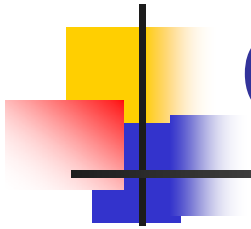
Metamorphic Code

Advanced Computer Networks

(705.010 / 705.011)

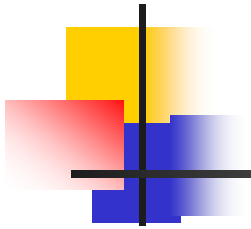
SS 2005

Malin Jürgen



Overview

- Introduction
  - Why polymorphic or metamorphic code?
  - Use the force
  - Difference between Metamorphic and Polymorphic code
  - Polymorphic code cycle
- Metamorphic code
  - Levels of Metamorphism
  - Code confusion – opcode level
  - Problems
- Personal interest



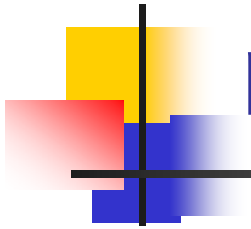
Introduction

- Some state of the art viruses use metamorphic code engines to “hide” their deciphering routines:

  Evol, Zperm, … → Z0mbie

- Because of identifying that viruses is a great problem, that lead us to some questions:
  - What is metamorphic code?
  - Can it be used for shellcode?
  - Are there problems?

Why polymorphic or metamorphic code?

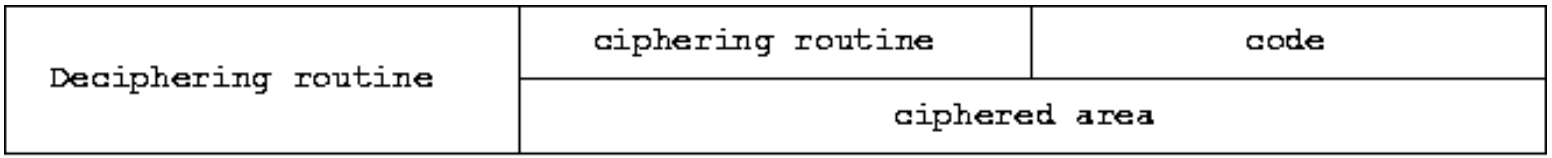


- Shellcode and Virus detection is “easy” if there are no caveats
- Virus scanners and IDS systems search for similar code parts
  - Producers of malicious code have to prevent this:
    → ciphering techniques
    → hide/exchange of decryption part
- Timing constraints are friendly to the bad!

Use the force

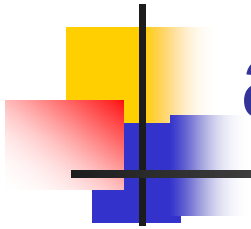
- Virus
- Shellcode

→ “Vulnerability“ is the deciphering routine



Malicious code structure (virus, shellcode)

Difference between Metamorphic and Polymorphic code

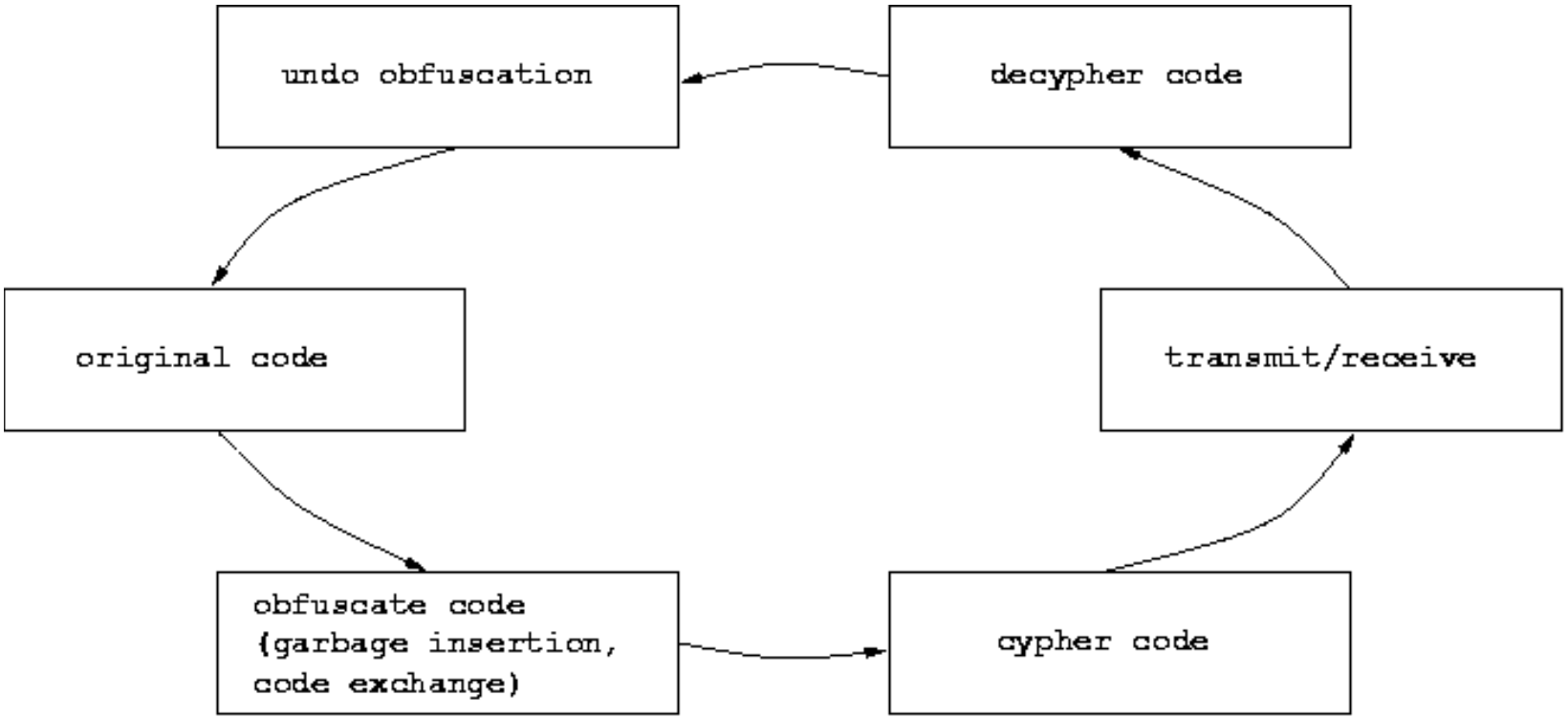


- Polymorphic code always “returns” to the code origin

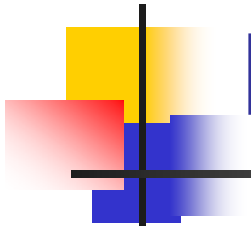
- Metamorphic code doesn’t!

  It creates an incredible amount of different versions by using simple (theoretical!) methods.

Polymorphic code cycle



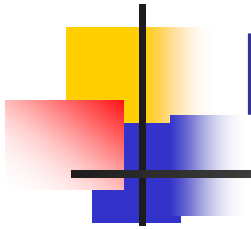
Malicious polymorphic code cycle



Levels of Metamorphism

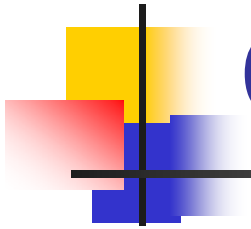
- Algorithm level
  - → Interchangeable blocks of algorithmic code may have variants
- Opcode level
  - → Each pseudo-language element may be converted into different sets of opcode

Levels of Metamorphism - Example



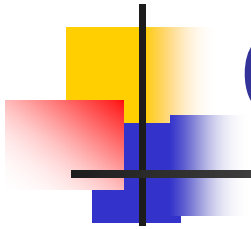
```
Task: calculate f(x) = 10 * sin(2 * x)

  ------------------------ algorithm-level ---------------------->
  |
  |        f(x) = 10*sin(2*x)            f(x) = sin(x)*cos(x)*20
  |
  |        variant 1                     variant 2
  |
  |          t = x                         t = x
 opcode-     t = t * 2                   a = sin(t)
-level       t = sin(t)                  b = cos(t)
  |          t = t * 10                  t = a * b
  |                                      t = t * 20
  |
  |        variant 3                     variant 4
  |
  |        t = 2 * x                     b = cos(x)
  |        a = sin(t)                    t = 20 * b
  |        t = 10 * a                    a = sin(x)
  |                                      t = t * a
```



Code confusion – opcode level

- Swap sequence of blocks/procedures
  → *n* blocks lead to *n!* different generations!
- Insert NOPs (*garbage insertion*)
  → NOP – opcode tables
- Insert Jumps and/or Calls
- Exchange well known code sequences
  → `xor eax, eax` changes to
  `sub eax, eax`



Code confusion – opcode level

- Change relative addresses
  - `mov edx,5151EC8Bh`

    → `mov edx,5FC000CBh`
  - code alignment techniques!
- Register exchange

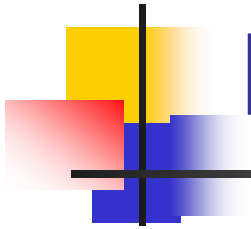
  `mov [esi],edi`

  → `mov [esi],ebx`

NOP equivalent opcodes for shellcodes

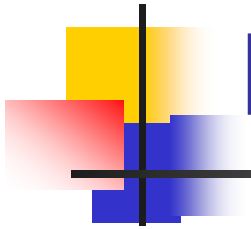
Arch	Code (hex, 00=wild)	Opcode
HPPA	08 21 02 9a	xor %r1,%r1,%r26
HPPA	08 41 02 83	xor %r1,%r2,%r3
HPPA	08 a4 02 46	or %r4,%r5,%r6
HPPA	09 04 06 8f	shladd %r4,2,%r8,%r15
HPPA	09 09 04 07	sub %r9,%r8,%r7
HPPA	09 6a 02 8c	xor %r10,%r11,%12
HPPA	09 cd 06 0f	add %r13,%r14,%r15
Sprc	20 bf bf 00	bn -random
IA32	27	daa '
IA32	2f	das /
IA32	33 c0	xor %eax,%eax
IA32	37	aaa 7
IA32	3f	aas ?
IA32	40	inc %eax @
IA32	41	inc %ecx A
IA32	42	inc %edx B
IA32	43	inc %ebx C
IA32	44	inc %esp D
IA32	45	inc %ebp E
IA32	46	inc %esi F
IA32	47	inc %edi G
IA32	48	dec %eax, H
IA32	4a	dec %edx J
IA32	4b	dec %ebx K
IA32	4c	dec %esp L
IA32	4d	dec %ebp, M
IA32	4e	dec %esi N
IA32	4f	dec %edi O

...



Problems

- Metamorphic code generation is very complex
- No general code generation engines available – *fortunately!*
- opcode level generation requires assembler pro
- Self-modifying code has some operating system spin offs!? (e.g. caching)



Personal interest

- Finding and using latest information to metamorphic code generation
- Developing and writing of a simple metamorphic engine
- Use the engine on shellcodes and challenge with detection-group at IAIK ;-)

Questions?