StarForce 3 - Brief insight into a hidden world.

By [yAtEs]
[http://www.yates2k.net] [http://www.reteam.org]

These notes are intended for anyone wishing to study the working elements of this protection. It's very brief and unhelpful. :)

StarForce 3 executes most of its code in a virtual machine. The virtual machine has a set of instructions. Below you can find an example instruction. In order to find out what makes this protection strong you need to spend 30 mins tracing through several instructions and analyzing their operations. You will quickly find StarForce has what seems to be a binded flow of instructions. In other words you cannot inspect the opcode data, as it appears dynamic.

The fact is, there is structure but its so large its easy to be over come by the amount of work one would have to do, as you trace over each 'instruction' the opcode data will be extracted in different ways. You must learn each way, which is probably only feasible with good heuristics; however that's all boring ;-) its much better to study a few instructions to see how it all works.

n'joy

-yAtEs

StarForce Mov Instruction

```
.sforce:00BA28C8 sub_BA28C8      proc near
.sforce:00BA28C8                 mov     eax, [edi+8]
.sforce:00BA28CB                 add     eax, [edi+14h]  ; Add VM Base
to Opcode Ptr(EIP)
.sforce:00BA28CE                 mov     ecx, [eax]      ; Get Opcode
Data Dword 1 - 0x98004200
.sforce:00BA28D0                 mov     edx, [eax+4]    ; Get Opcode
Data Dword 2 - 0x08600135
.sforce:00BA28D3                 add     dword ptr [edi+14h], 8 ;
Increase EIP
.sforce:00BA28D7
.sforce:00BA28D7 ########## Extract Data for Metaphorsis
################
.sforce:00BA28D7 .
.sforce:00BA28D7                 push    ecx             ; Save Opcode
1
.sforce:00BA28D8 .
.sforce:00BA28D8                 mov     eax, edx        ; Get Opcode 2
.sforce:00BA28DA .
.sforce:00BA28DA                 shl     eax, 4
.sforce:00BA28DD                 shr     eax, 1Dh        ; extract bits
28,27,26 - EAX = 04
.sforce:00BA28E0 .
```

```
.sforce:00BA28E0                mov     ebx, ecx        ; Get Opcode 1
.sforce:00BA28E2 .
.sforce:00BA28E2                shl     ebx, 5
.sforce:00BA28E5                shr     ebx, 1Bh        ; extract bits
23,24,25,26,27 - EBX = 0
.sforce:00BA28E8 .
.sforce:00BA28E8                mov     ecx, edx        ; Get Opcode 2
.sforce:00BA28EA .
.sforce:00BA28EA                shl     ecx, 7
.sforce:00BA28ED                shr     ecx, 1Bh        ; extract bits
21,22,23,24,25 - ECX = 6
.sforce:00BA28F0
.sforce:00BA28F0
.sforce:00BA28F0 ############## START STREAM/METAPHORSIS ENCODING
##############
.sforce:00BA28F0
.sforce:00BA28F0                or      eax, eax
.sforce:00BA28F2                jz      short loc_BA2914
.sforce:00BA28F4                cmp     eax, 1
.sforce:00BA28F7                jz      short loc_BA291A ; EIP STREAM
.sforce:00BA28F9                cmp     eax, 2
.sforce:00BA28FC                jz      short loc_BA2920 ; EIP STREAM
.sforce:00BA28FE                cmp     eax, 3
.sforce:00BA2901                jz      short loc_BA2926 ; EIP STREAM
.sforce:00BA2903                cmp     eax, 4
.sforce:00BA2906                jz      short loc_BA292B ; EIP STREAM
.sforce:00BA2908                cmp     eax, 5
.sforce:00BA290B                jz      short loc_BA2930 ; DESTINATION
STREAM
.sforce:00BA290D                cmp     eax, 6
.sforce:00BA2910                jz      short loc_BA293A ; SOURCE
STREAM
.sforce:00BA2912                jmp     short loc_BA2944 ; Restore
Opcode 1
.sforce:00BA2914 ; ---------------------------------------------------
----------------------
.sforce:00BA2914
.sforce:00BA2914 loc_BA2914:                            ; CODE XREF:
sub_BA28C8+2A_j
.sforce:00BA2914                bts     [edi+24h], ecx
.sforce:00BA2918                jmp     short loc_BA2944 ; Restore
Opcode 1
.sforce:00BA291A ; ---------------------------------------------------
----------------------
.sforce:00BA291A
.sforce:00BA291A loc_BA291A:                            ; CODE XREF:
sub_BA28C8+2F_j
.sforce:00BA291A                btr     [edi+24h], ecx
.sforce:00BA291E                jmp     short loc_BA2944 ; Restore
Opcode 1
.sforce:00BA2920 ; ---------------------------------------------------
----------------------
.sforce:00BA2920
.sforce:00BA2920 loc_BA2920:                            ; CODE XREF:
sub_BA28C8+34_j
.sforce:00BA2920                btc     [edi+24h], ecx
.sforce:00BA2924                jmp     short loc_BA2944 ; Restore
```

```
Opcode 1
.sforce:00BA2926 ; ---------------------------------------------------
----------------------
.sforce:00BA2926
.sforce:00BA2926 loc_BA2926:                             ; CODE XREF:
sub_BA28C8+39_j
.sforce:00BA2926                rol     byte ptr [edi+24h], cl
.sforce:00BA2929                jmp     short loc_BA2944 ; Restore
Opcode 1
.sforce:00BA292B ; ---------------------------------------------------
----------------------
.sforce:00BA292B
.sforce:00BA292B loc_BA292B:                             ; CODE XREF:
sub_BA28C8+3E_j
.sforce:00BA292B                ror     byte ptr [edi+24h], cl
.sforce:00BA292E                jmp     short loc_BA2944 ; Restore
Opcode 1
.sforce:00BA2930 ; ---------------------------------------------------
----------------------
.sforce:00BA2930
.sforce:00BA2930 loc_BA2930:                             ; CODE XREF:
sub_BA28C8+43_j
.sforce:00BA2930                shl     ecx, 5
.sforce:00BA2933                or      ebx, ecx
.sforce:00BA2935                mov     [edi+0Ch], ebx
.sforce:00BA2938                jmp     short loc_BA2944 ; Restore
Opcode 1
.sforce:00BA293A ; ---------------------------------------------------
----------------------
.sforce:00BA293A
.sforce:00BA293A loc_BA293A:                             ; CODE XREF:
sub_BA28C8+48_j
.sforce:00BA293A                shl     ecx, 5
.sforce:00BA293D                or      ebx, ecx
.sforce:00BA293F                mov     [edi+10h], ebx
.sforce:00BA2942                jmp     short $+2
.sforce:00BA2944
#########################################################################
#########
.sforce:00BA2944
.sforce:00BA2944 loc_BA2944:                             ; CODE XREF:
sub_BA28C8+4A_j
.sforce:00BA2944                                         ;
sub_BA28C8+50_j ...
.sforce:00BA2944                pop     ecx              ; Restore
Opcode 1
.sforce:00BA2945                mov     ebx, [edi]       ;
VMRAM/REGISTERBASE Into EBX
.sforce:00BA2947 .
.sforce:00BA2947                mov     esi, edx         ; Get Opcode 2
.sforce:00BA2949
.sforce:00BA2949 ######### GET DESTINATION REGISTER ############
.sforce:00BA2949
.sforce:00BA2949                shl     esi, 14h         ; Decode
Source Register
.sforce:00BA294C                shr     esi, 18h         ; Bits 5->12
.sforce:00BA294F                add     esi, [edi+10h]   ; ESI = 0x13 +
```

```
[EDI+10h] = 0x23 SRC.REG+SRC_STREAM = REG 0x36
.sforce:00BA2952                and     esi, 0FFh       ; 1 Byte Reg
.sforce:00BA2952
.sforce:00BA2952 ################################################
.sforce:00BA2958 .
.sforce:00BA2958                mov     esi, [ebx+esi*4] ; REG * 4 =
REG_D8 + vmbase for location of data
.sforce:00BA295B .
.sforce:00BA295B ######## GET SOURCE REGISTER ##################
.sforce:00BA295B
.sforce:00BA295B                mov     eax, ecx        ; Get Opcode 1
.sforce:00BA295D .
.sforce:00BA295D                shl     eax, 0Fh        ; Decode
Destination Register
.sforce:00BA2960                shr     eax, 18h        ; Bits 10->18
.sforce:00BA2963                add     eax, [edi+0Ch]  ; EAX = 0x21 +
[EDI+0Ch] = 0x15 DEST.REG+DEST_STREAM = REG 0x36
.sforce:00BA2966                and     eax, 0FFh       ; 1 Byte Reg
.sforce:00BA296B
.sforce:00BA296B ################################################
.sforce:00BA296B
.sforce:00BA296B
.sforce:00BA296B
.sforce:00BA296B
.sforce:00BA296B .
.sforce:00BA296B                mov     [ebx+eax*4], esi ; !MOV
INSTRUCTION!  - COPY DATA INTO REG
.sforce:00BA296E
.sforce:00BA296E
.sforce:00BA296E
.sforce:00BA296E
.sforce:00BA296E
.sforce:00BA296E #### Obtain Instruction Index i.e find next
instruction ####
.sforce:00BA296E
.sforce:00BA296E                mov     esi, ecx        ; Get Opcode 1
.sforce:00BA2970 .
.sforce:00BA2970                shr     esi, 1Bh        ; ESI = 13
.sforce:00BA2973                mov     ebp, edx        ; Get Opcode 2
.sforce:00BA2975                shl     ebp, 1Ch
.sforce:00BA2978                shr     ebp, 17h        ; EBP=0xA0
.sforce:00BA297B                or      esi, ebp        ; ESI = 0xB3,
INDEX = 0xB3 !
.sforce:00BA297D
.sforce:00BA297D
############################################################
.sforce:00BA297D
.sforce:00BA297D ########## Get Index Decrypt Key ###################
.sforce:00BA297D
.sforce:00BA297D                mov     eax, ecx        ; Get Opcode 1
.sforce:00BA297F .
.sforce:00BA297F                shl     eax, 17h
.sforce:00BA2982                shr     eax, 17h
.sforce:00BA2985                and     eax, [edi+24h]  ; EAX = 0
[EDI+24] = 0x20    0&&0x20 = 0
.sforce:00BA2988
.sforce:00BA2988 ############################################################
```

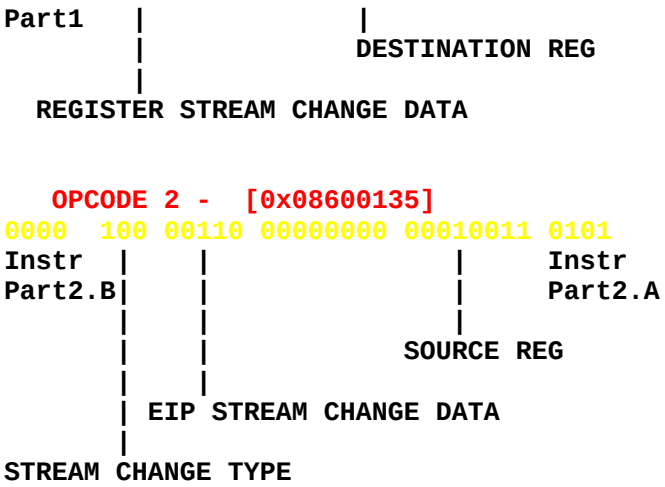
```
.sforce:00BA2988 .
.sforce:00BA2988
.sforce:00BA2988                xor     esi, eax        ; Index
Decrypt with key from opcode, still 0xB3
.sforce:00BA298A
.sforce:00BA298A
.sforce:00BA298A .
.sforce:00BA298A                mov     eax, [edi+1Ch]  ; Get
Instruction Table Offset
.sforce:00BA298D                mov     esi, [eax+esi*4] ; Add our
instruction ID*4
.sforce:00BA2990                add     esi, [edi+8]    ; Add VMBASE
.sforce:00BA2993                jmp     esi             ; Jmp to
Instruction
.sforce:00BA2993 sub_BA28C8     endp
```

StarForce VM RAM

```
EDI =

Stack[0000033C]:0012E238                dd 12E238h   <-> [EDI+0] VM
RAM START
Stack[0000033C]:0012E23C                dd 12E638h   <-> [EDI+4] VM
RAM END
Stack[0000033C]:0012E240                dd 0B97000h  <-> [EDI+8] VM
BASE

Stack[0000033C]:0012E244                dd 15h       <-> [EDI+0C]
      SRC_REG STREAM  - SHL WITH OPCODE DATA
Stack[0000033C]:0012E248                dd 23h       <-> [EDI+10]
      DEST_REG STREAM - SHL WITH OPCODE DATA

Stack[0000033C]:0012E24C                dd 21D2E4h   <-> [EDI+14]
      EIP
Stack[0000033C]:0012E250                dd 0         <-> [EDI+18]
Stack[0000033C]:0012E254                dd 0BCC320h  <-> [EDI+1C]
      INSTRUCTION TABLE
Stack[0000033C]:0012E258                dd 0BA28C8h  <-> [EDI+20]
Stack[0000033C]:0012E25C                dd 8         <-> [EDI+24]
      EIP STREAM - ROL/ROR/BTC/BTS/BTR
Stack[0000033C]:0012E260                dd 302h      <-> [EDI+28]
      EFLAGS
Stack[0000033C]:0012E264                dd 21D2E4h   <-> [EDI+2C]
```

StarForce Example Data Encoding

```
98004200
08600135

   OPCODE 1 -  [0x98004200]
10011 00000 00000 00100001 000000000
  |     |             |       |
Instr   |             |     Instruction Xor
```


Part1
DESTINATION REG
REGISTER STREAM CHANGE DATA
OPCODE 2 - [0x08600135]
0000 100 00110 00000000 00010011 0101
Instr Part2.B
Instr Part2.A
SOURCE REG
EIP STREAM CHANGE DATA
STREAM CHANGE TYPE