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

二进制程序中加解密函数的定位

焦龙龙 研究生

2018年06月18日

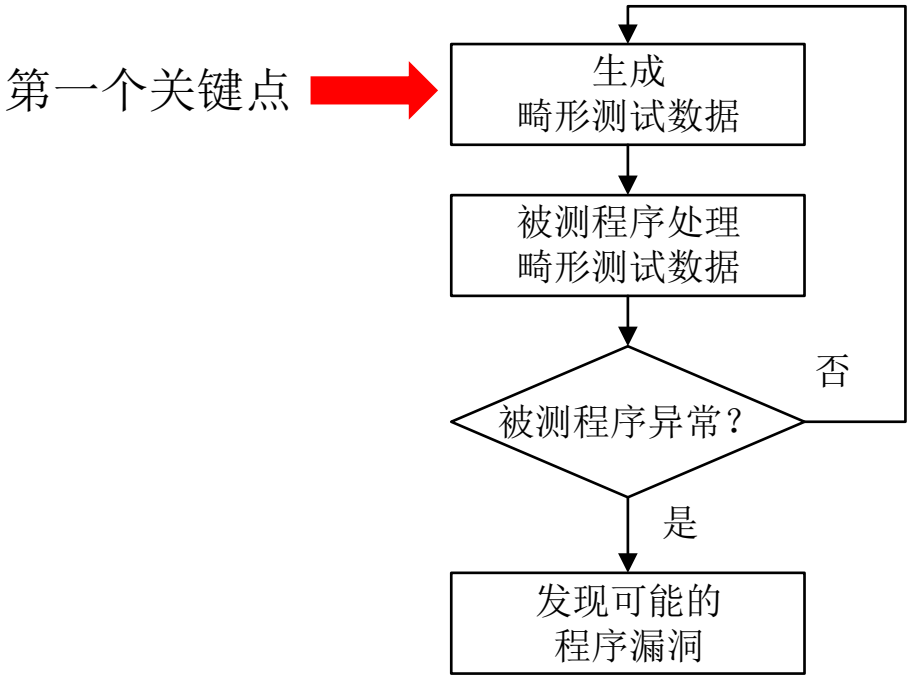
内容提要

- 背景简介
- 基本概念
- 加解密函数定位
  - 人工定位
  - 流量分析
  - 静态分析
  - 动态分析
- 优劣分析
- 应用总结
- 参考文献

背景简介

背景简介

- 模糊测试是一种广泛使用的自动化漏洞挖掘方法
- 基本原理
  - 对原始测试数据进行变异生成畸形测试数据
  - 监控被测程序处理畸形测试数据的过程
  - 被测程序出现异常行为时就可能发现了漏洞



背景简介

- 通过变异的方式生成测试数据

- 把模糊测试的起点设置为加解密操作完成之后
  - 需要进行加解密函数的定位

基本概念

基本概念

- 加密
  - 将明文数据按某种算法进行处理，使之成为不可读的密文
- 解密
  - 加密的逆过程，即将密文还原为明文的过程

明文

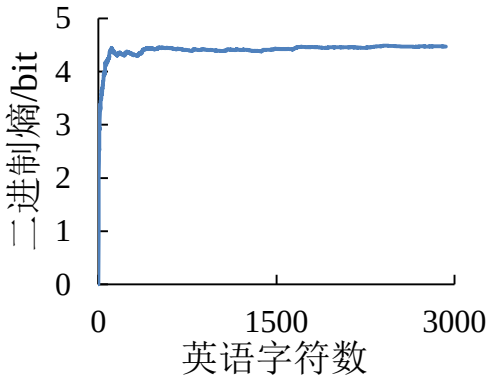
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

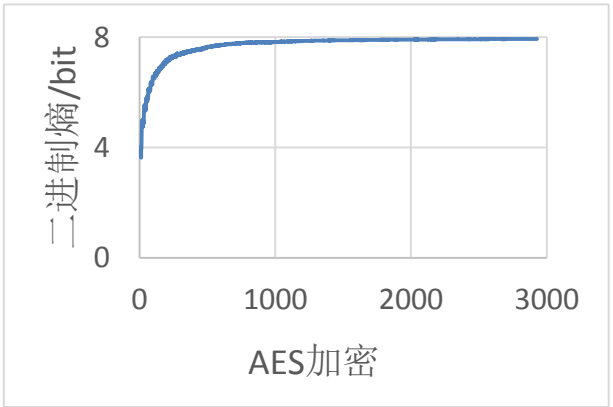
AES加密

AES解密

密文

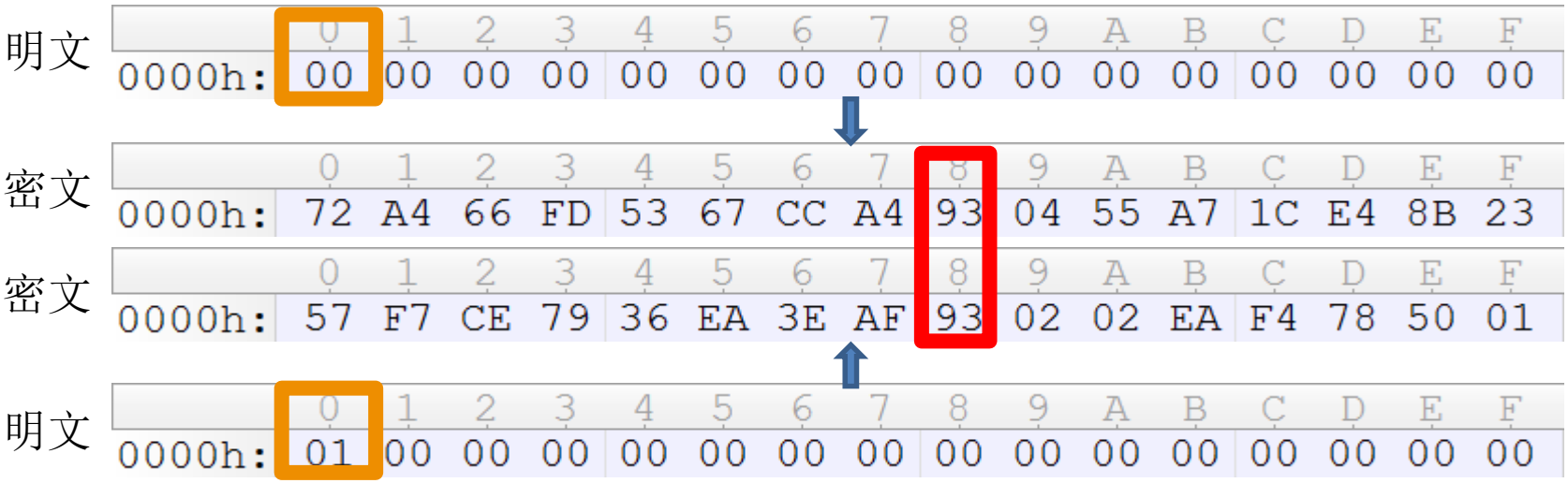
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000h:	72	A4	66	FD	53	67	CC	A4	93	04	55	A7	1C	E4	8B	23



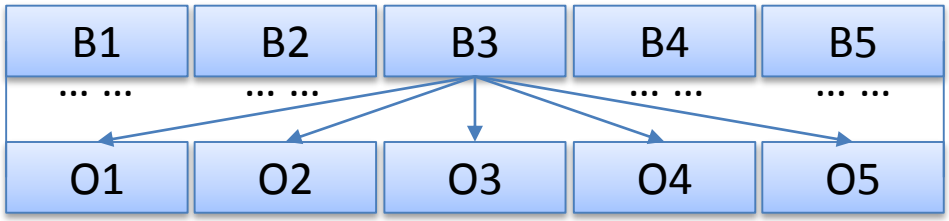


基本概念

- 一个加密算法应该具有良好的扩散性和混乱性
- 扩散性：明文数据的每一个微小变化都应该影响到密文中尽可能多的数据，以隐藏明文数据的统计特征

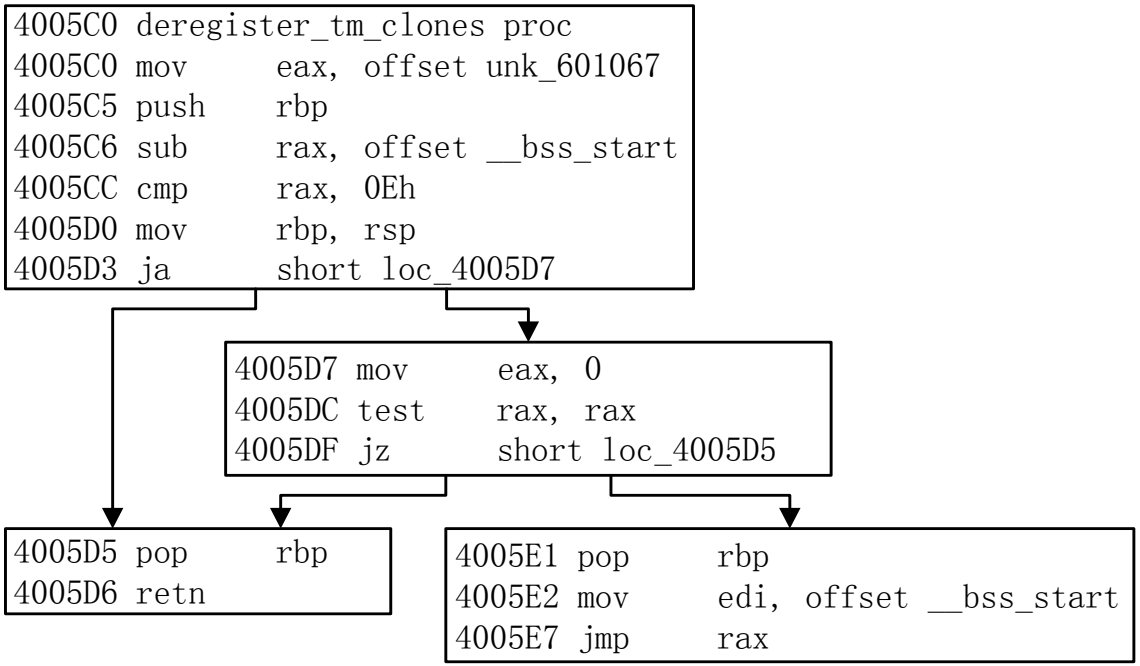


- 混乱性：明文数据在使用密钥进行加密之前，应该经过一个可逆的计算进行“混合”



基本概念

- 基本块（Basic Block，BBL）
  - 二进制程序可以划分成很多的基本块
  - 每一个基本块都是一个没有分支的代码指令序列
  - 一个基本块只会有一个入口和一个出口



加解密函数定位

加解密函数定位

- 加解密函数
  - 执行加密或解密过程的函数
- 加解密函数定位方法
  - 人工定位
  - 流量分析
  - 静态分析
  - 动态分析

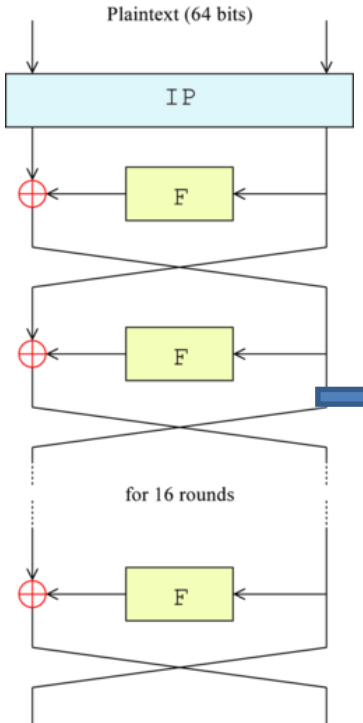
DES加密函数的部分代码

```
lea     rax, [rsp+70h+var_58]
lea     r13, _ZN5Botan10DES_IPTAB1E ; Botan::DES_
lea     r14, _ZN5Botan10DES_FPTAB1E ; Botan::DES_
lea     rbp, [rsp+70h+var_48]
mov     rbx, rdi
mov     r15, rsi
mov     r10, rdx
xor     r12d, r12d
mov     [rsp+70h+var_70], rax
nop     dword ptr [rax+00h]
```

```
loc_4EBB98:
movzx   edx, byte ptr [r15]
movzx   eax, byte ptr [r15+7]
mov     rsi, rbp
lea     rcx, _ZN5Botan10DES_IPTAB2E ; Botan::DES_IPTAB2
mov     rdi, [rsp+70h+var_70]
mov     [rsp+70h+var_68], r10
mov     rax, [rcx+rax*8]
or      rax, [r13+rdx*8+0]
movzx   edx, byte ptr [r15+1]
mov     rdx, [r13+rdx*8+0]
add     rdx, rdx
or      rax, rdx
movzx   edx, byte ptr [r15+2]
mov     rdx, [r13+rdx*8+0]
shl     rdx, 2
```

加解密函数定位 人工定位

- 人工定位法
  - 专业的分析人员利用其拥有的经验和专业知识，对程序和程序的输入数据进行分析，进而确定加解密函数的位置
- 劣势
  - 专业的分析人员耗费大量的时间才能完成加解密函数定位




```
for(int i=0; i<64; ++i)
    currentBits[63-i] = plain[64-IP[i]];
for(int i=32; i<64; ++i)
    left[i-32] = currentBits[i];
for(int i=0; i<32; ++i)
    right[i] = currentBits[i];
for(int round=0; round<16; ++round)
{
    newLeft = right;
    right = left ^ f(right,subKey[round]);
    left = newLeft;
}
for(int i=0; i<32; ++i)
    cipher[i] = left[i];
for(int i=32; i<64; ++i)
    cipher[i] = right[i-32];
currentBits = cipher;
for(int i=0; i<64; ++i)
    cipher[63-i] = currentBits[64-IP_1[i]];
```

```
mov     ebx, ecx
xor     ecx, [rdx+r8*8+4]
rol     ebx, 1Ch
xor     ebx, [rdx+r8*8]
mov     eax, ecx
movzx   r15d, cl
shr     eax, 18h
cdqe
mov     eax, [r14+rax*4]
xor     eax, [r13+r15*4+0]
mov     r15d, ebx
shr     r15d, 18h
xor     eax, [rdi]
movsxd  r15, r15d
xor     eax, [r12+r15*4]
mov     r15d, ecx
movzx   ecx, ch
shr     r15d, 10h
movzx   r15d, r15b
xor     eax, [rbp+r15*4+0]
xor     eax, [r11+rcx*4]
movzx   ecx, bl
xor     eax, [r10+rcx*4]
mov     ecx, ebx
movzx   ebx, bh
shr     ecx, 10h
movzx   ecx, cl
xor     eax, [r9+rcx*4]
lea     rcx, _ZN5Botan10DES_SPBOX5E
xor     eax, [rcx+rbx*4]
mov     [rdi], eax
mov     ebx, eax
xor     eax, [rdx+r8*8+0Ch]
rol     ebx, 1Ch
xor     ebx, [rdx+r8*8+8]
```

加解密函数定位 流量分析

- 流量分析法
  - 流量分析是一种通过分析网络报文中各个字节的变化规律提取报文对应的协议格式信息的网络协议分析方法
  - 通过分析数据的“混乱”程度来确定是否被加密

明文

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

密文

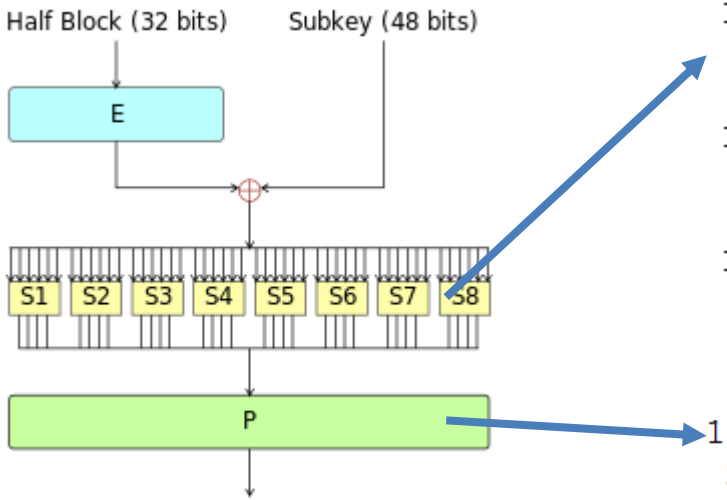
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000h:	72	A4	66	FD	53	67	CC	A4	93	04	55	A7	1C	E4	8B	23

- 劣势
  - 仅判断了是否存在加解密过程

加解密函数定位 静态分析

- 静态分析法
  - 查找加解密运算过程中使用的某些特征来确定一个二进制程序中是否包含某种加解密算法

DES加密



S1盒压缩置换表

14,	4,	13,	1,	2,	15,	11,	8,	3,	10,	6,	12,	5,	9,	0,	7
0,	15,	7,	4,	14,	2,	13,	1,	10,	6,	12,	11,	9,	5,	3,	8
4,	1,	14,	8,	13,	6,	2,	11,	15,	12,	9,	7,	3,	10,	5,	0
15,	12,	8,	2,	4,	9,	1,	7,	5,	11,	3,	14,	10,	0,	6,	13

S2盒压缩置换表

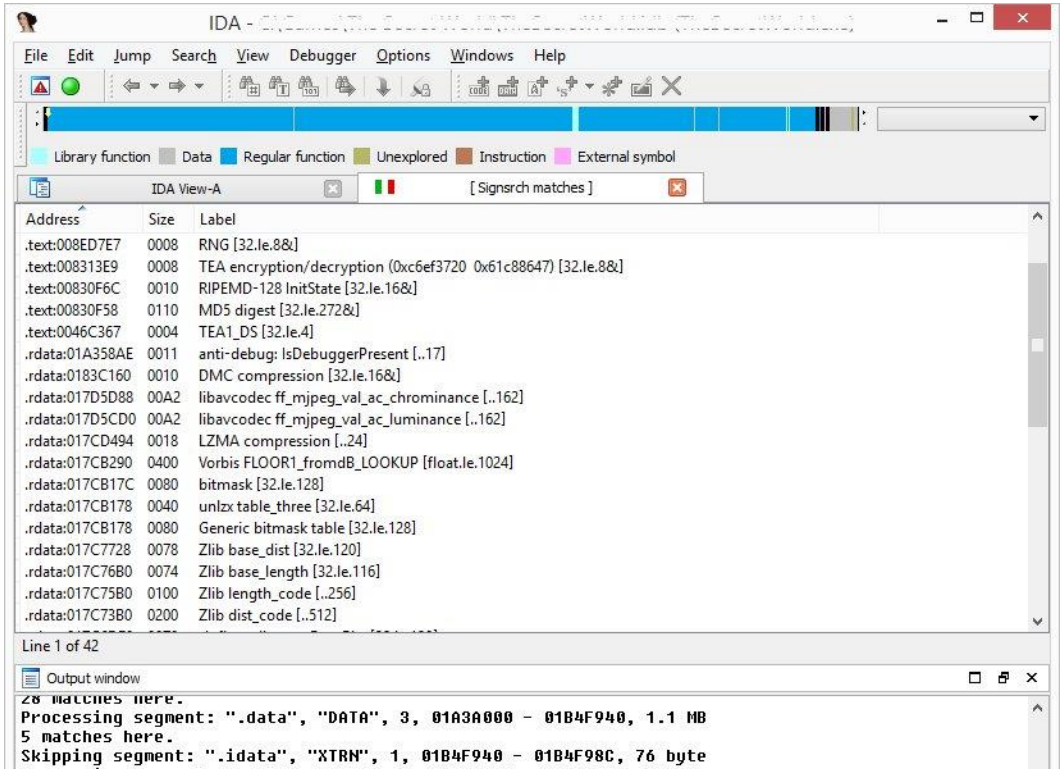
15,	1,	8,	14,	6,	11,	3,	4,	9,	7,	2,	13,	12,	0,	5,	10
3,	13,	4,	7,	15,	2,	8,	14,	12,	0,	1,	10,	6,	9,	11,	5
0,	14,	7,	11,	10,	4,	13,	1,	5,	8,	12,	6,	9,	3,	2,	15
13,	8,	10,	1,	3,	15,	4,	2,	11,	6,	7,	12,	0,	5,	14,	9

P盒置换表

16,	7,	20,	21,	29,	12,	28,	17,	1,	15,	23,	26,	5,	18,	31,	10
2,	8,	24,	14,	32,	27,	3,	9,	19,	13,	30,	6,	22,	11,	4,	25

加解密函数定位 静态分析

- IDA中的FindCrypt、IDA Signsrch插件
  - 查找加解密算法的魔术常数
- OllyDbg中的SnD Reverser Tool插件
  - 查找加解密算法具有固定特征的双字节或字节流



加解密函数定位 静态分析

- 考勤系统Attendance.exe的分析结果

IDA View-A | [Signsrch matches] | Hex View-1 | Stru

Address	Size	Label
.data:0000000000467B08	0040	DES initial permutation IP [..64]
.data:0000000000467B88	0038	DES permuted choice table (key) [..56]
.data:0000000000467BC0	0030	DES permuted choice key (table) [..48]
.data:0000000000467BF0	0030	DES_ei [..48]
.data:0000000000467C20	0020	DES_p32i [..32]
.data:0000000000467C60	0200	DES S-boxes [..512]

```
.data:00467C20 byte_467C20      db 10h
.data:00467C20
.data:00467C20
.data:00467C21                  db    7
.data:00467C22                  db  14h
.data:00467C23                  db  15h
.data:00467C24                  db  1Dh
.data:00467C25                  db  0Ch
.data:00467C26                  db  1Ch
.data:00467C27                  db  11h
.data:00467C28                  db    1
.data:00467C29                  db  0Fh
.data:00467C2A                  db  17h
.data:00467C2B                  db  1Ah
.data:00467C2C                  db    5
.data:00467C2D                  db  12h
.data:00467C2E                  db  1Fh
.data:00467C2F                  db  0Ah
.data:00467C30                  db    2
.data:00467C31                  db    8
.data:00467C32                  db  18h
.data:00467C33                  db  0Eh
.data:00467C34                  db  20h
.data:00467C35                  db  1Bh
.data:00467C36                  db    3
.data:00467C37                  db    9
.data:00467C38                  db  13h
.data:00467C39                  db  0Dh
```

```
loc_42AA05:
movzx   dx, byte ptr [eax]
movzx   bx, byte ptr [eax+5]
add     eax, 6
add     esi, 40h
lea     edx, [ebx+edx*2]
add     ecx, 4
movzx   bx, byte ptr [eax-5]
lea     edx, [ebx+edx*2]
movzx   bx, byte ptr [eax-4]
lea     edx, [ebx+edx*2]
movzx   bx, byte ptr [eax-3]
lea     edx, [ebx+edx*2]
movzx   bx, byte ptr [eax-2]
lea     edx, [ebx+edx*2]
mov     [esp+0Ch+arg_0], edx
movsx   edx, dx
movzx   dx, byte_467C20[edx+esi] ; <$ignsrch> "DES_p32i [..32]"
mov     [esp+0Ch+arg_0], edx
and     dl, 1
sar     word ptr [esp+0Ch+arg_0], 1
mov     [ecx-3], dl
mov     dl, byte ptr [esp+0Ch+arg_0]
sar     word ptr [esp+0Ch+arg_0], 1
and     dl, 1
mov     [ecx-4], dl
```

加解密函数定位 静态分析

- 静态分析法的劣势
  - 不能识别私有加解密函数或一些被修改过的加解密函数
  - 可能存在误判

sbox_entry()

```
uint8_t Botan::GOST_28147_89_Params::sbox_entry ( size_t row,
                                                  size_t col
                                                ) const
```

sbox_pair()

```
uint8_t Botan::GOST_28147_89_Params::sbox_pair ( size_t row,
                                                 size_t col
                                               ) const
```

使用函数实现S盒

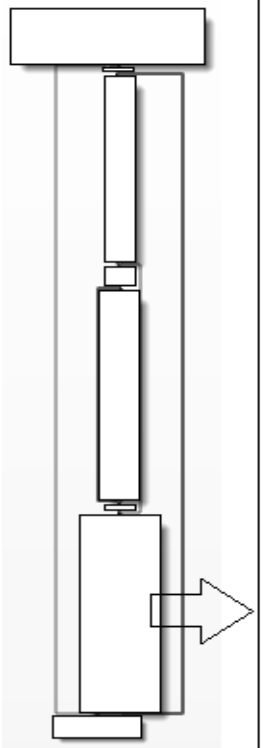
```
//inversepermutationbox
static BYTE inversepermutationbox[]=
{ /*  0    1    2    3    4    5    6    7    8    9    a
    0x52,0x09,0x6a,0xd5,0x30,0x36,0xa5,0x38,0xbf,0x40,0xa3,
    0x7c,0xe3,0x39,0x82,0x9b,0x2f,0xff,0x87,0x34,0x8e,0x43,
    0x54,0x7b,0x94,0x32,0xa6,0xc2,0x23,0x3d,0xee,0x4c,0x95,
    0x08,0x2e,0xa1,0x66,0x28,0xd9,0x24,0xb2,0x76,0x5b,0xa2,
    0x72,0xf8,0xf6,0x64,0x86,0x68,0x98,0x16,0xd4,0xa4,0x5c,
    0x6c,0x70,0x48,0x50,0xfd,0xed,0xb9,0xda,0x5e,0x15,0x46,
    0x90,0xd8,0xab,0x00,0x8c,0xbc,0xd3,0x0a,0xf7,0xe4,0x58,
    0xd0,0x2c,0x1e,0x8f,0xca,0x3f,0x0f,0x02,0xc1,0xaf,0xbd,
    0x3a,0x91,0x11,0x41,0x4f,0x67,0xdc,0xea,0x97,0xf2,0xcf,
    0x96,0xac,0x74,0x22,0xe7,0xad,0x35,0x85,0xe2,0xf9,0x37,
    0x47,0xf1,0x1a,0x71,0x1d,0x29,0xc5,0x89,0x6f,0xb7,0x62,
    0xfc,0x56,0x3e,0x4b,0xc6,0xd2,0x79,0x20,0x9a,0xdb,0xc0,
    0x1f,0xdd,0xa8,0x33,0x88,0x07,0xc7,0x31,0xb1,0x12,0x10,
    0x60,0x51,0x7f,0xa9,0x19,0xb5,0x4a,0x0d,0x2d,0xe5,0x7a,
    0xa0,0xe0,0x3b,0x4d,0xae,0x2a,0xf5,0xb0,0xc8,0xeb,0xbb,
    0x17,0x2b,0x04,0x7e,0xba,0x77,0xd6,0x26,0xe1,0x69,0x14,
};

int main( int argc, char* argv[] ) {
    ...
    for (int i = 0; i < sizeof(inversepermutationbox); i++)
    {
        printf("%d, %x\n", i, inversepermutationbox[i]);
    }
    ...
    return 0;
}
```

巧合的包含了AES的特征数据

加解密函数定位 动态分析

- 动态分析法
  - 通过分析程序的执行过程来判断其中是否包含加解密函数

AES加密函数



```
.text:04E0EA5   mov     eax, [rsp+30h+var_64]
.text:04E0EA9   mov     r15, [rsp+30h+var_48]
.text:04E0EAE   mov     esi, ecx
.text:04E0EB0   mov     rdi, [r15]
.text:04E0EB3   lea     r14, _ZN5Botan12_GLOBAL__N_1L2SEE
.text:04E0EBA   shr     esi, 18h
.text:04E0EBD   lea     r10, _ZN5Botan12_GLOBAL__N_1L2SEE
.text:04E0EC4   movsxd  rsi, esi
.text:04E0EC7   movzx   esi, byte ptr [r14+rsi]
.text:04E0ECC   mov     r14, [rsp+30h+var_50]
.text:04E0ED1   xor     sil, [rdi]
.text:04E0ED4   mov     [r14], sil
.text:04E0ED7   mov     esi, edx
.text:04E0ED9   mov     rdi, [r15]
.text:04E0EDC   shr     esi, 10h
.text:04E0EDF   movzx   esi, sil
.text:04E0EE3   movzx   esi, byte ptr [r10+rsi]
                ......
.text:04E0FEF   xor     sil, [rdi+0Bh]
.text:04E0FF3   mov     [r14+0Bh], sil
.text:04E0FF7   mov     rdi, [r15]
.text:04E0FFA   movzx   esi, byte ptr [r10+rbx]
.text:04E0FFF   xor     sil, [rdi+0Ch]
.text:04E1003   mov     [r14+0Ch], sil
.text:04E1007   mov     rsi, [r15]
.text:04E100A   xor     cl, [rsi+0Dh]
.text:04E100D   mov     [r14+0Dh], cl
.text:04E1011   mov     rcx, [r15]
.text:04E1014   xor     dl, [rcx+0Eh]
.text:04E1017   mov     [r14+0Eh], dl
.text:04E101B   mov     rdx, [r15]
.text:04E101E   xor     al, [rdx+0Fh]
.text:04E1021   mov     [r14+0Fh], al
.text:04E1025   mov     rax, [rsp+30h+var_40]
.text:04E102A   cmp     [rsp+30h+var_60], rax
.text:04E102F   jnz     loc_4E0BA0
```

共103条指令
计算类指令3条
逻辑类指令16条
移位类指令8条

加解密函数定位 动态分析

- 加解密函数的特征
  - 进行大量的计算以完成加解密工作
  - 一般仅含有非常少的跳转指令
  - 函数中的基本块包含的指令数通常明显多于其他函数
    - 一般情况下，加解密函数中的基本块会包含20条以上的指令
  - 加解密函数中计算工作会大量使用运算类指令，是一种密集计算函数
    - 加解密函数的基本块中运算类指令的比例明显高于其他基本块
    - 加解密函数中运算类指令的占比在25%~80%

加解密函数定位 动态分析

- 定位方案
  - 统计程序中各个部分的指令特征，将包含大量的算术运算和位运算的密集计算部分视为执行加解密操作的部分

种类	指令
计算类	add, sub, inc, dec, adc, sbb, mul, div, imul, idiv
逻辑类	and, or, not, xor, neg
移位类	shl, shr, shld, shrd, rcl, rol, rcr, ror, sal, sar

加解密函数定位 动态分析

- Mov类指令 17条
- 运算类指令 11条
- 总指令数 34

- 运算类指令占比32.35%

```
loc_42AA05:
movzx   dx, byte ptr [eax]
movzx   bx, byte ptr [eax+5]
add     eax, 6
add     esi, 40h
lea     edx, [ebx+edx*2]
add     ecx, 4
movzx   bx, byte ptr [eax-5]
lea     edx, [ebx+edx*2]
movzx   bx, byte ptr [eax-4]
lea     edx, [ebx+edx*2]
movzx   bx, byte ptr [eax-3]
lea     edx, [ebx+edx*2]
movzx   bx, byte ptr [eax-2]
lea     edx, [ebx+edx*2]
mov     [esp+0Ch+arg_0], edx
movsx   edx, dx
movzx   dx, byte_467C20[edx+esi] ; <$ignsrch> "DES_p32i [..32]"
mov     [esp+0Ch+arg_0], edx
and     dl, 1
sar     word ptr [esp+0Ch+arg_0], 1
mov     [ecx-3], dl
mov     dl, byte ptr [esp+0Ch+arg_0]
sar     word ptr [esp+0Ch+arg_0], 1
and     dl, 1
mov     [ecx-4], dl
mov     dl, byte ptr [esp+0Ch+arg_0]
mov     bl, dl
sar     dl, 1
and     bl, 1
and     dl, 1
mov     [ecx-5], bl
mov     [ecx-6], dl
dec     edi
jnz     short loc_42AA05
```

考勤系统Attendance.exe

加解密函数定位 动态分析

- 动态分析法的劣势
  - 无法区分加解密函数和其他密集计算函数

- 一个非加解密型的密集计算函数
  - 共有22条指令
  - 运算类指令17条
  - 运算类指令占比77.27%

```
 1  push    ebp
 2  mov     ebp, esp
 3  xor     eax eax
 4  add     eax, 1h
 5  add     eax, 2h
 6  add     eax, 3h
 7  add     eax, 4h
 8  add     eax, 5h
 9  add     eax, 6h
10  add     eax, 7h
11  add     eax, 8h
12  add     eax, 9h
13  add     eax, 10h
14  add     eax, 11h
15  add     eax, 12h
16  add     eax, 13h
17  add     eax, 14h
18  add     eax, 15h
19  add     eax, 16h
20  mov     esp, ebp
21  pop     ebp
22  retn
```

优劣分析

优劣分析

- 适用于模糊测试的加解密函数定位
  - 人工定位
  - 静态分析
  - 动态分析

	优势	劣势
人工定位	适宜各类场景	人力成本、时间成本高
静态分析	能够获取算法名称	不能识别私有加解密函数
动态分析	不依赖加密算法的特征数据块	无法区分加解密函数和其他密集计算函数

应用总结

应用总结

- IDA Signsrch对基于Botan算法库构建的加密程序的分析结果

IDA Signsrch的特征库中没有IDEA的特征

GOST、Serpent加密算法没有使用固定的数组储存IDA Signsrch记录特征数据

程序	发现加密算法特征数据块
AES	√
Blowfish	√
Camellia	√
CAST	√
DES	√
3DES	√
GOST	×
IDEA	×
KASUMI	√
MISTY1	√
Noekeon	√
RC6	√
SEED	√
Serpent	×
Skipjack	√
Twofish	√

应用总结

- 程序中包含输入数据的解密时的漏洞挖掘结果

发现漏洞所需的测试数据的数量

工具	最小值	最大值	平均值
zzuf	2793	152407	64940
AFL	41684	212105	84460

```
fp = fopen(file_name, "rb");
n = fread(buff, 16, 1, fp);
Decrypt(buff, 16);
if ((buff[1] + buff[2]) > buff[0] &&
    (buff[2] + buff[0]) > buff[1] &&
    (buff[0] + buff[1]) > buff[2] ) {
    if (buff[1] == buff[0] || buff[2] == buff[1] || buff[1] == buff[2]) {
        program crash;
    }
}
```

应用总结

- 去除加解密函数的影响后的漏洞挖掘结果

发现漏洞所需的测试数据的数量

工具	最小值	最大值	平均值
zzuf	39	12949	3363
AFL	426	657	491

```
if ((buff[1] + buff[2]) > buff[0] &&
    (buff[2] + buff[0]) > buff[1] &&
    (buff[0] + buff[1]) > buff[2] ) {
    if (buff[1] == buff[0] || buff[2] == buff[1] || buff[1] == buff[2]) {
        program crash;
    }
}
```

参考文献

[1] Wondracek G, Comparetti P M, Kruegel C, et al. Automatic Network Protocol Analysis [C]. 16th Annual Network & Distributed System Security Symposium (NDSS 2008), San Diego, CA, 2008: 1-14.
[2] Wang Z, Jiang X, Cui W, et al. ReFormat: Automatic Reverse Engineering of Encrypted Messages [C]// 14th European Symposium on Research in Computer Security. Saint-Malo, France:[s.n.], 2009: 200-215.
[3] Li X, Wang X, Chang W. CipherXRay: Exposing Cryptographic Operations and Transient Secrets from Monitored Binary Execution [J]. IEEE Transactions on Dependable & Secure Computing, 2014, 11(2): 101-114.

道德经

道可道，非常道。名可名，非常名。无名天地之始。有名万物之母。故常无欲以观其妙。常有欲以观其徼。此两者同出而异名，同谓之玄。玄之又玄，众妙之门。

谢谢！