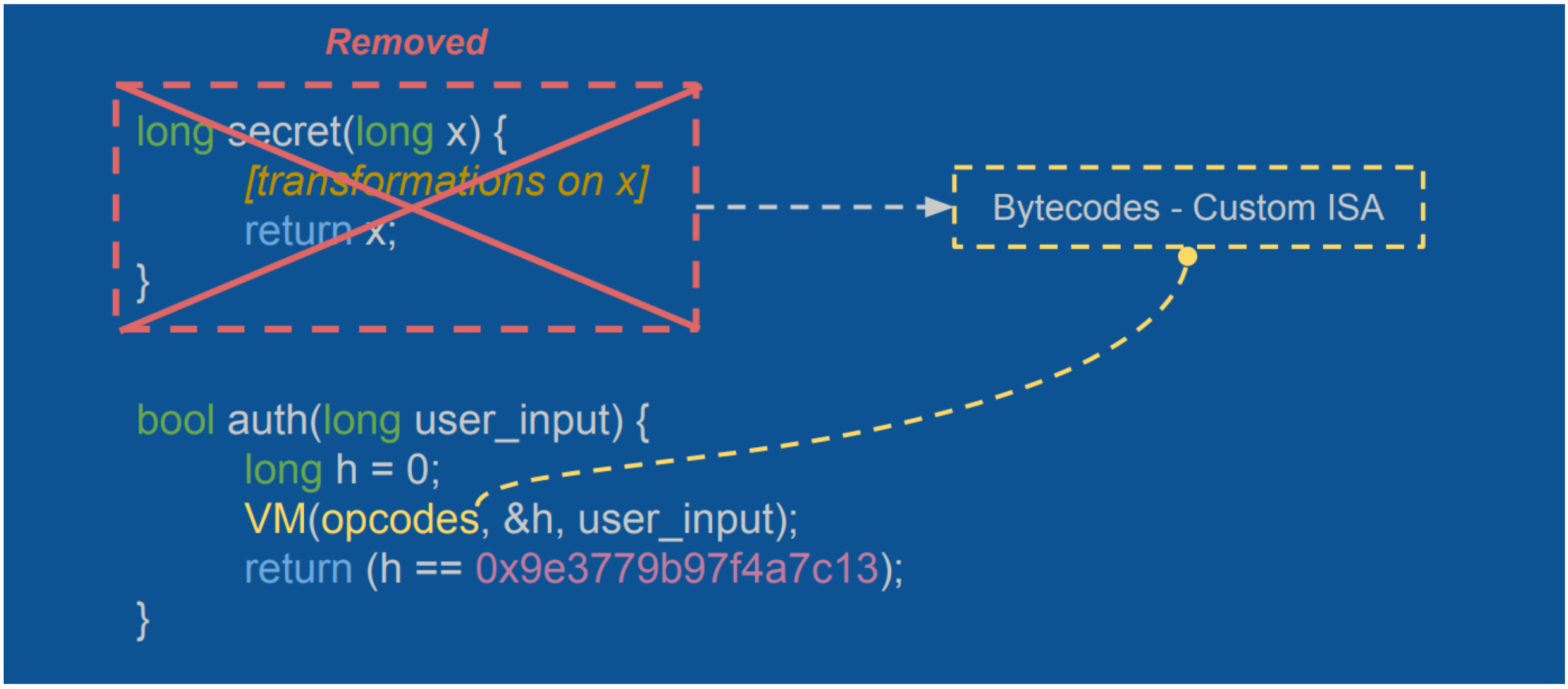
```
long secret(long x) {
        [transformations on x]
        return x;
}

bool auth(long user_input) {
        long h = secret(user_input);
        return (h == 0x9e3779b97f4a7c13);
}
```

credit to Deobfuscation of VM based software protection, SSTIC 2017, quarklab

```
long secret(long x) {
        [transformations on x]
        return x;
}
```

Bytecodes - Custom ISA

```
bool auth(long user_input) {
        long h = secret(user_input);
        return (h == 0x9e3779b97f4a7c13);
}
```


Removed
long secret(long x) {
[transformations on x]
return x;
}
Bytecodes - Custom ISA
bool auth(long user_input) {
long h = 0;
VM(opcodes, &h, user_input);
return (h == 0x9e3779b97f4a7c13);
}

demo of ***vmprotect***

https://github.com/eaglx/VMPROTECT

Take a peek of a real VM by hammou

https://gist.github.com/SouhailHammou/6b4069b07e538a48bb70

- from source code

to

- how to reverse

写 de-assembler 咯

VM reverse is so tedious

especially …

当 VM 程序和你不熟悉的语言、平台结合的时候

- revvm by hxpCTF 2021

甚至，VM模拟的架构都是你不熟悉的

- Manchester architecture

解决办法

- dumb one

找到译码逻辑；
逆向所有的opcode，还原虚拟机逻辑；
写 decompiler 将 vm code 转化为熟悉的 code
完全逆清楚

- clever one

trace based 分析
更牛掰的技术？

研究前沿

自动去虚拟化: **auto de-virtualization**

- http://tigress.cs.arizona.edu/challenges.html
- https://github.com/JonathanSalwan/Tigress_protection

Summary

- 复习了一点点二进制“基础”

- 壳保护，简单的脱壳方法
  - 特定壳用特定壳的脱发
  - 动态调试一把梭

- 虚拟机保护
  - 举例

About homework

基础：

1. 完成简单加密壳

2. 完成简单栈虚拟机

挑战：

- 实现 linux 上的 dropper

详见报告