文件级恶意代码扫描引擎中的加壳识别技术

安天实验室 Swordlea

2003 年 12 月 25 日

在文件级恶意代码扫描引擎的设计与实现过程中，加壳后样本的特征选取和识别是较为棘手的问题。恶意代码的制作者为了使其作品传播更广，往往使用软件加壳的方式。样本被加壳以后，其原有特征码被变形或彻底隐藏，在一定程度中增加了样本分析与识别的难度。

目前反病毒公司在对加壳后样本的处理上一般采用以下几种方式：

1. 不加处理，捕获流行样本，直接在加壳后的样本上选取特征码，例如：Panda、NAV。该方法比较简单，只要样本分析人员不将特征码选取在壳程序自身即可，能够在发现流行样本的第一时间作出快速响应，可以作为大规模疫情的应急处理方案。其缺点是大大增加了样本特征库的记录数量，尤其是在流行样本为多个加壳工具加壳或某个加壳工具多次加壳的情况下。

2. 查毒程序预置流行加壳工具的对应脱壳模块，例如 AVP。在 12 月 24 日升级后，通过相关工具的分析，可以发现其 unpack.avc 模块中，集成了包括 TeLock、UPX 、ASPack 在内的 1210 个脱壳模块。该方式要求反病毒公司对流行软件加壳工具及时了解、深入分析，或者与相关软件的作者有较为密切的联系。

3. 在虚拟机环境下，利用被加壳样本程序自身的脱壳代码进行脱壳，速度较慢。例如：瑞星 2003 提出的“智能解包还原技术”（对 UPX、ASPack 等支持很好，但对 telock、asprotect 等效果有限）。

4. 在真实环境下，利用被加壳样本程序自身的脱壳代码进行脱壳。速度非常快，但需要对加壳样本的脱壳代码进行精确校验，否则可能被样本程序伪装欺骗。可是，精确校验又会丧失对变形加壳代码的识别能力。

下面以 UPX1.24 for win 加壳为例，说明各处理方式的实现思路。

首先，使用 VC 6.0 的 AppWizard 生成 Dialog Based 应用程序 test，编译 Release 版后，得到 20,480 字节的 test.exe。其 MZ/PE 头信息如下：

```
00000000  4D 5A 90 00 03 00 00 00-04 00 00 00 FF FF 00 00   MZ..............
00000010  B8 00 00 00 00 00 00 00-40 00 00 00 00 00 00 00   ........@.......
00000020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
00000030  00 00 00 00 00 00 00 00-00 00 00 00 D8 00 00 00   ................
00000040  0E 1F BA 0E 00 B4 09 CD-21 B8 01 4C CD 21 54 68   ........!..L.!Th
00000050   69 73 20 70 72 6F 67 72-61 6D 20 63 61 6E 6E 6F   is program
canno
00000060  74 20 62 65 20 72 75 6E-20 69 6E 20 44 4F 53 20   t be run in
```

```
DOS
00000070  6D 6F 64 65 2E 0D 0D 0A-24 00 00 00 00 00 00 00   mode....$.......
00000080  FB C6 9A 8E BF A7 F4 DD-BF A7 F4 DD BF A7 F4 DD   ................
00000090  DD B8 E7 DD BB A7 F4 DD-3C BB FA DD BE A7 F4 DD   ........<.......
000000A0  57 B8 FE DD B4 A7 F4 DD-57 B8 F0 DD BA A7 F4 DD   W.......W.......
000000B0  BF A7 F5 DD CC A7 F4 DD-57 B8 FF DD B9 A7 F4 DD   ........W.......
000000C0  07 A1 F2 DD BE A7 F4 DD-52 69 63 68 BF A7 F4 DD   ........Rich....
000000D0  00 00 00 00 00 00 00 00-50 45 00 00 4C 01 04 00   ........PE..L...
000000E0  6B 49 EA 3F 00 00 00 00-00 00 00 00 E0 00 0F 01   kI.?............
000000F0  0B 01 06 00 00 10 00 00-00 30 00 00 00 00 00 00   .........0......
00000100  E0 16 00 00 00 10 00 00-00 20 00 00 00 00 40 00   ......... ....@.
00000110  00 10 00 00 00 10 00 00-04 00 00 00 00 00 00 00   ................
00000120  04 00 00 00 00 00 00 00-00 50 00 00 00 10 00 00   .........P......
00000130  00 00 00 00 02 00 00 00-00 00 10 00 00 10 00 00   ................
00000140  00 00 10 00 00 10 00 00-00 00 00 00 10 00 00 00   ................
00000150  00 00 00 00 00 00 00 00-A8 25 00 00 64 00 00 00   .........%..d...
00000160  00 40 00 00 A0 0A 00 00-00 00 00 00 00 00 00 00   .@..............
00000170  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
00000180  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
00000190  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
000001A0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
000001B0  00 20 00 00 DC 01 00 00-00 00 00 00 00 00 00 00   . ..............
000001C0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
000001D0  2E 74 65 78 74 00 00 00-42 09 00 00 00 10 00 00   .text...B.......
000001E0  00 10 00 00 00 10 00 00-00 00 00 00 00 00 00 00   ................
000001F0  00 00 00 00 20 00 00 60-2E 72 64 61 74 61 00 00   .... ..`.rdata..
00000200  B6 09 00 00 00 20 00 00-00 10 00 00 00 20 00 00   ..... ....... ..
```

对该程序开始部分进行反汇编，得到反汇编代码如下。同时可以看到入口点为 16E0。

```
004016E0 >PUSH EBP
004016E1  MOV EBP,ESP
004016E3  PUSH -1
004016E5  PUSH test.004024F8
004016EA    PUSH  <JMP.&MSVCRT._except_handler3>          ;    SE  handler
installation
004016EF  MOV EAX,DWORD PTR FS:[0]
004016F5  PUSH EAX
```

```
004016F6  MOV DWORD PTR FS:[0],ESP
004016FD  SUB ESP,68
00401700  PUSH EBX
00401701  PUSH ESI
00401702  PUSH EDI
00401703  MOV DWORD PTR SS:[EBP-18],ESP
00401706  XOR EBX,EBX
00401708  MOV DWORD PTR SS:[EBP-4],EBX
0040170B  PUSH 2
0040170D          CALL      DWORD      PTR      DS:[<&MSVCRT.__set_app_ty>;
MSVCRT.__set_app_type
00401713  POP ECX
 ……
```

使用 UPX1.24 对其加壳，加壳后，前 512 字节内容如下：

```
00000000  4D 5A 90 00 03 00 00 00-04 00 00 00 FF FF 00 00   MZ..............
00000010  B8 00 00 00 00 00 00 00-40 00 00 00 00 00 00 00   ........@.......
00000020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
00000030  00 00 00 00 00 00 00 00-00 00 00 00 D8 00 00 00   ................
00000040  0E 1F BA 0E 00 B4 09 CD-21 B8 01 4C CD 21 54 68   ........!..L.!Th
00000050   69 73 20 70 72 6F 67 72-61 6D 20 63 61 6E 6E 6F   is program
canno
00000060   74 20 62 65 20 72 75 6E-20 69 6E 20 44 4F 53 20   t be run in
DOS
00000070  6D 6F 64 65 2E 0D 0D 0A-24 00 00 00 00 00 00 00   mode....$.......
00000080  FB C6 9A 8E BF A7 F4 DD-BF A7 F4 DD BF A7 F4 DD   ................
00000090  DD B8 E7 DD BB A7 F4 DD-3C BB FA DD BE A7 F4 DD   ........<.......
000000A0  57 B8 FE DD B4 A7 F4 DD-57 B8 F0 DD BA A7 F4 DD   W.......W.......
000000B0  BF A7 F5 DD CC A7 F4 DD-57 B8 FF DD B9 A7 F4 DD   ........W.......
000000C0  07 A1 F2 DD BE A7 F4 DD-52 69 63 68 BF A7 F4 DD   ........Rich....
000000D0  00 00 00 00 00 00 00 00-50 45 00 00 4C 01 03 00   ........PE..L...
000000E0  6B 49 EA 3F 00 00 00 00-00 00 00 00 E0 00 0F 01   kI.?............
000000F0  0B 01 06 00 00 10 00 00-00 10 00 00 00 50 00 00   .............P..
00000100  B0 69 00 00 00 60 00 00-00 70 00 00 00 00 40 00   .i...`...p....@.
00000110  00 10 00 00 00 02 00 00-04 00 00 00 00 00 00 00   ................
00000120  04 00 00 00 00 00 00 00-00 80 00 00 00 10 00 00   ................
00000130  00 00 00 00 02 00 00 00-00 00 10 00 00 10 00 00   ................
```

```
00000140  00 00 10 00 00 10 00 00-00 00 00 00 10 00 00 00   ................
00000150  00 00 00 00 00 00 00 00-DC 78 00 00 F8 00 00 00   ........x......
00000160  00 70 00 00 DC 08 00 00-00 00 00 00 00 00 00 00   .p..............
00000170  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
00000180  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
00000190  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
000001A0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
000001B0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
000001C0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
000001D0  55 50 58 30 00 00 00 00-00 50 00 00 00 10 00 00   UPX0.....P......
000001E0  00 00 00 00 00 04 00 00-00 00 00 00 00 00 00 00   ................
000001F0  00 00 00 00 80 00 00 E0-55 50 58 31 00 00 00 00   ........UPX1....
00000200  00 10 00 00 00 60 00 00-00 0C 00 00 00 04 00 00   .....`..........
```

加壳后，其入口部分反汇编代码如下，入口点为 69B0。

```
004069B0 > PUSHAD
004069B1   MOV ESI,test.00406000
004069B6   LEA EDI,DWORD PTR DS:[ESI+FFFFB000]
004069BC   PUSH EDI
004069BD   OR EBP,FFFFFFFF
004069C0   JMP SHORT test.004069D2  ; EB 10
004069C2   NOP  ; 90
004069C3   NOP  ; 90
004069C4   NOP  ; 以上三条指令的机器码可以作为快速判定加壳器依据
004069C5   NOP
004069C6   NOP
004069C7   NOP
004069C8   MOV AL,BYTE PTR DS:[ESI] ; 取密文，始于文件偏移 0x400 处
004069CA   INC ESI
004069CB   MOV BYTE PTR DS:[EDI],AL ; 解压缓冲区
004069CD   INC EDI
004069CE   ADD EBX,EBX              ; EBX 为当前实例句柄
004069D0   JNZ SHORT test.004069D9
004069D2   MOV EBX,DWORD PTR DS:[ESI]
004069D4   SUB ESI,-4
004069D7   ADC EBX,EBX
004069D9   JB SHORT test.004069C8
```

```
004069DB   MOV EAX,1
004069E0   ADD EBX,EBX
004069E2   JNZ SHORT test.004069EB
004069E4   MOV EBX,DWORD PTR DS:[ESI]
004069E6   SUB ESI,-4
004069E9   ADC EBX,EBX
004069EB   ADC EAX,EAX
004069ED   ADD EBX,EBX
004069EF   JNB SHORT test.004069E0
004069F1   JNZ SHORT test.004069FC
004069F3   MOV EBX,DWORD PTR DS:[ESI]
004069F5   SUB ESI,-4
004069F8   ADC EBX,EBX
004069FA   JNB SHORT test.004069E0
004069FC   XOR ECX,ECX
004069FE   SUB EAX,3
00406A01   JB SHORT test.00406A10
00406A03   SHL EAX,8
00406A06   MOV AL,BYTE PTR DS:[ESI]
00406A08   INC ESI
00406A09   XOR EAX,FFFFFFFF
00406A0C   JE SHORT test.00406A82
00406A0E   MOV EBP,EAX
00406A10   ADD EBX,EBX
00406A12   JNZ SHORT test.00406A1B
00406A14   MOV EBX,DWORD PTR DS:[ESI]
00406A16   SUB ESI,-4
00406A19   ADC EBX,EBX
00406A1B   ADC ECX,ECX
00406A1D   ADD EBX,EBX
00406A1F   JNZ SHORT test.00406A28
00406A21   MOV EBX,DWORD PTR DS:[ESI]
00406A23   SUB ESI,-4
00406A26   ADC EBX,EBX
00406A28   ADC ECX,ECX
00406A2A   JNZ SHORT test.00406A4C
```

```
00406A2C   INC ECX
00406A2D   ADD EBX,EBX
00406A2F   JNZ SHORT test.00406A38
00406A31   MOV EBX,DWORD PTR DS:[ESI]
00406A33   SUB ESI,-4
00406A36   ADC EBX,EBX
00406A38   ADC ECX,ECX
00406A3A   ADD EBX,EBX
00406A3C   JNB SHORT test.00406A2D
00406A3E   JNZ SHORT test.00406A49
00406A40   MOV EBX,DWORD PTR DS:[ESI]
00406A42   SUB ESI,-4
00406A45   ADC EBX,EBX
00406A47   JNB SHORT test.00406A2D
00406A49   ADD ECX,2
00406A4C   CMP EBP,-0D00
00406A52   ADC ECX,1
00406A55   LEA EDX,DWORD PTR DS:[EDI+EBP]
00406A58   CMP EBP,-4
00406A5B   JBE SHORT test.00406A6C
00406A5D   MOV AL,BYTE PTR DS:[EDX]
00406A5F   INC EDX
00406A60   MOV BYTE PTR DS:[EDI],AL
00406A62   INC EDI
00406A63   DEC ECX
00406A64   JNZ SHORT test.00406A5D
00406A66   JMP test.004069CE
00406A6B   NOP
00406A6C   MOV EAX,DWORD PTR DS:[EDX]   ; 从缓冲区复制
00406A6E   ADD EDX,4
00406A71   MOV DWORD PTR DS:[EDI],EAX   ; 写入应用程序空间
00406A73   ADD EDI,4
00406A76   SUB ECX,4
00406A79   JA SHORT test.00406A6C
00406A7B   ADD EDI,ECX
00406A7D   JMP test.004069CE
```

```
00406A82   POP ESI
00406A83   MOV EDI,ESI
00406A85   MOV ECX,36
00406A8A   MOV AL,BYTE PTR DS:[EDI]
00406A8C   INC EDI
00406A8D   SUB AL,0E8
00406A8F   CMP AL,1
00406A91   JA SHORT test.00406A8A
00406A93   CMP BYTE PTR DS:[EDI],1
00406A96   JNZ SHORT test.00406A8A
00406A98   MOV EAX,DWORD PTR DS:[EDI]
00406A9A   MOV BL,BYTE PTR DS:[EDI+4]
00406A9D   SHR AX,8
00406AA1   ROL EAX,10
00406AA4   XCHG AH,AL
00406AA6   SUB EAX,EDI
00406AA8   SUB BL,0E8
00406AAB   ADD EAX,ESI
00406AAD   MOV DWORD PTR DS:[EDI],EAX   ; 写入应用程序空间
00406AAF   ADD EDI,5
00406AB2   MOV EAX,EBX
00406AB4   LOOPD SHORT test.00406A8F
00406AB6   LEA EDI,DWORD PTR DS:[ESI+4000]
00406ABC   MOV EAX,DWORD PTR DS:[EDI]
00406ABE   OR EAX,EAX
00406AC0   JE SHORT test.00406B07
00406AC2   MOV EBX,DWORD PTR DS:[EDI+4]
00406AC5   LEA EAX,DWORD PTR DS:[EAX+ESI+68DC] ; 加载应用程序引入的 DLL
00406ACC   ADD EBX,ESI
00406ACE   PUSH EAX
00406ACF   ADD EDI,8
00406AD2   CALL DWORD PTR DS:[ESI+6940]       ; KERNEL32.LoadLibraryA
00406AD8   XCHG EAX,EBP
00406AD9   MOV AL,BYTE PTR DS:[EDI]
00406ADB   INC EDI
00406ADC   OR AL,AL
```

```
00406ADE   JE SHORT test.00406ABC
00406AE0   MOV ECX,EDI
00406AE2   JNS SHORT test.00406AEB
00406AE4   MOVZX EAX,WORD PTR DS:[EDI]
00406AE7   INC EDI
00406AE8   PUSH EAX
00406AE9   INC EDI
00406AEA   DB B9
00406AEB   PUSH EDI
00406AEC   DEC EAX
00406AED   REPNE SCAS BYTE PTR ES:[EDI] ; 导入表中各函数
00406AEF   PUSH EBP
00406AF0   CALL DWORD PTR DS:[ESI+6944] ; KERNEL32.GetProcAddress
00406AF6   OR EAX,EAX
00406AF8   JE SHORT test.00406B01
00406AFA   MOV DWORD PTR DS:[EBX],EAX
00406AFC   ADD EBX,4
00406AFF   JMP SHORT test.00406AD9
00406B01   CALL DWORD PTR DS:[ESI+6948] ; 失败，调用 KERNEL32.ExitProcess
00406B07   POPAD
00406B08   JMP test.004016E0    ; 跳到原入口点
```

当跟踪执行到 00406B08 处时，可以在 004016E0 处找到如下代码：

```
004016E0 >PUSH EBP
004016E1  MOV EBP,ESP
004016E3  PUSH -1
004016E5  PUSH test.004024F8
004016EA   PUSH  <JMP.&MSVCRT._except_handler3>          ;   SE  handler
installation
……
```

即是加壳前 test 程序入口代码。

在对加壳器的识别上不应该寄希望于判断节名。目前有许多加壳器的修补工具可以修改节名及其它大部分特征。比较快速的方法是匹配代码段附近的若干字节，例如 UPX1.24 加壳后固定在代码段偏移 0x10 处有 DWORD 值 0x909010EB，就可以作为该版 UPX 的识别依据，而比较谨慎的方法是匹配脱壳代码的 MD5 或 CRC32 校验值。

参照前面说过的几种处理方式，对于该版本 UPX 加壳的可执行文件可以分别作如下处理：

若采用第一种方式，即直接在加壳后的样本上选取特征码，可以把特征码取在 0x400 处以后的若干字节；若采用第二种方式，即预置脱壳模块，则可以根据上面的反汇编代码很容易地写出针对该版本 UPX 的脱壳模块（由于只需要实现加壳代码的还原，00406AC2 以下的部分可以不必关心）；若采用第三种方式，即在虚拟机环境下利用被加壳样本程序自身的脱壳代码进行脱壳，应该首先快速地判断加壳工具（比如使用脱壳代码中的某些固定指令），然后执行该种脱壳代码所需的脱壳步数或有限的步数，再对脱壳缓冲区进行特征码匹配；若采用第四种方式，即在真实环境下利用被加壳样本程序自身的脱壳代码进行脱壳，应该首先准确地判断加壳工具（比如使用脱壳代码的 MD5 或 CRC32 校验值），然后执行该种脱壳代码所需的脱壳步数，再对脱壳缓冲区进行特征码匹配。

考虑到多次加壳的情况，扫描引擎的处理流程应如下设计：