Andrea Di Sorbo, PhD student
Sicurezza delle Reti e dei Sistemi Software
CdLM in Ingegneria Informatica
Università degli Studi del Sannio
(disorbo@unisannio.it)

Metamorphic Malware

Implementation of a Metamorphic Engine

Obfuscation techniques

2. **Register usage exchange:** Different mutations of the malware will have the same code, but will use different registers. Signature based-detection is possible through wildcards. Win95.Regswap (December,1998) used this technique.

Version 1

Binary Opcode	Assembly Code
5A	pop edx
BF04000000	mov edi,0004h
8BF5	mov esi,ebp
B80C000000	mov eax,000Ch
81C288000000	add edx,0088h
8B1A	mov ebx,[edx]
899C8618110000	mov [esi+eax*4+00001118],ebx

String Signature:

5ABF040000008BF5B80C00000081C2880000008B1A899C8618110000

Version 2

Binary Opcode	Assembly Code
58	pop eax
BB04000000	mov ebx,0004h
8BD5	mov edx,ebp
BF0C000000	mov edi,000Ch
81C088000000	add eax,0088h
8B30	mov esi,[eax]
89B4BA1811000	mov [edx+edi*4+00001118],esi

String Signature:

58BB040000008BD5BF0C00000081C0880000008B3089B4BA18110000

Exercise 2

- In the metamorphic engine implemented in the previous exercise, add a new method to perform Register Usage Exchange operations on the target code.
- The new method:
  1. Takes in input a file in assembly code (hello_mutation.s)
  2. Returns in output a new variant (hello_mutation2.s) of the input file obtained through operations of Register Usage Exchange (Each execution may produce a different variant of the original file)
- Recompile the resulting file and verify that the two executions (hello_mutation.s and hello_mutation2.s) are equivalent.

hello.s

```
        .file "hello.c"
        .def ___main;   .scl 2;    .type 32;   .endef
        .section .rdata,"dr"
LC0:
        .ascii "Hello world!\0"
        .text
        .globl      _main
        .def _main;     .scl 2;    .type 32;   .endef
_main:
LFB7:
        .cfi_startproc
        pushl %ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        movl %esp, %ebp
        .cfi_def_cfa_register 5
        andl $-16, %esp
        subl $16, %esp
        call ___main
        movl $LC0, (%esp)
        call _puts
        leave
        .cfi_restore 5
        .cfi_def_cfa 4, 4
        ret
        .cfi_endproc
LFE7:
        .ident     "GCC: (GNU) 4.8.3"
        .def _puts;     .scl 2;    .type 32;   .endef
```

Target Code. In this code block we can apply obfuscation techniques

hello_mutation2.s

hello_mutation.s

```
_main:
        movl  %eax, %eax # garbage instruction
LFB7:
        movl  %edi, %edi # garbage instruction
        .cfi_startproc
        pushl %ebp
        movl  %edi, %edi # garbage instruction
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        movl  %esp, %ebp
        .cfi_def_cfa_register 5
        andl  $-16, %esp
        subl  $16, %esp
        call  ___main
        andl  $1, %esi # garbage instruction
        movl  $LC0, (%esp)
        orl   %ecx, %ecx # garbage instruction
        call  _puts
        orl   %ecx, %ecx # garbage instruction
        leave
        .cfi_restore 5
        .cfi_def_cfa 4, 4
        ret
        .cfi_endproc
LFE7:
        .ident     "GCC: (GNU) 4.8.3"
        .def  _puts;     .scl  2;    .type 32;   .endef
```

hello_mutation2.s

```
%eax -> %edi
%edi -> %eax
%edi -> %eax
%esi -> %ecx
%ecx -> %ebx
%ecx -> %ebx
```

```
_main:
        movl  %edi, %edi # register swapped
LFB7:
        movl  %eax, %eax # register swapped
        .cfi_startproc
        pushl %ebp
        movl  %eax, %eax # register swapped
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        movl  %esp, %ebp
        .cfi_def_cfa_register 5
        andl  $-16, %esp
        subl  $16, %esp
        call  ___main
        andl  $1, %ecx # register swapped
        movl  $LC0, (%esp)
        orl   %ebx, %ebx # register swapped
        call  _puts
        orl   %ebx, %ebx # register swapped
        leave
        .cfi_restore 5
        .cfi_def_cfa 4, 4
        ret
        .cfi_endproc
LFE7:
        .ident     "GCC: (GNU) 4.8.3"
        .def  _puts;     .scl  2;    .type 32;   .endef
```

Recompiling and executing hello_mutation2.s

```
Computer@Computer-PC ~
$ gcc -c hello_mutation2.s -o hello_mutation2.o

Computer@Computer-PC ~
$ gcc hello_mutation2.o -o hello_mutation2

Computer@Computer-PC ~
$ ./hello_mutation2
Hello world!

Computer@Computer-PC ~
$ |
```

Some useful tips

- A static swap (e.g., %eax is always substituted by %edi), would produce at each iteration the same code.
- The substitutions should be established in a random manner at each iteration.
- Designed substitutions should be applied for all the instructions of the target code, (e.g. if %ebx -> %eax, %eax should be replaced by %ebx every time it appears in the code).
- A designed register may replace only one other register (e.g., %eax and %ecx cannot be both replaced by %ebx)

Some useful tips

- Through a data structure you can keep track of the substitutions.
- Stack pointer (%esp) and base pointer (%ebp) registers should not be replaced

Register	Replaced by
%eax	%edi
%ebx	%esi
%ecx	%ebx
%edx	%edx
%edi	%eax
%esi	%ecx

```
_main:
        movl  %edi, %edi # register swapped
LFB7:
        movl  %eax, %eax # register swapped
        .cfi_startproc
        pushl %ebp
        movl  %eax, %eax # register swapped
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        movl  %esp, %ebp
        .cfi_def_cfa_register 5
        andl  $-16, %esp
        subl  $16, %esp
        call  ___main
        andl  $1, %ecx # register swapped
        movl  $LC0, (%esp)
        orl   %ebx, %ebx # register swapped
        call  _puts
        orl   %ebx, %ebx # register swapped
        leave
        .cfi_restore 5
        .cfi_def_cfa 4, 4
        ret
        .cfi_endproc
```