Beijing Forest Studio
北京理工大学信息系统及安全对抗实验中心

基于深度学习的恶意软件检测

静态分析和反检测技术

李玉　硕士研究生

2020年09月20日

内容提要

- 背景简介
- 基本概念
- 恶意软件检测发展历史
- 算法原理
- 反检测技术
- 参考文献

背景简介

- 预期收获
  - 1. 了解恶意软件检测发展历史和现状
  - 2. 了解深度学习在恶意软件检测上的应用
  - 3. 了解恶意软件常用反检测方法和对抗手段

背景简介

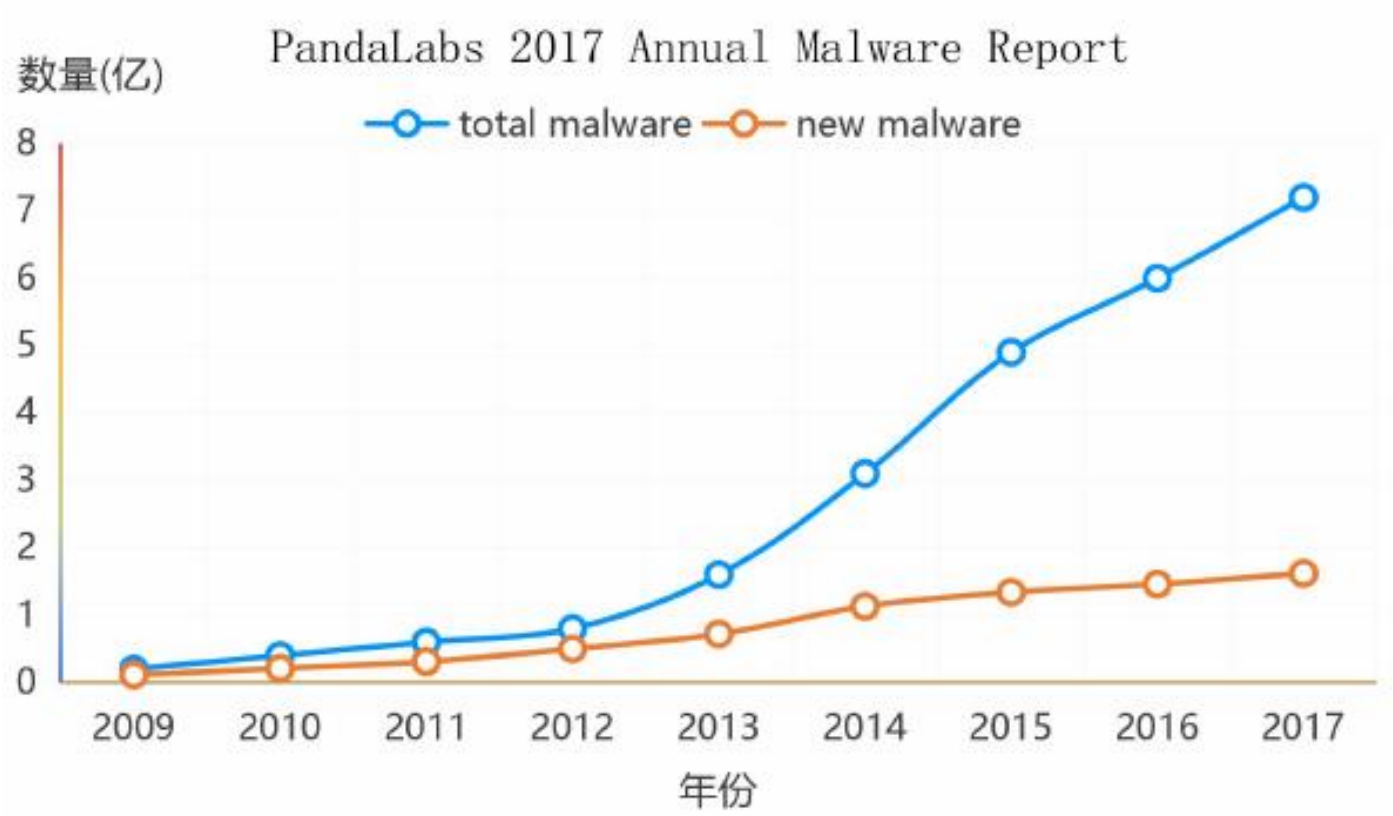
- 为什么要检测恶意软件?
  - 危害大
  - 数量多

背景简介

- 2015年12月，名为“暗黑能量”的病毒造成乌克兰大面积停电，电力系统瘫痪一个小时
- 2017年Wannacry大规模爆发，造成直接经济损失80亿

背景简介

- 恶意软件数量呈爆发式增长
  - 自2014年起，每年新增恶意软件数量超过1亿
  - 至2017年，已知的恶意软件总数达到8亿



基本概念

基本概念

- 恶意软件
  - 在未经用户授权的情况下，非法侵入用户计算机或其他终端，严重侵害用户个人利益的软件。

- 形式多样：
  - 病毒
  - 木马
  - 蠕虫
  - 勒索软件
  - …

基本概念

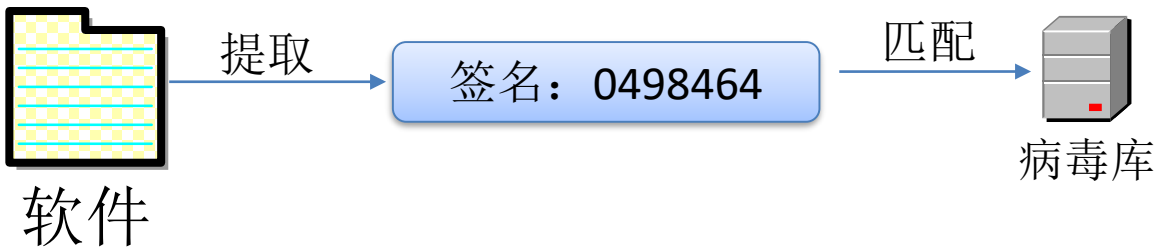
- 常见恶意软件分类：
  - 病毒：一段可以附加到其他程序中的代码，无法独立运行，需要宿主程序激活（一旦中招，难以清除）
  - 蠕虫：能够独立运行，可以自我复制；通常利用自我复制性大量占用网络和系统资源，影响网站的访问和系统的运行（Iloveyou，导致电话系统过载，电视宕机;熊猫烧香）
  - 木马：伪装成合法程序，但运行时执行恶意行为
  - 间谍软件：非法收集用户信息，监视用户活动
  - 勒索软件：对文件加密，通过勒索赎金获利
  - （Wannacry）

基本概念

- 传统的恶意软件检测方法：
  - 基于签名检测

- 什么是签名？
  - 一段能够唯一标识程序的特征码，通常用哈希值表示

- 根据签名怎么检测？
  - 在本地建立病毒库，包含已知的恶意软件的签名；通过签名匹配，判断是否为恶意软件



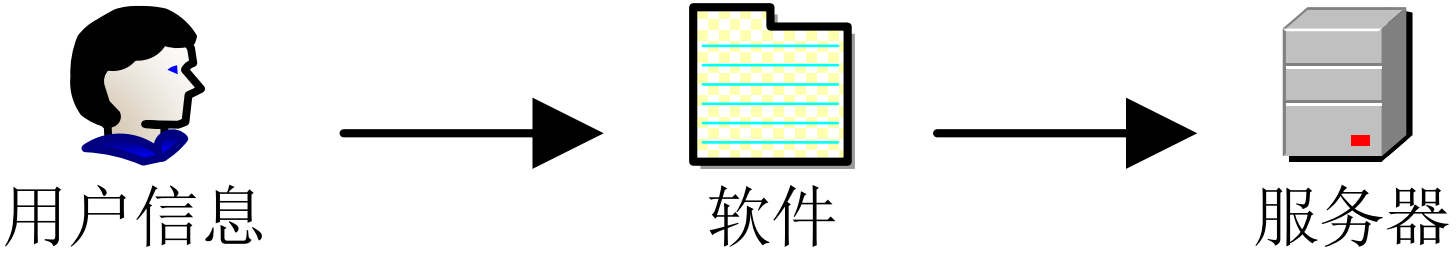
基本概念

- 基于签名检测：
  - 优点：
    - 速度快，效率高 ——不需要额外的分析，只需要提取签名并匹配
  - 缺点：
    - 无法检测未知的恶意软件
    - 病毒库数据爆炸
    - 病毒库的更新依赖专家手工分析，且更新速度慢，一个恶意软件从发现到病毒库更新平均时间为54天

基本概念

- 如何改进？
  - 为什么不能检测未知的？——因为通过签名精准匹配检测
  - 为什么必须要精准匹配？——因为签名的生成和恶意软件类别没有关系，只能表示当前恶意软件的特征，不能够表示某一类恶意软件的特征
- 改进思路：
  - 找到一种能够表示一类恶意软件的特征（泛化能力强）——“行为”

获取用户信息->连接远程服务器->发送信息



基本概念

- 基于启发式检测：
  - 通过捕获软件行为判断恶意性
  - 如何捕获软件行为：
    - 静态分析：不执行文件，从二进制文件或源代码中提取特征
      - 优点：代码覆盖率高、速度快
      - 缺点：依赖于二进制文件和源代码，易受到代码混淆影响
    - 动态分析：在可靠的虚拟环境中运行样本，获取样本运行过程中调用的API等信息
      - 优点：不受代码混淆影响
      - 缺点：耗时耗资源、代码覆盖率低，难以覆盖到程序的全部分支

基本概念

- 静态分析常用的特征包括：
  - 汇编指令：程序在操作系统层面的行为
    - 每一条汇编指令包含一个操作码和两个操作数
    - 操作码表示指令的操作类型
    - 操作数表示指令的操作对象

```
mov   eax , 0x01
```

操作码	目的操作数	源操作数

  - 自定义的函数：函数名通常表示该函数的功能
  - 字符串（变量名）：表示变量的用途

算法原理

算法原理

- 基于深度学习的恶意软件检测

T	对PC端可执行文件进行分析，判断是否为恶意软件
I	PE文件（10868个）
P	1.预处理 2.特征图像构建 3.深度学习模型分类
O	PE文件是否为恶意文件（准确率99%）

P	静态分析技术易受到代码混淆的影响
C	具备大量有标签的PE样本
D	识别恶意指令/正常指令
L	Computers & Security

算法原理

- 经典的基于汇编指令的检测算法流程：


```
push   esi
lea    eax          [ esp + 8 ]
mov    esi          ecx
mov    eax          0x00405B74
call   sub_41c8c9
```

```
push
lea
mov
mov
call
```

```
1 0 0 1
0 0 1 0
0 1 0 0
0 1 0 0
0 1 1 0
```

汇编代码　　操作码序列　　向量表示　　特征图像

- 删除操作数的原因：操作数的含义与运行环境密切相关，同一个操作数在不同的运行环境下含义不同，因此不能确定操作数的含义，属于噪声数据
- 地址0x00401B59在样本1中表示A函数，在样本2中表示B函数

算法原理

- 存在的问题：
  - 去除操作数后，指令行为描述不完整，无法抵抗花指令和代码混淆的干扰

```
push   esi
lea    eax          [ esp + 8 ]
mov    esi          ecx
mov    eax          0x00405B74
call   sub_41c8c9
```

→

```
push
lea
mov
mov
call
```

加入干扰指令

```
push   esi
lea    eax          [ esp + 8 ]
push   eax
pop    eax
add    eax          0x01
sub    eax          0x01
mov    esi          ecx
mov    eax          0x00405B74
call   sub_41c8c9
```

→

```
push
lea
push
pop
add
sub
mov
mov
call
```

算法原理

- 解决办法：
  - 尽管操作数含义随运行环境而变化，但是操作数的类型是不变的，与运行环境无关，因此将操作数类型抽象出来作为特征，可以有效利用操作数信息，抵抗代码混淆的干扰

  - 将操作数抽象成如下五类：
    - MEM：内存地址，通常用“[ ]”修饰，如[eax+0x01]
    - REG：寄存器，包括专用寄存器和通用寄存器，如eax
    - IMM：立即数，一个数值，如0x40
    - MARK：字符串，包括函数名、跳转地址，如sub_xx
    - NULL：空占位符，表示该位置无操作数

算法原理

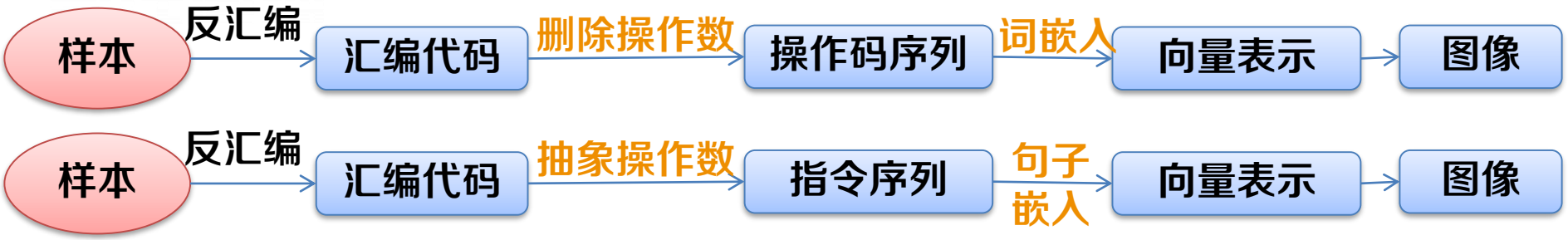
- 效果说明：
  - 能一定程度抵抗代码混淆和无关指令的干扰

```
push   esi
lea    eax          [ esp + 8 ]
push   eax
pop    eax
add    eax          0x01
sub    eax          0x01
mov    esi          ecx
mov    eax          0x00405B74
call   sub_41c8c9
```



```
push   REG    NULL
lea    REG    MEM
push   REG    NULL
pop    REG    NULL
add    REG    IMM
sub    REG    IMM
mov    REG    REG
mov    REG    IMM
call   MARK   NULL
```

算法原理



- 嵌入（指令向量化）
  - 经典方法中，对操作码做词嵌入，提取操作码的上下文信息
  - 我们的方法中，将每条指令视为一个句子，采用句子嵌入的方法，保留指令上下文信息的同时，还能学习指令内部各元素之间的关系

——操作码和操作数如何构成一条指令

```
push
lea
mov
mov
call
```

```
push   REG   NULL
lea    REG   MEM
push   REG   NULL
pop    REG   NULL
```

算法原理

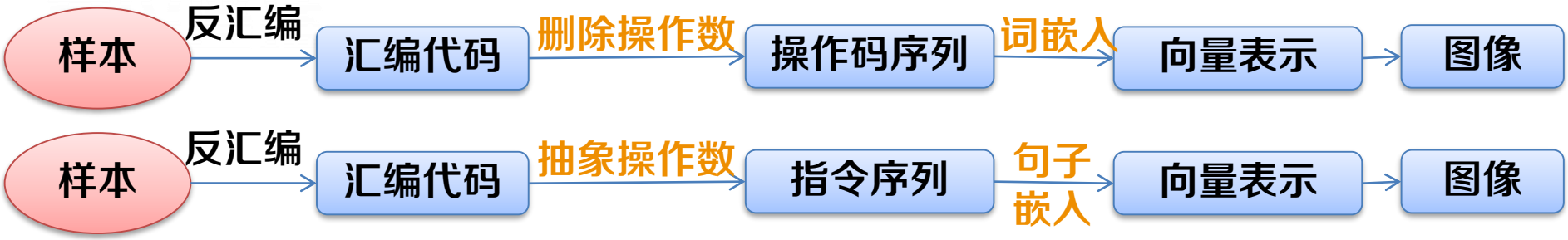
- 句子嵌入：
  - P-means：关注单词信息
    - 操作码和操作数各自的含义

```
push    REG     NULL
lea     REG     MEM
push    REG     NULL
pop     REG     NULL
```

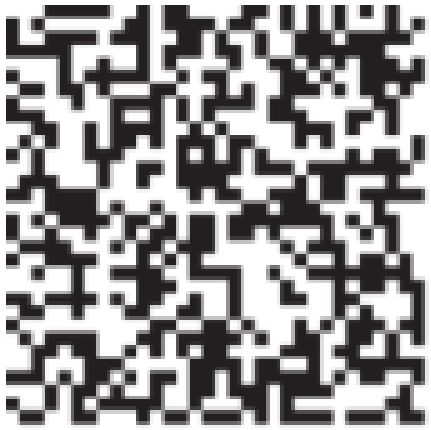
  - SIF：关注单词之间的关系
    - 操作码和操作数是如何构成一条指令的
  - Paragraph2vec：关注句子间的关系
    - 各个指令是如何构成一个软件的

网络安全-学术报告-无监督句子嵌入方法-李玉-2019.11.24

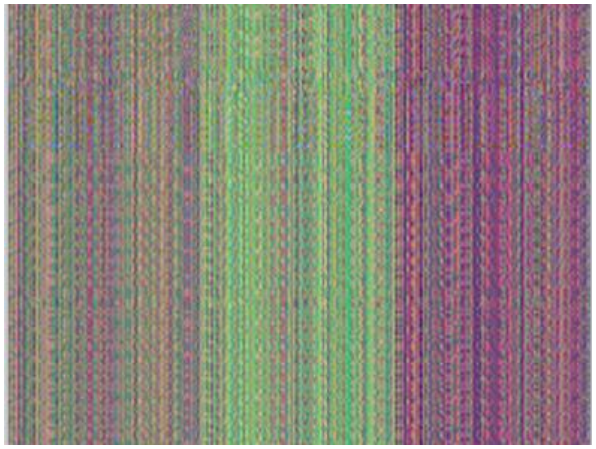
算法原理



- 生成图像效果展示：



经典方法



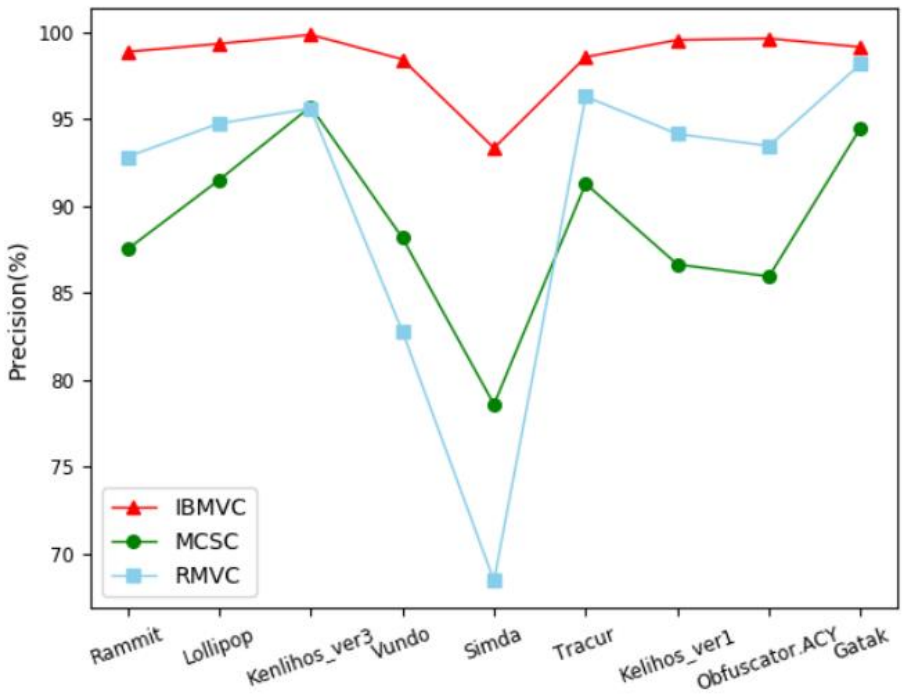
改进方法

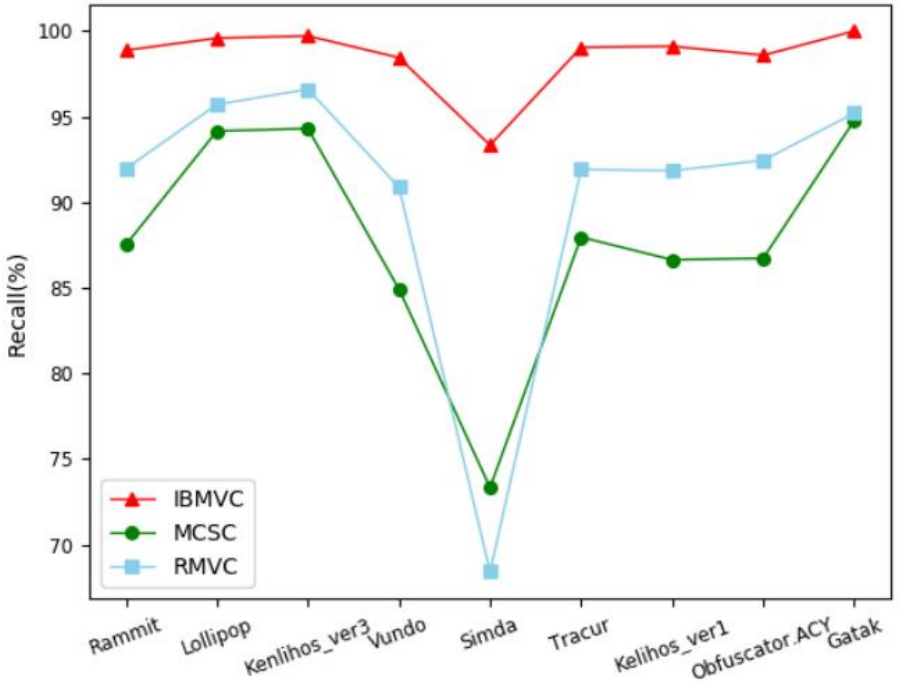
算法原理

- 实验结果：

Table 12 Experiment results for the comparison

Methods	Accuracy(%)	Precision(%)	Recall(%)	F1 score
MCSC[4]	91.49	88.87	87.81	88.32
RMVC[3]	94.20	95.92	89.07	91.00
IBVMC	**99.38**	**98.53**	**98.52**	**95.52**





算法原理

- 创新点总结：
  - 1. 通过抽象出操作数类型，使操作数的含义脱离运行环境的约束，从而合理的利用部分操作数信息，增加了对代码混淆和花指令的抵抗能力
  - 2. 从句子角度对指令进行向量化，相比于关注操作码，我们更关注指令内部各元素之间的关系，即操作码和操作数如何构成一条指令，保留更多的行为信息

反检测技术

反检测技术

- 对运行环境的检测
  - 是否处于沙箱等虚拟环境中
  - 是否被下断点，处于调试状态

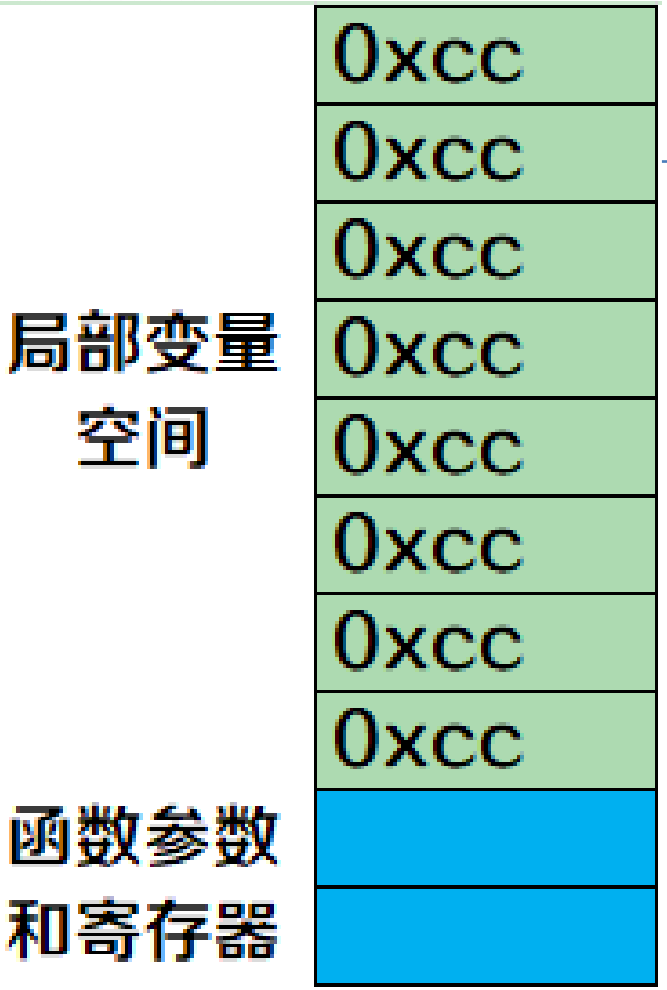
- 对恶意软件自身的保护：
  - 对内部数据加密，难以分析和破解

反检测技术

- 对运行环境的检测
  - 1. 虚拟环境检测
    - 执行特权指令：普通操作系统会因权限不足而报错，但是沙箱等处于保护模式下的虚拟环境会正常执行
  - 2. 反调试
    - 断点检测： 0xcc　INT3中断 （INT=interrupt≠int）

```
Please enter string S:ABCDEFG123456

The result is: 246
Press any key to continue
```

反检测技术



INT 3中断

Unicode编码：烫

为了防止编译器把栈上的内容当做指令来执行

反检测技术

- 对运行环境的检测
  - 2. 反调试
    - 断点检测
    - 校验和检测：加入断点后，指令内容变成0xcc，程序的校验和会发生变化

反检测技术

- 对运行环境检测
  - 2. 反调试
    - 断点检测
    - 校验和检测
    - 花指令：
      - 可执行的花指令：无意义的跳转，不影响反汇编结果，只是干扰分析
      - 不可执行的花指令

```
push   esi
lea    eax          [ esp + 8 ]
mov    esi          ecx
mov    eax          0x00405B74
call   sub_41c8c9
```

加入可执行花指令

```
push   esi
lea    eax          [ esp + 8 ]
push   eax
pop    eax
add    eax          0x01
sub    eax          0x01
mov    esi          ecx
mov    eax          0x00405B74
call   sub_41c8c9
```

反检测技术

- 不可执行的花指令
  - 利用了反汇编引擎仅根据机器码进行反汇编的漏洞

```
start:    mov     ax      0B46
          mov     es      ax
          mov     ds      ax
          mov     ax      0B47
          mov     ss      ax

          jmp     A
          dw      1h

A:        mov     ax      1
          add     ax      1
```

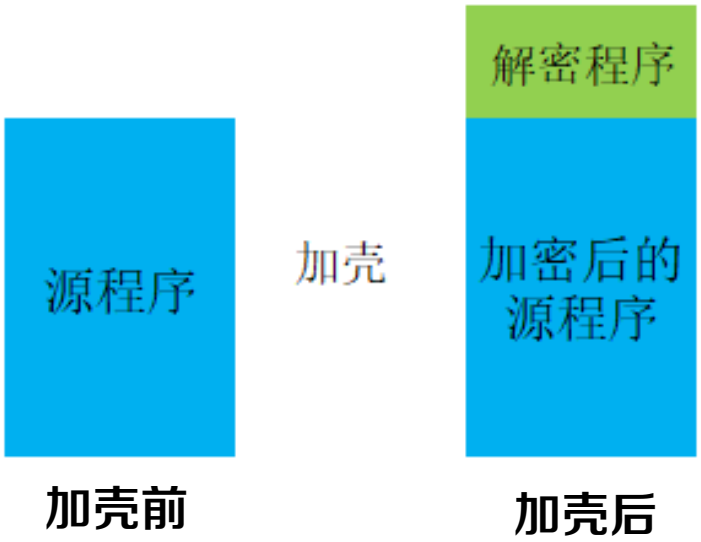
程序实际汇编代码

```
start:    mov     ax      0B46
          mov     es      ax
          mov     ds      ax
          mov     ax      0B47
          mov     ss      ax

          jmp     000F
          add     bx+si+1 di

A:        sub     ax      1
          push    1370
```

反汇编代码

反检测技术

- 对自身的保护
  - 1. 时序逃避：规定每天10点执行恶意行为
  - 2. 加壳（加密）



  - 3. 代码混淆
    - 打乱代码块，通过jmp确保程序正常执行，干扰分析

参考文献

- Ni S, Qian Q, Zhang R. Malware identification using visualization images and deep learning [J]. Computers & Security, 2018. 77: p. 871-885.

- Sun G, Qian Q, Deep Learning and Visualization for Identifying Malware Families [J]. IEEE Transactions on Dependable and Secure Computing, 2018: p. 1-1.

知人者智，自知者明。胜人者有力，自胜者强。知足者富。强行者有志。不失其所者久。死而不亡者，寿。

谢谢！