Japan Computer Emergency Response Team Coordination Center

JPCERT CC®

Understanding Malware

2015/08/13 Security Camp 2015 10-D

JPCERT/CC Analysis Center
You NAKATSURU

Notice

- These training materials are used for "Security Camp 2015" in Japan
  - Security training program for students to discover & nurture young talent
  - https://www.ipa.go.jp/jinzai/camp/ (Japanese only)
- The training course consists of the following 2 parts
  - Malware, Malware analysis basics, Static analysis basics
    - Learning basic knowledge for malware analysis
  - Malware analysis
    - Understanding details of malware samples using static analysis method
- The training mainly focuses on 32bit Windows malware
- Some slides have display problems due to animation
- Any questions and comments are welcome
  - Please contact us at aa-info@jpcert.or.jp

Agenda

- Malware Basics
- Malware Analysis Basics
- Static Analysis Basics

Objectives of This Session

Understanding malware

- What malware is
- What malware does
- Malware trends
- Typical prevention/response methods

Understanding malware analysis

- What malware analysis is
- Malware analysis methods
- Static analysis techniques

Malware Basics

Definition

Malicious Software

- Broader in concept than a computer virus
  - Virus, Worm, Trojan Horse, Rootkit, Bot, DoS Tool, Exploit kit, Spyware

Malware Purpose

Mischief

- Crashing a system
- DoS

For Profit

- Havoc via DDoS
- Sending Spam
- Visiting affiliate sites

Others

- Stalking
- Self-assertion

For Profit

Selling

- Sensitive information
- Malware, malware builder

Sending spam emails

- Rental business

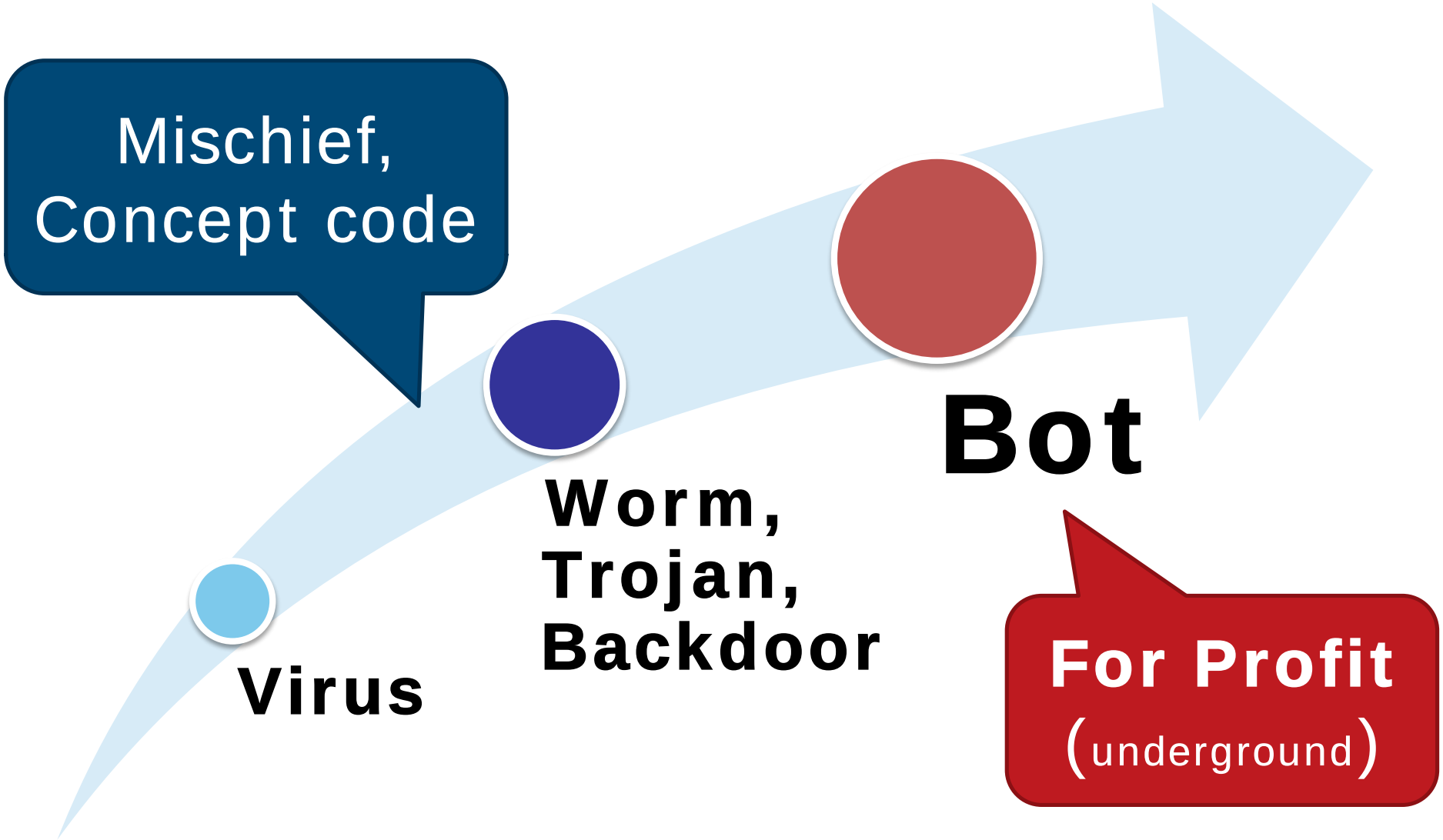
DDoS

- Blackmail

Affiliate

- Let user access to the site using malware

Growth of Malware



Infection Method

Software

- Attack software vulnerabilities
  - OS, Office, Browser
- Make machines to execute malware

Human

- Trick users to execute malware
  - provide a line about software contents
  - camouflage an "icon"

Exploiting Software Vulnerability

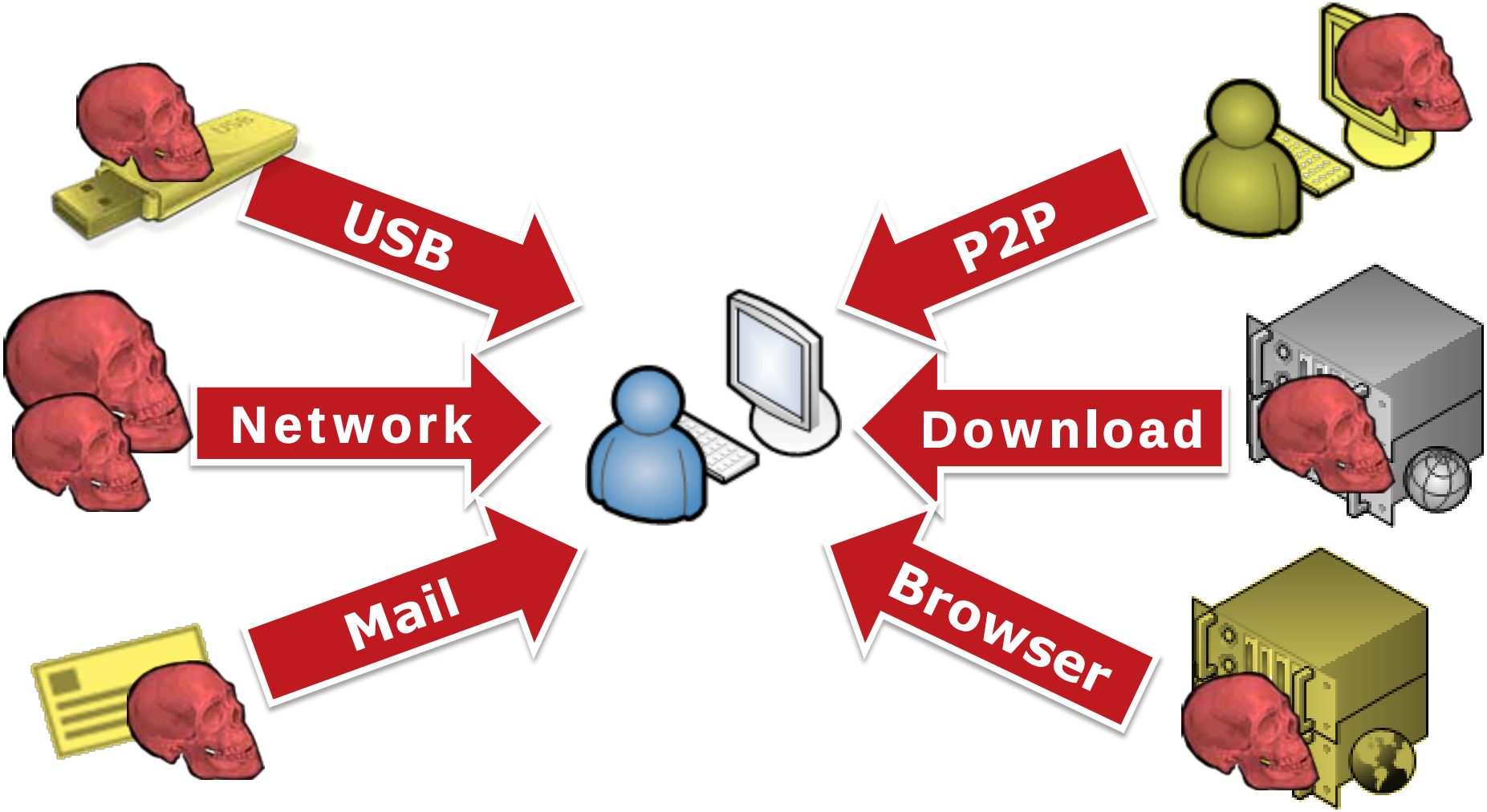
Attack Vulnerability → Execute arbitrary code

- Buffer overflow, etc.
- Take control and execute arbitrary code

- Shellcode for malware execution
- Malware

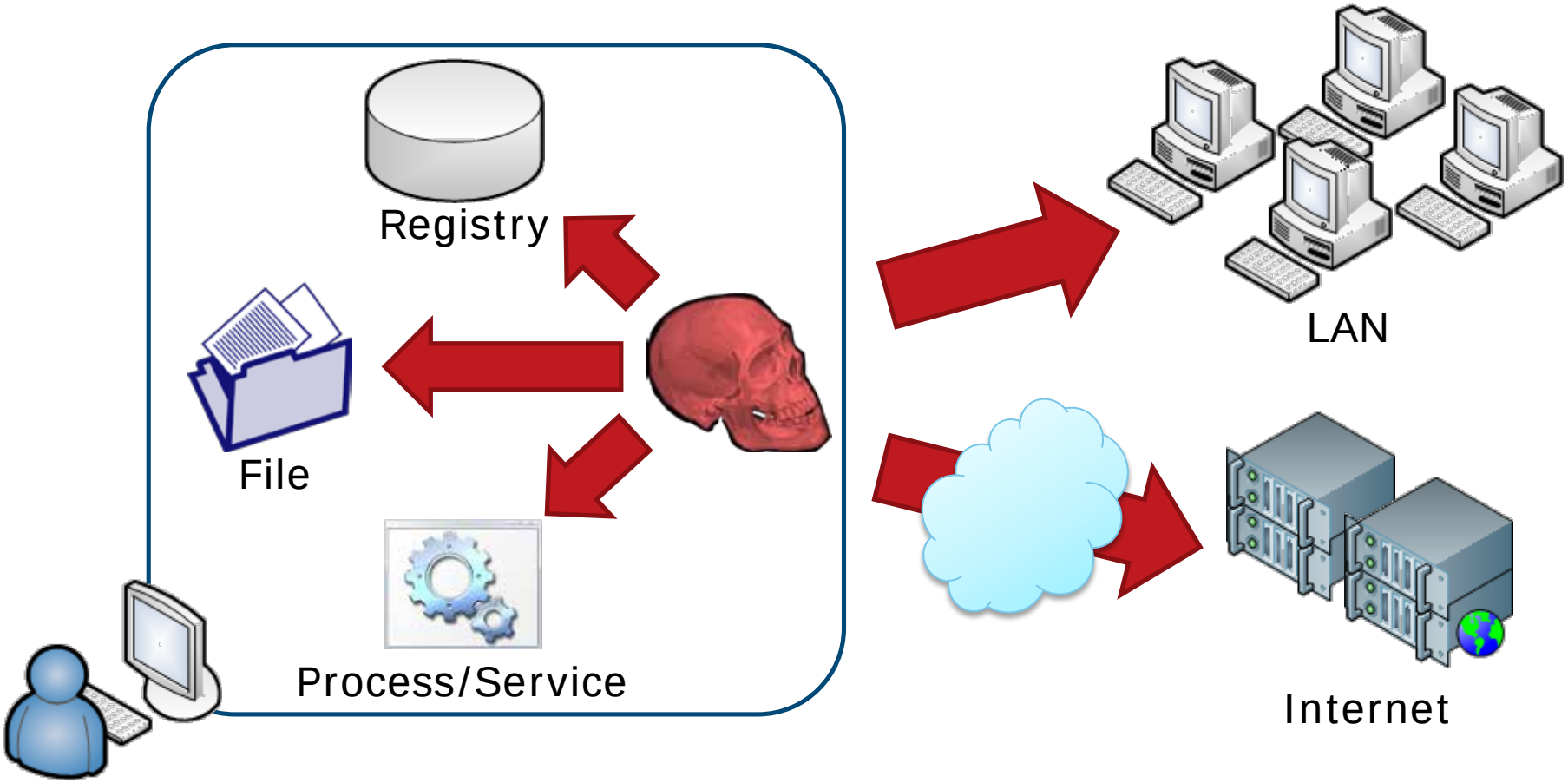
Malware Attack Vector

- Vulnerability, Social engineering



Malware Behavior

- Can do anything on the infected machine
  - Within the limits of infected user's privilege



Malware Behavior: Installation

Create main module

- download, creation

Copy / Delete itself

- copy to the system folder

Run after reboot

- registry entry related to Autorun
- Startup folder
- register as a service

Malware Behavior: Modifying System

Disable security features

- Windows Firewall
- Windows Update

Avoid security programs

- Anti-Virus software
- Analysis tool

Hide itself

- modifying other processes

Malware Behavior: Main Behavior

Steal information

- read registry entries / config files
- key logging, packet capture

Bot

- connect to C&C servers
  - execute commands

Spread

- attack other machines

Important Points

Network activity is important

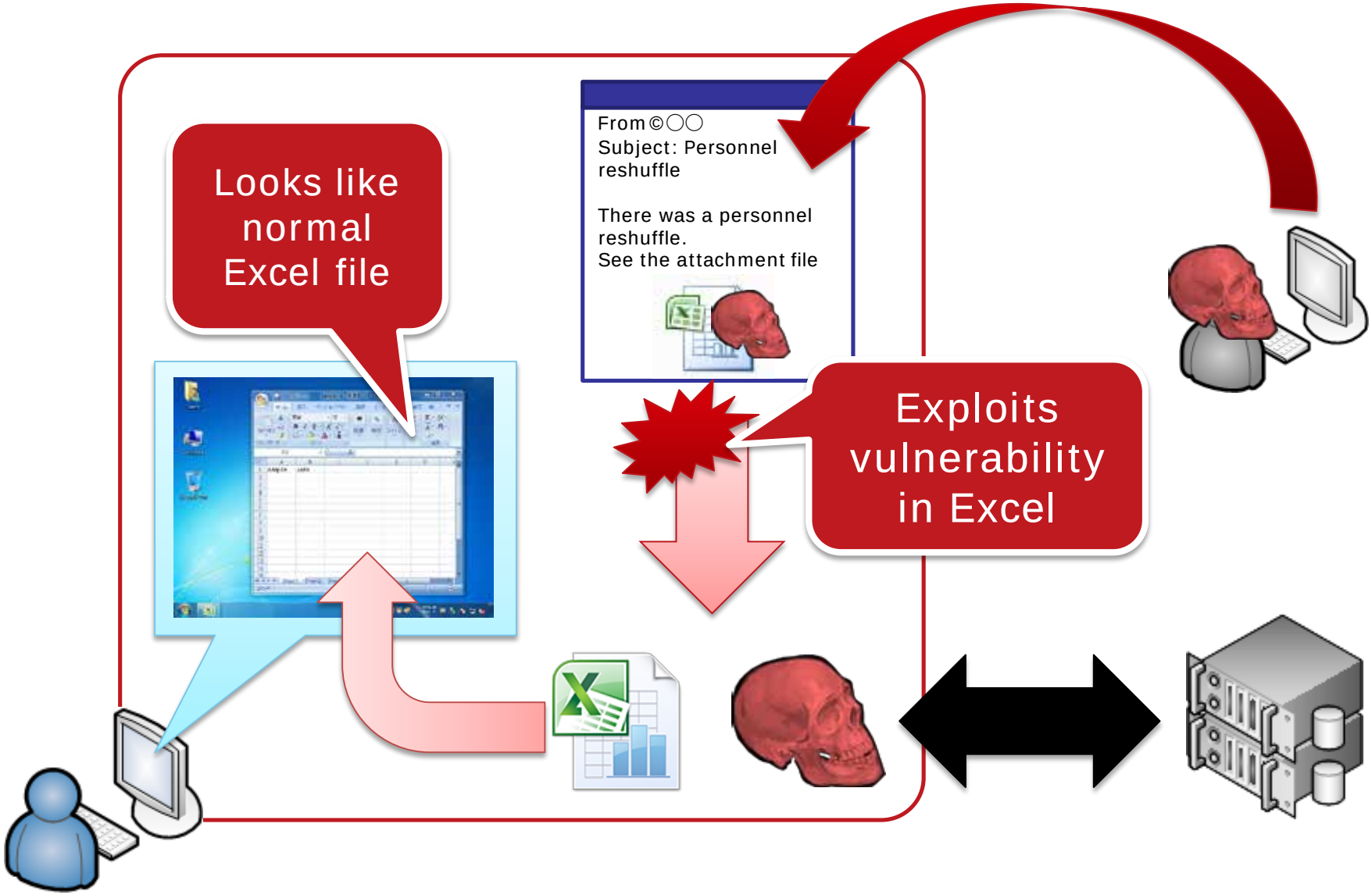
- Attackers need to take out information for profit
- Able to recognize damage by analyzing packets

Do not trust infected machines

- Possibility of data falsification
  - Such as anti-virus software results on infected machines
    - Recommended to re-install Windows
- Preventing malware infection is the most important

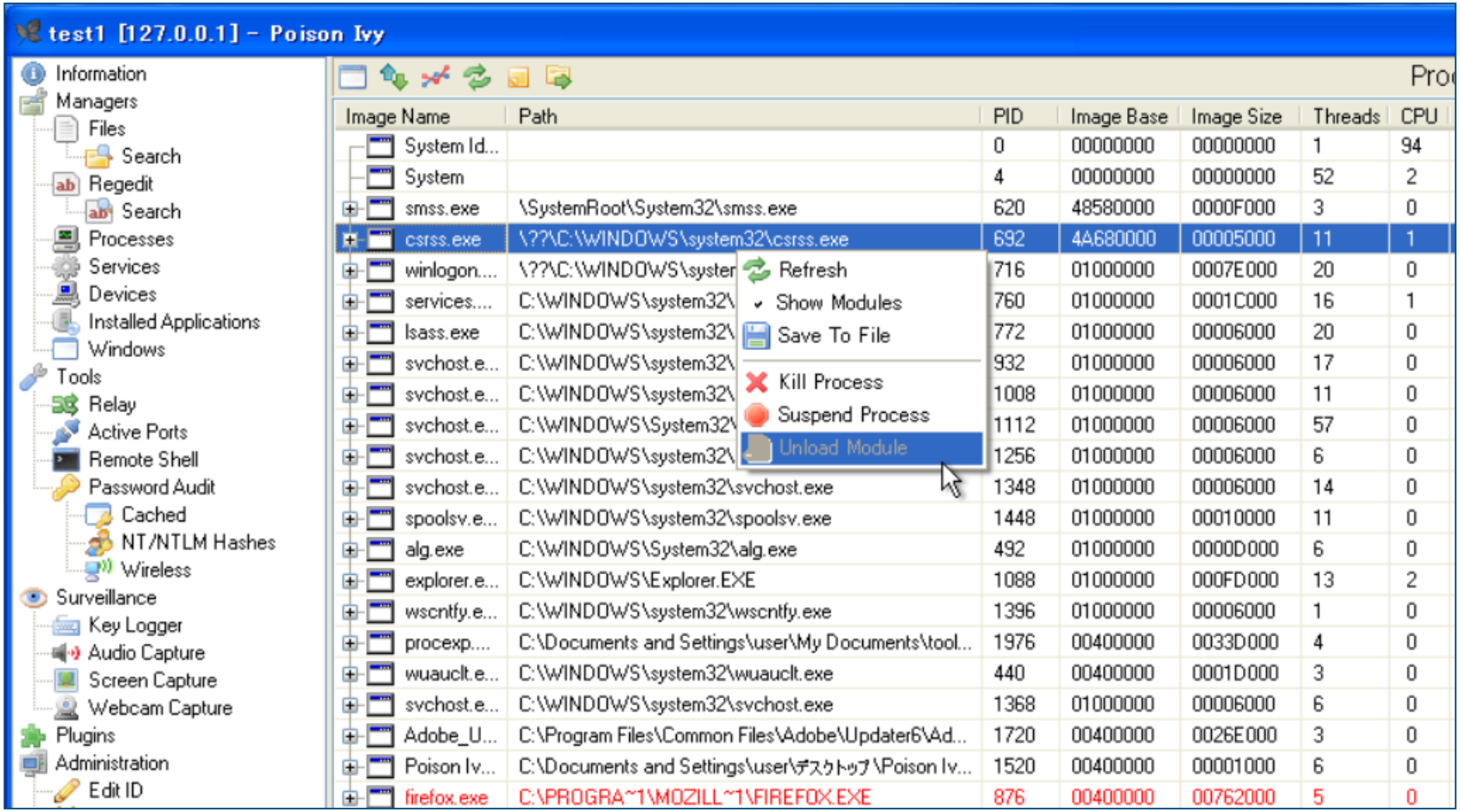
MALWARE EXAMPLE

Targeted Attack



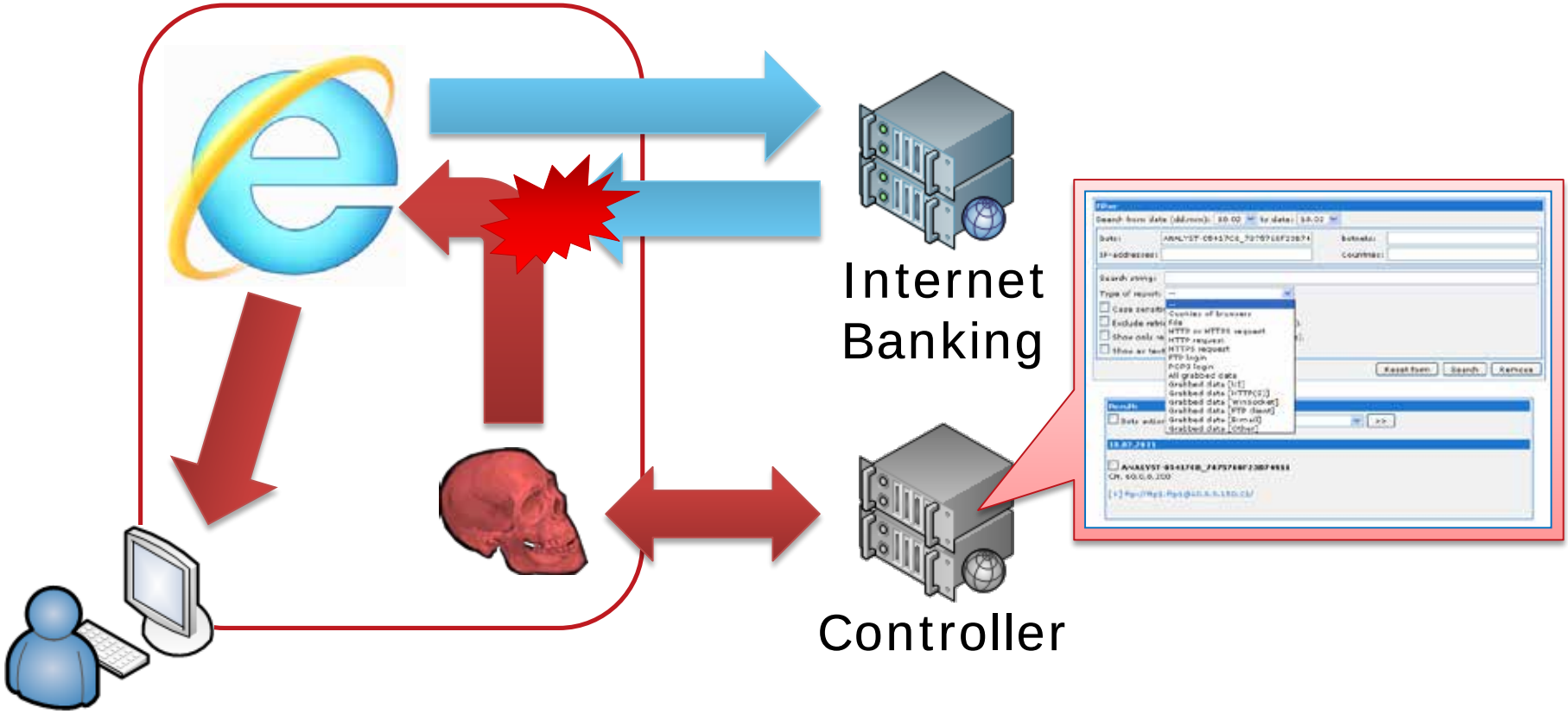
RAT

- Remote Administration Tool/Trojan
  - —Often used for targeted attack



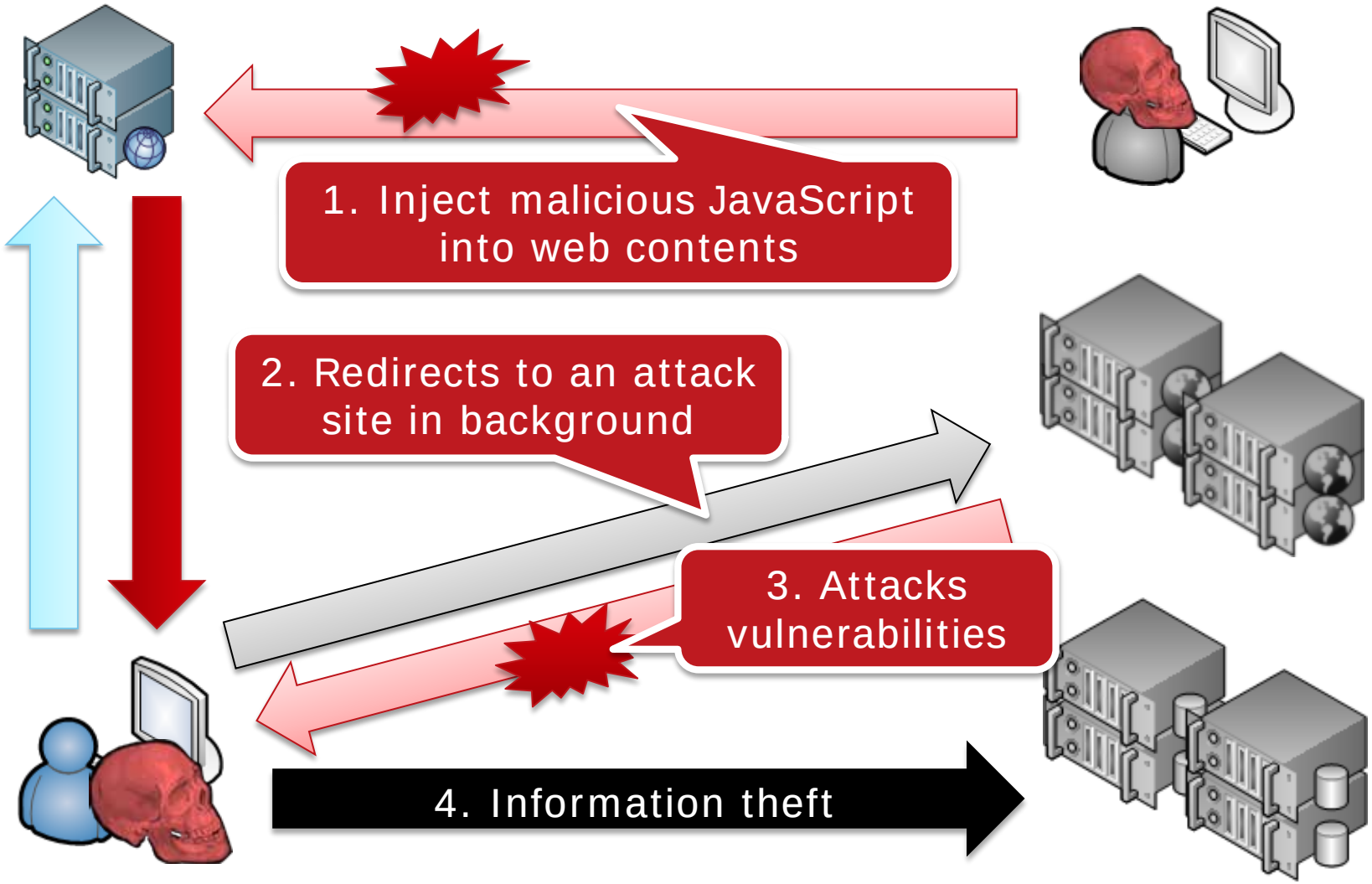
Banking Trojan

- Attempts to steal users credential of the Internet banking
  - Inject additional input form on the web page



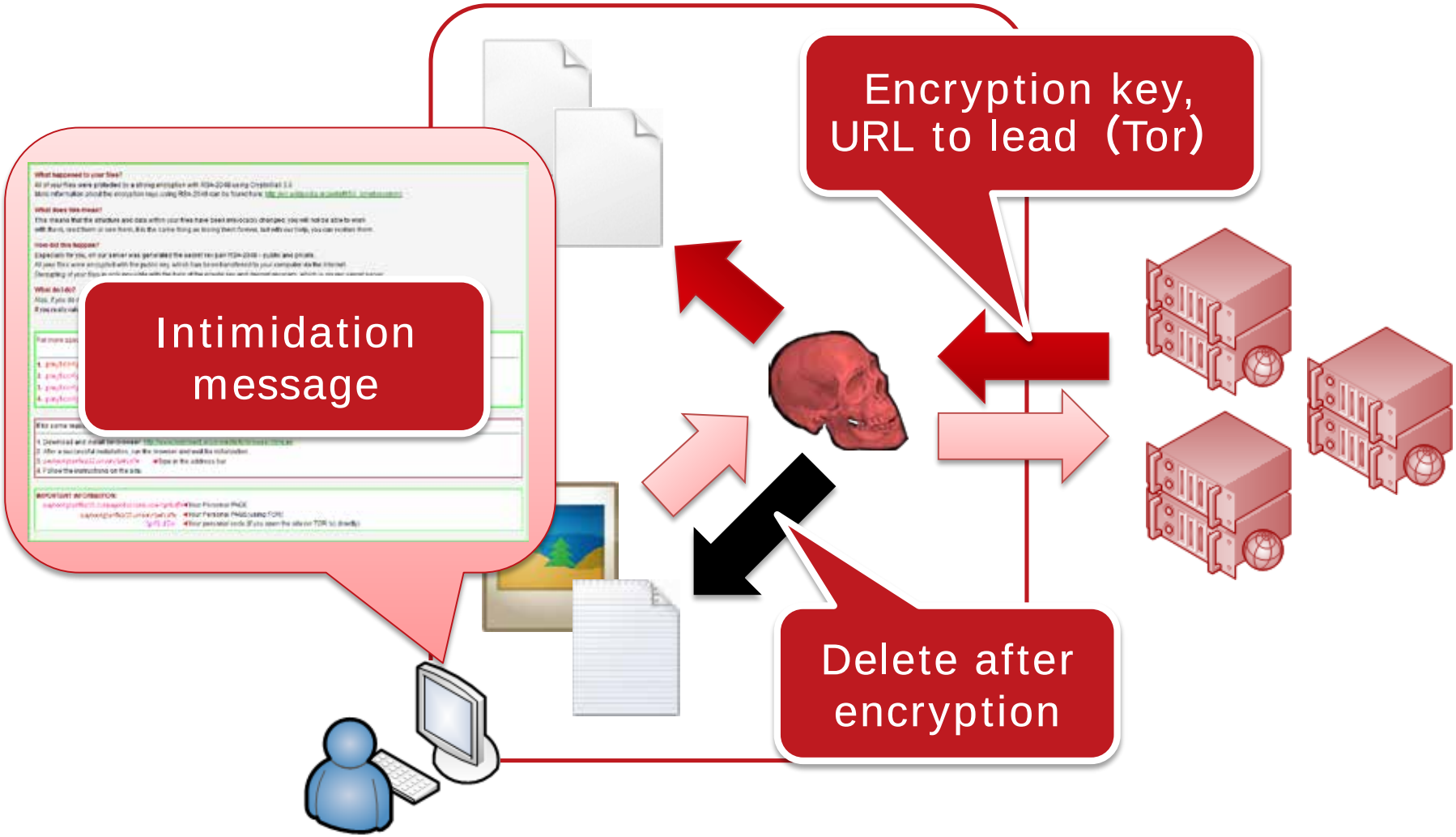
Web-based Attack

- Attacking web browsers or add-ons



Ransomware

- "All your important files are encrypted!"



Android Malware

- Re-package popular legitimate apps
  - Distributed through
    - 3rd party market
    - Android Market

MALWARE TRENDS IN JAPAN

Banking Trojan

- ZeuS, Citadel, Gameover were over

Vawtrak

Dyre

Tsukuba

Tinba

Dridex

Chthonic

KINS

Ransomware

- Spread through Drive-by-Download attack

https://isc.sans.edu/forums/diary/Angler+exploit+kit+pushes+new+variant+of+ransomware/19681

JPCERT-AT-2015-0015
JPCERT/CC
2015-05-26

<<< JPCERT/CC Alert 2015-05-26 >>>

ランサムウエア感染に関する注意喚起

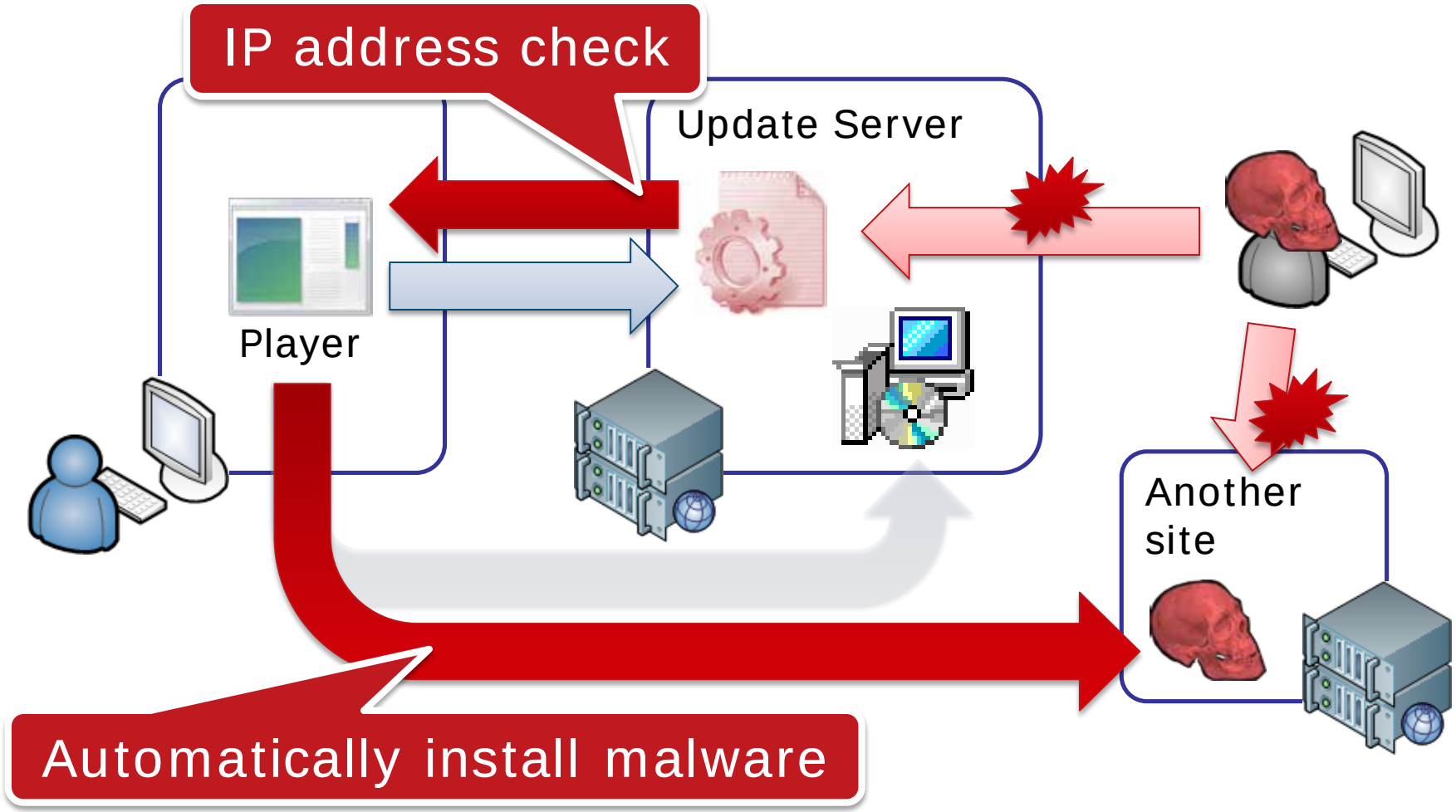
https://www.jpcert.or.jp/at/2015/at150015.html

I. 概要

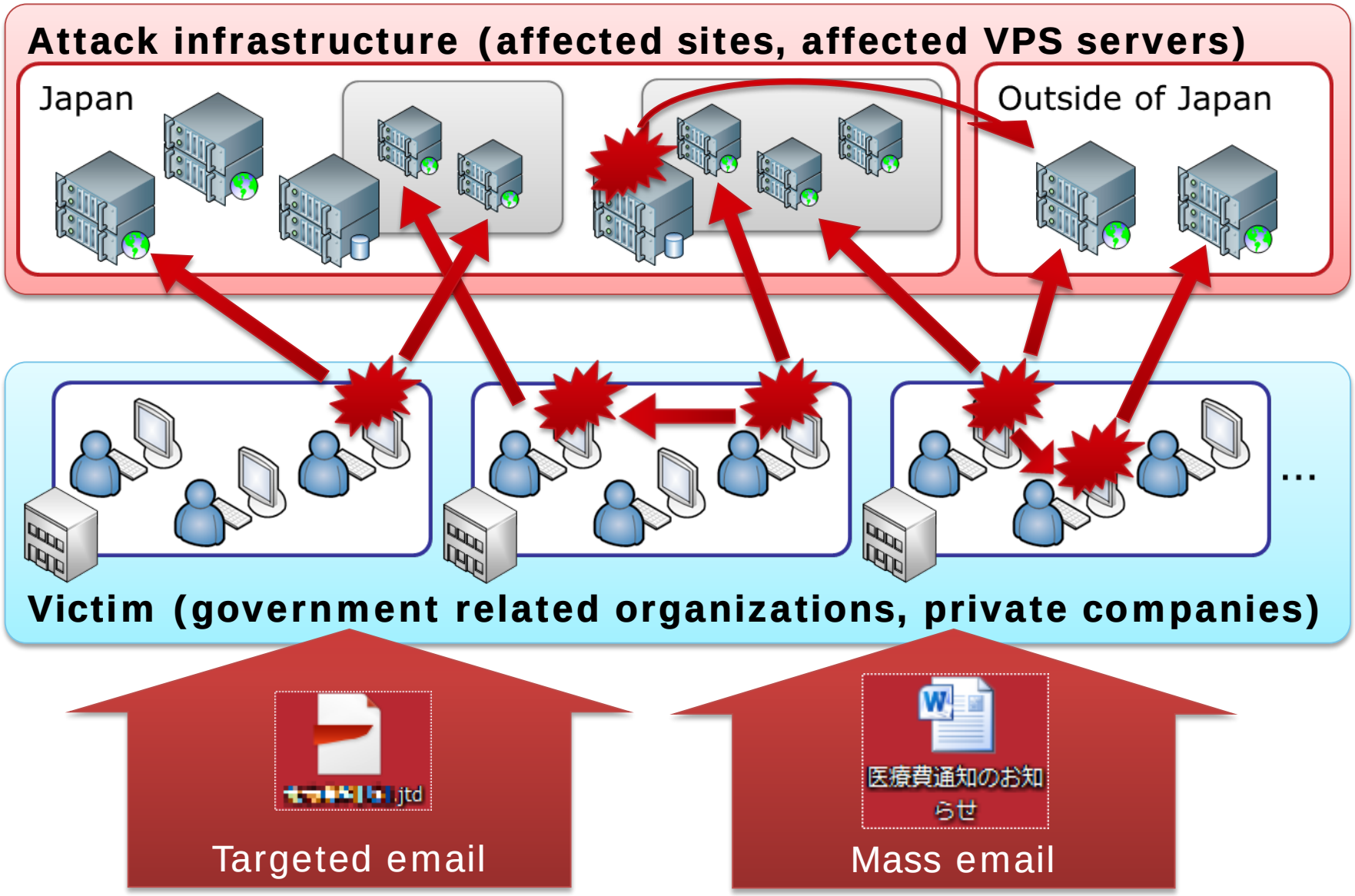
JPCERT/CC では、いわゆるランサムウエアと呼ばれるマルウエアを用いて、端末内のファイルを暗号化し、復号の為に金銭等を要求する攻撃の被害を多数確認しています。

Targeted Attack: Watering Hole Attack

- A case of compromised site for a media player update



CloudyOmega / Blue Termite



ANTI-MALWARE

Typical Malware Prevention

Fix vulnerabilities

- Update OS & software
- Configure security options for OS & software

Use anti-virus software

- Possibly false results

Do not open a file without confirming

- Beware of social engineering

Typical Response

Disconnect network connection first

- To stop information leakage & attack to outside

Re-install OS

- Basically malware can do anything on infected machines

Recurrence prevention

- Identify & fix up the cause of infection

Worldwide Activity

Botnet takedown

- Microsoft, FBI, anti-virus vendors, etc.
- Major activities
  - Rustock takedown
  - ZeroAccess takedown
  - Citadel takedown

Convention on Cybercrime

- Drawn up by the Council of Europe
  - Convention for co-investigation of cyber criminals

Malware Analysis Basics

Who Analyzes Malware?

- CSIRTs
- Security product developers
- Security service providers
- Anti-malware researchers
- Software developers
- Law enforcement

Why Analyze Malware?

- Incident response
- Product development/improvement
- Signature creation
- Cutting-edge countermeasure
- Vulnerability analysis
- Criminal arrest

Malware Analysis Method

- Environment Setup
- Malware collection
- Surface analysis
- Runtime analysis
- Static analysis

IMPORTANT POINT

Security is a Key for Success

Analyze malware with great care

- If you make a mistake, it may bring serious consequences

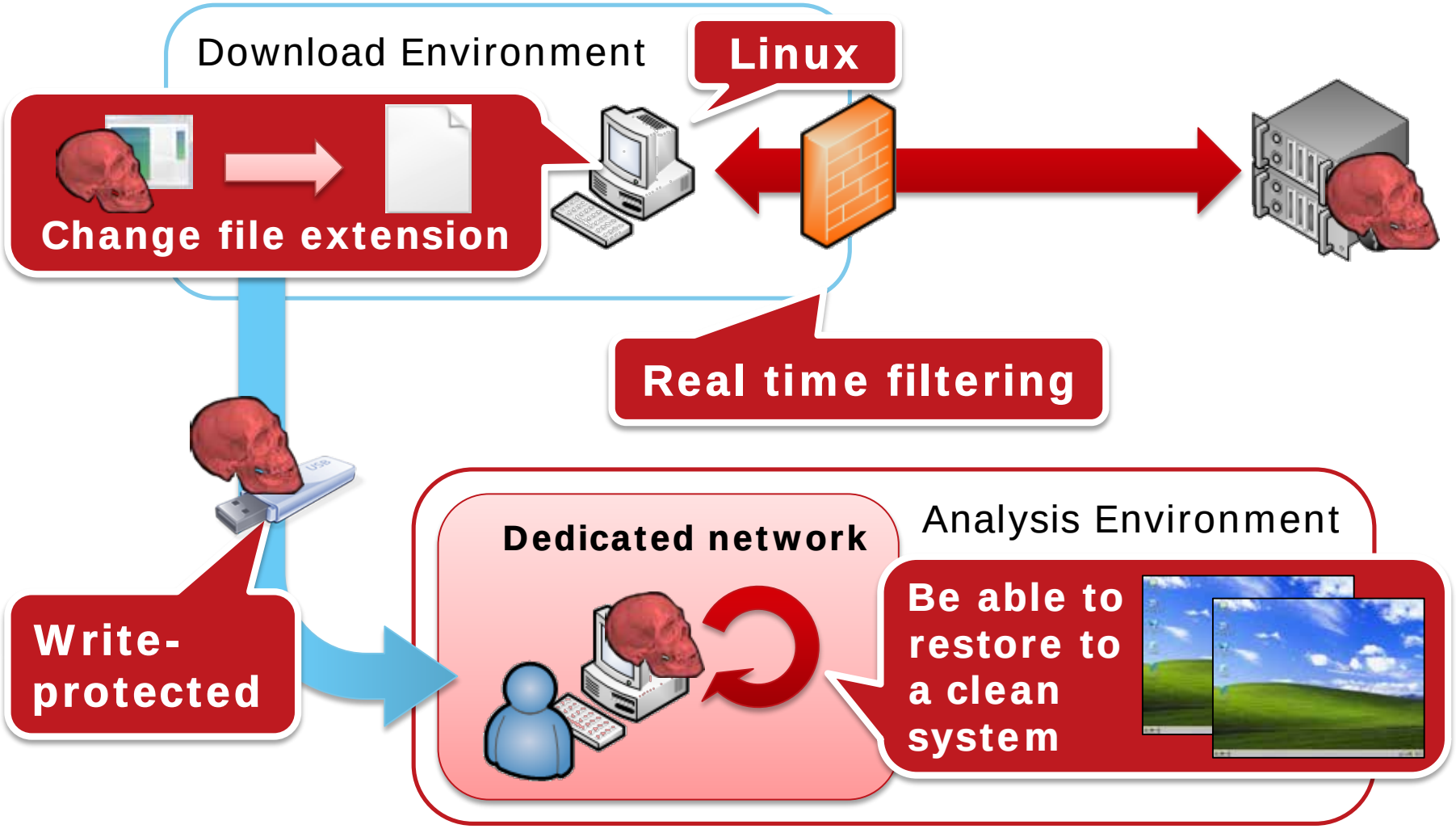
Develop environment with great care

- Pay great attention to environment for malware download and analysis

Publish results with great care

- Take great care in publishing details of malware
  - e.g. 0-day vulnerability

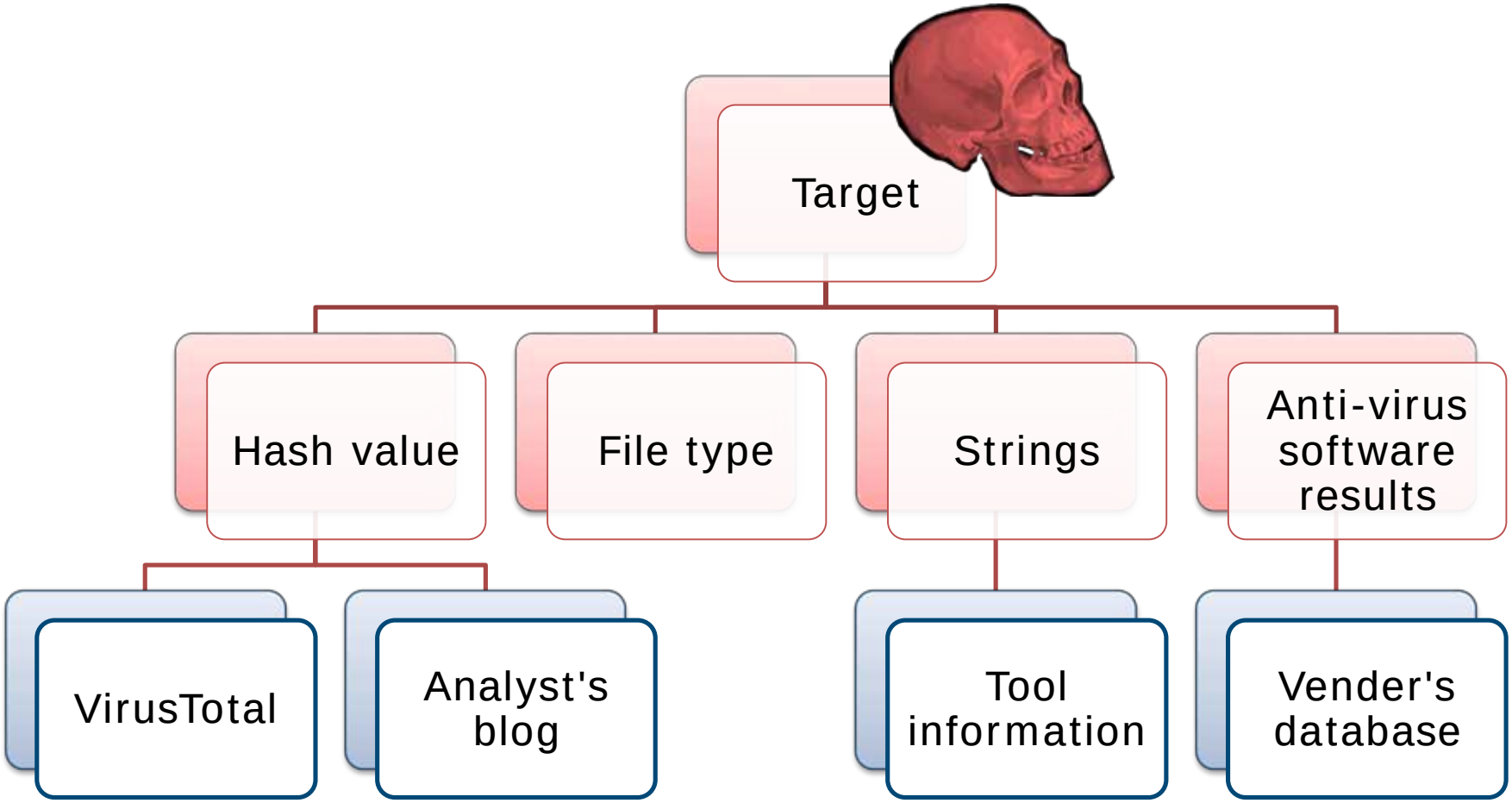
Sample Analysis Environment



SURFACE ANALYSIS

Surface Analysis

- Retrieve surface information from targets without execution



Runtime Analysis

- Execute malware and monitor its behavior
  - Difficult to reveal "all" of malware's behavior

Manual Analysis

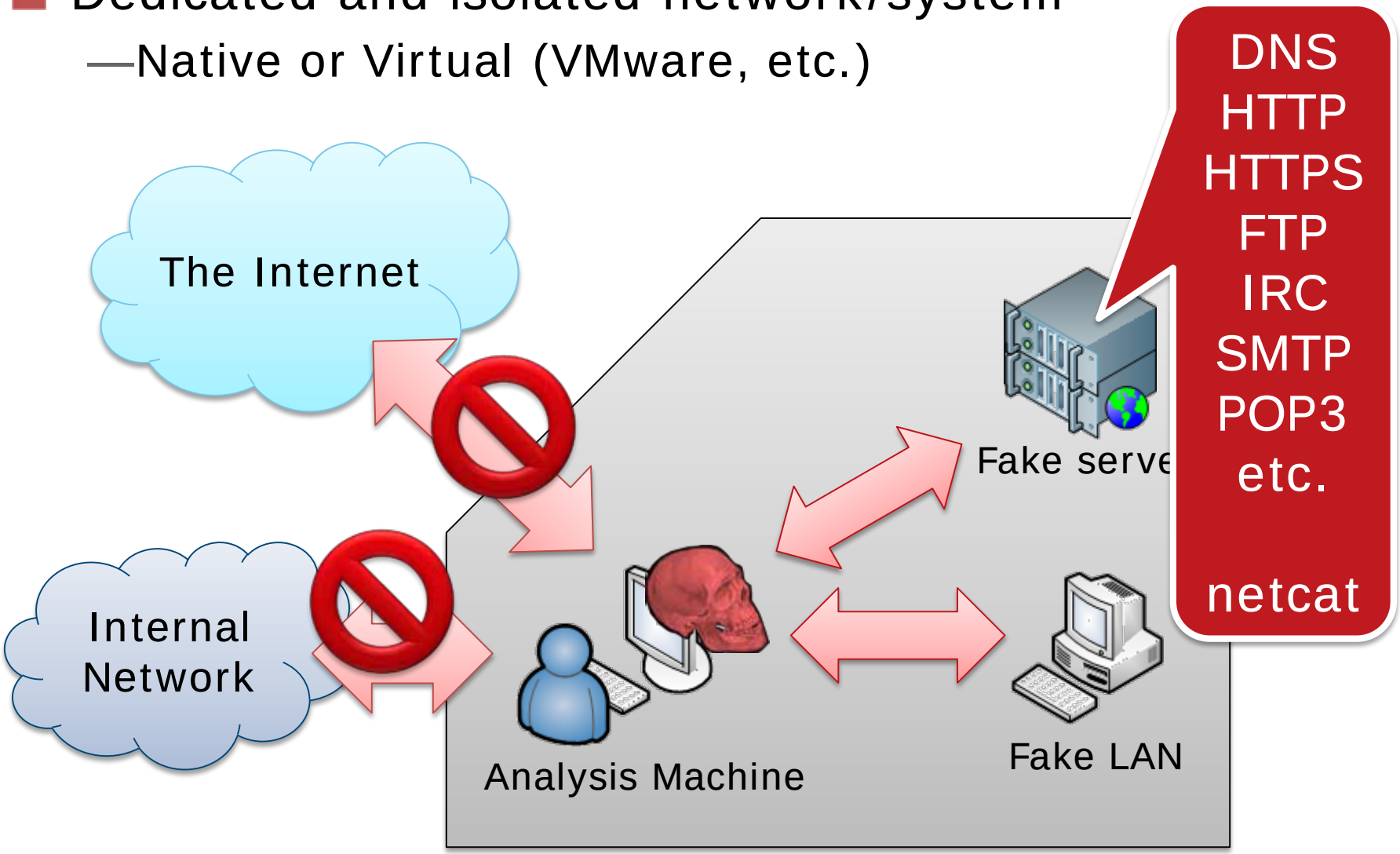
- Use monitoring tools on analysis environment
  - Sysinternals suite, etc.

Automation

- Use sandbox system
  - Free / Commercial products
- Can reduce analysis time

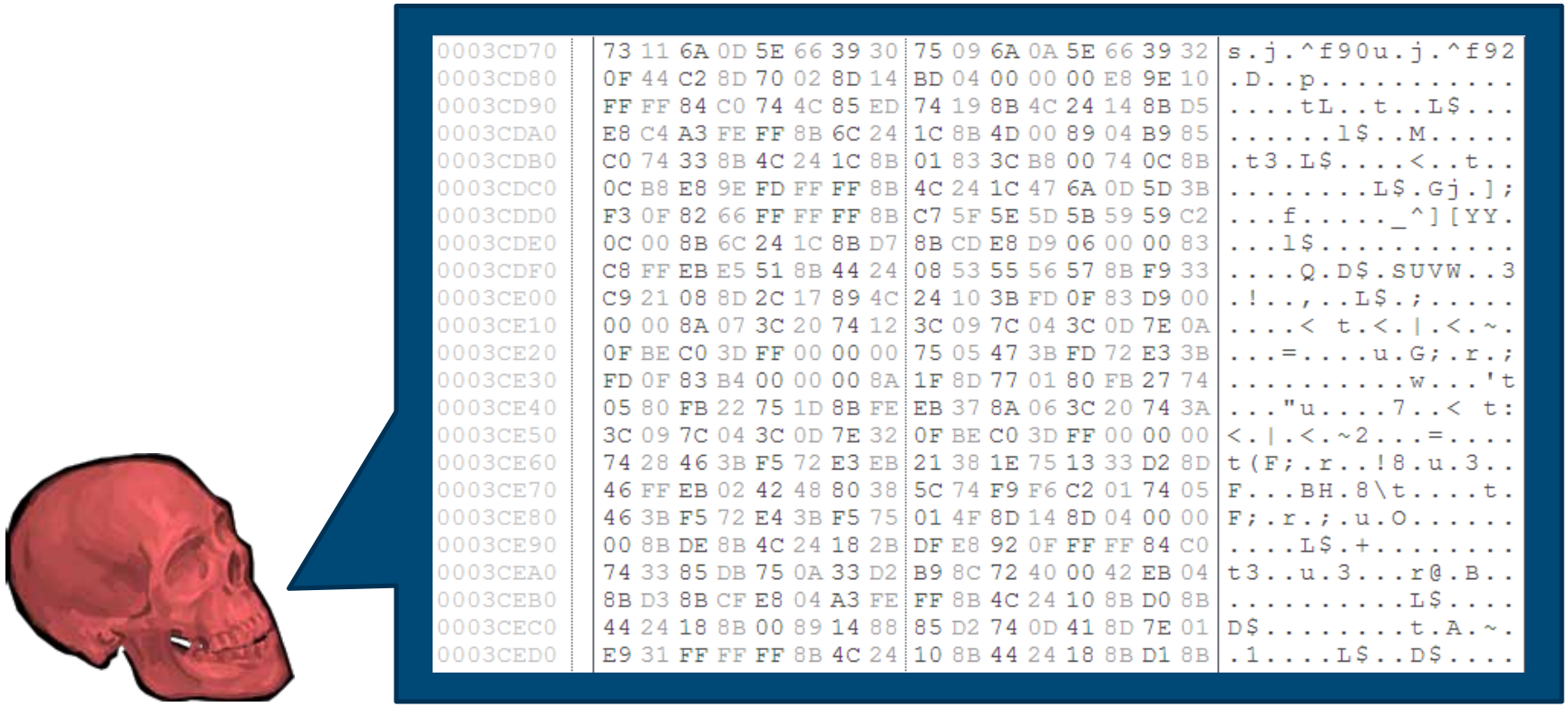
Runtime Analysis Environment

- Dedicated and isolated network/system
  - Native or Virtual (VMware, etc.)



What Static Analysis is

- Reading code in binary file and understanding its functionality
  - —Takes a long time
  - —Requires deep and broad knowledge



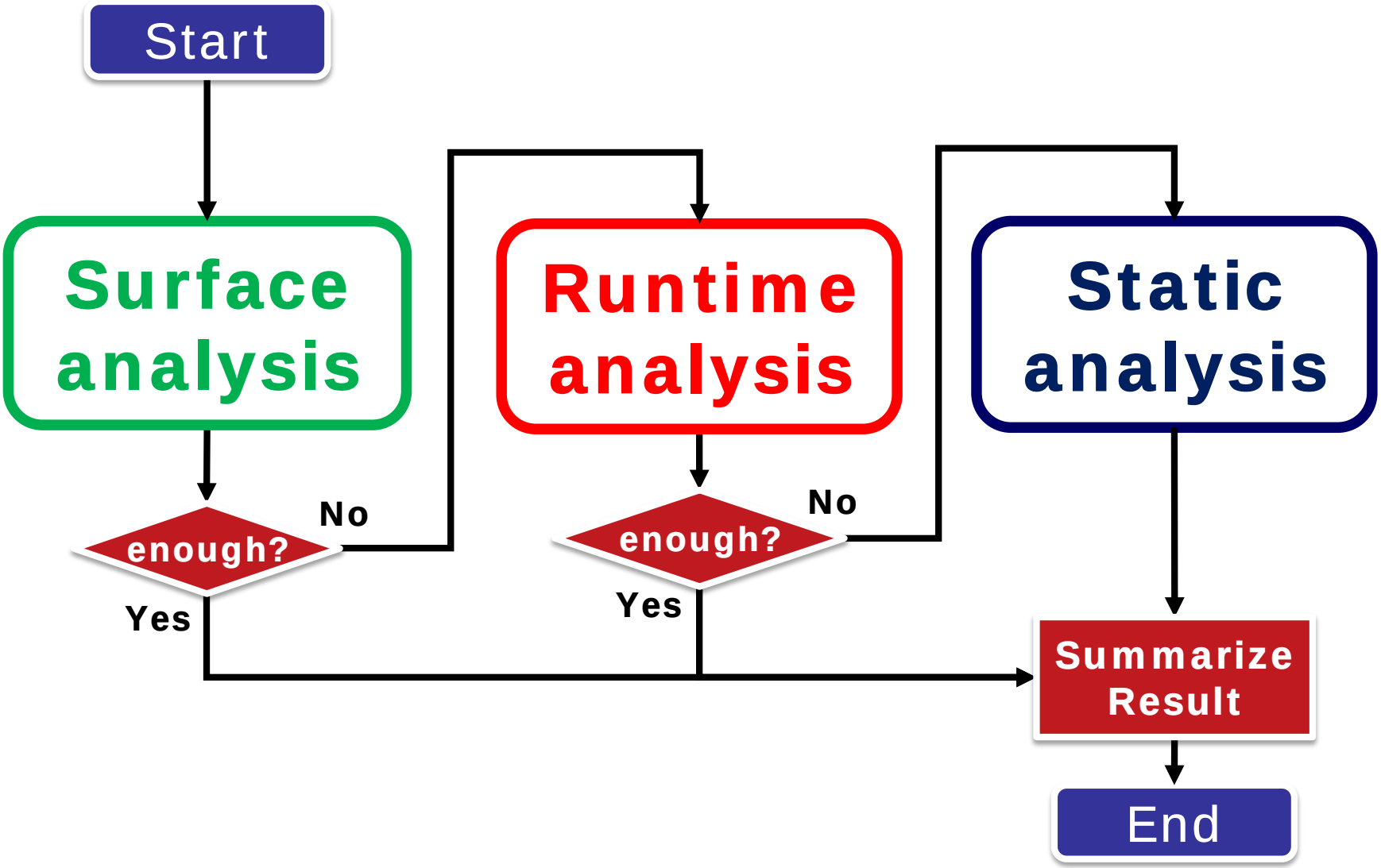
```
0003CD70  73 11 6A 0D 5E 66 39 30 75 09 6A 0A 5E 66 39 32  s.j.^f90u.j.^f92
0003CD80  0F 44 C2 8D 70 02 8D 14 BD 04 00 00 00 E8 9E 10  .D..p...........
0003CD90  FF FF 84 C0 74 4C 85 ED 74 19 8B 4C 24 14 8B D5  ....tL..t..L$...
0003CDA0  E8 C4 A3 FE FF 8B 6C 24 1C 8B 4D 00 89 04 B9 85  ......l$..M.....
0003CDB0  C0 74 33 8B 4C 24 1C 8B 01 83 3C B8 00 74 0C 8B  .t3.L$....<..t..
0003CDC0  0C B8 E8 9E FD FF FF 8B 4C 24 1C 47 6A 0D 5D 3B  ........L$.Gj.];
0003CDD0  F3 0F 82 66 FF FF FF 8B C7 5F 5E 5D 5B 59 59 C2  ...f.....^][YY.
0003CDE0  0C 00 8B 6C 24 1C 8B D7 8B CD E8 D9 06 00 00 83  ...l$...........
0003CDF0  C8 FF EB E5 51 8B 44 24 08 53 55 56 57 8B F9 33  ....Q.D$.SUVW..3
0003CE00  C9 21 08 8D 2C 17 89 4C 24 10 3B FD 0F 83 D9 00  .!..,..L$.;.....
0003CE10  00 00 8A 07 3C 20 74 12 3C 09 7C 04 3C 0D 7E 0A  ....< t.<.|.<.~.
0003CE20  0F BE C0 3D FF 00 00 00 75 05 47 3B FD 72 E3 3B  ...=....u.G;.r.;
0003CE30  FD 0F 83 B4 00 00 00 8A 1F 8D 77 01 80 FB 27 74  ..........w...'t
0003CE40  05 80 FB 22 75 1D 8B FE EB 37 8A 06 3C 20 74 3A  ..."u....7..< t:
0003CE50  3C 09 7C 04 3C 0D 7E 32 0F BE C0 3D FF 00 00 00  <.|.<.~2...=....
0003CE60  74 28 46 3B F5 72 E3 EB 21 38 1E 75 13 33 D2 8D  t(F;.r..!8.u.3..
0003CE70  46 FF EB 02 42 48 80 38 5C 74 F9 F6 C2 01 74 05  F...BH.8\t....t.
0003CE80  46 3B F5 72 E4 3B F5 75 01 4F 8D 14 8D 04 00 00  F;.r.;.u.O......
0003CE90  00 8B DE 8B 4C 24 18 2B DF E8 92 0F FF FF 84 C0  ....L$.+........
0003CEA0  74 33 85 DB 75 0A 33 D2 B9 8C 72 40 00 42 EB 04  t3..u.3...r@.B..
0003CEB0  8B D3 8B CF E8 04 A3 FE FF 8B 4C 24 10 8B D0 8B  ..........L$....
0003CEC0  44 24 18 8B 00 89 14 88 85 D2 74 0D 41 8D 7E 01  D$........t.A.~.
0003CED0  E9 31 FF FF FF 8B 4C 24 10 8B 44 24 18 8B D1 8B  .1....L$..D$....
```

ANALYSIS PROCESS

Analysis Process Comparison

	Surface analysis	Runtime analysis	Static analysis
Overview	Retrieve surface information from targets **without execution**	**Execute samples** and monitor its behavior	**Read codes** in binary files and understand its functionality
Output	- Hash values - Strings - File attributes - Packer info - Anti-virus detection info	Activity of - File system - Registry - Process - Network	Malware's functionality e.g. - Bot commands - Encode/decode methods
Security risk	Low	**High**	Moderate
Analysis coverage	Low	Moderate	**High**

Malware Analysis Flow



Static Analysis Basics

Important Points

No need to know all of malware

- You need much time for static analysis

Need much knowledge/experiences

- Need
  - OS knowledge
  - Assembly basics
  - Efficient reading techniques
  - Anti-analysis techniques
- We have to continue studying

Disassemble & Decompile

- Binary -> source code

```
55
8B EC
81 EC 04 01 00 00
83 7D 0C 01
74 06


33 C0
C9
C2 0C 00



68 04 01 00 00
8D 85 FC FE FF FF
50
6A 00
FF 15 2C 60 00 10
8D 85 FC FE FF FF
6A 5C
50
E8 83 01 00 00
```

```
; BOOL __stdcall DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvF
_DllMain@12     proc near               ; CODE XREF: ___DllMainCRTStartup
                                        ; ___DllMainCRTStartup+7B↓p

var_108         = dword ptr -108h
Filename        = byte ptr -104h
hinstDLL        = dword ptr  8
fdwReason       = dword ptr  0Ch
lpvReserved     = dword ptr  10h

                push    ebp

                retn    0Ch
; ---------------------------------------------------------------------

loc_10001E9:                            ; CODE XREF: DllMain(x,x,x)+0F↑j
                push    104h            ; nSize
                lea     eax, [ebp+Filename]
                push    eax             ; lpFilename
                push    0               ; hModule
                call    ds:GetModuleFileNameA
                lea     eax, [ebp+Filename]
                push    5Ch             ; unsigned int
                push    eax             ; unsigned __int8 *
```

Disassemble

```
BOOL __stdcall DllMain(HINSTANCE hinstDLL,
{
  unsigned __int8 *v4; // eax@3
  unsigned __int8 *v5; // eax@4
  __int16 v6; // ax@5
  CHAR Filename; // [sp+4h] [bp-104h]@3

  if ( fdwReason == 1 )
  {
    v5 = _mbsinc(v4);
    if ( !_mbscmp(v5, (const unsigned __i
    {
      v6 = (unsigned int)GetKeyboardLayou
      if ( v6 == 1033 )
      {
        sub_1000126C();
      }
```

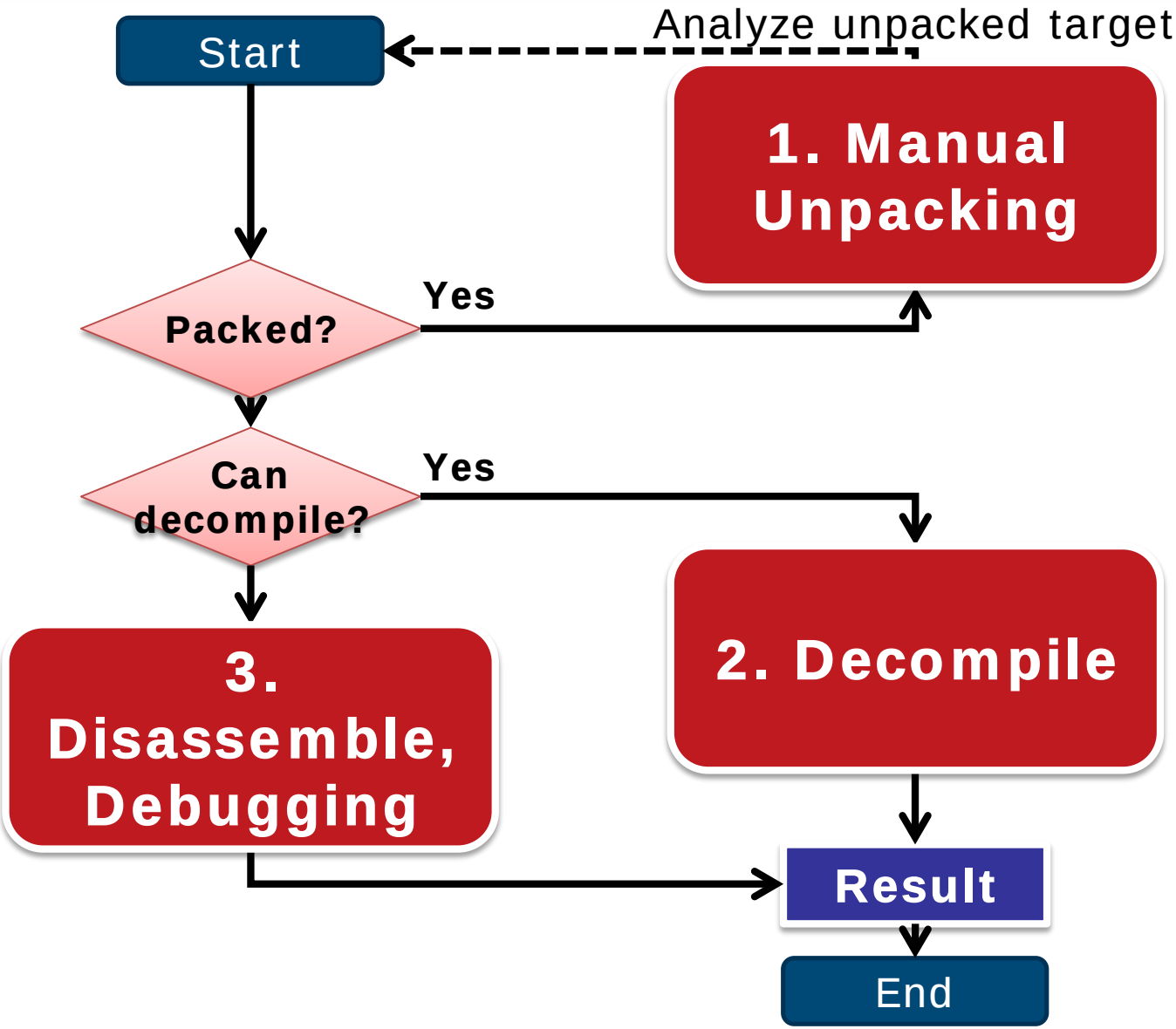
Decompile

Debugging

- Read assembly code while executing step by step

```
- [CPU - main thread, module test]
File View Debug Plugins Options Window Help

Address   Hex dump    Disassembly                          Comment
004012E8  E8 7C030000 CALL test.00401669
004012ED  E9 B3FDFFFF JMP test.004010A5
004012F2  8BFF        MOV EDI,EDI
004012F4  55          PUSH EBP
004012F5  8BEC        MOV EBP,ESP
004012F7  8B45 08     MOV EAX,DWORD PTR SS:[EBP+8]
004012FA  8B00        MOV EAX,DWORD PTR DS:[EAX]
004012FC  8138 63736DE0 CMP DWORD PTR DS:[EAX],E06D7363
00401302  75 2A       JNZ SHORT test.0040132E
00401304  8378 10 03  CMP DWORD PTR DS:[EAX+10],3
00401308  75 24       JNZ SHORT test.0040132E
0040130A  8B40 14     MOV EAX,DWORD PTR DS:[EAX+14]
0040130D  3D 20059319 CMP EAX,19930520
00401312  74 15       JE SHORT test.00401329
00401314  3D 21059319 CMP EAX,19930521
00401319  74 0E       JE SHORT test.00401329
0040131B  3D 22059319 CMP EAX,19930522
00401320  74 07       JE SHORT test.00401329
00401322  3D 00409901 CMP EAX,1994000
00401327  75 05       JNZ SHORT test.0040132E
00401329  E8 D6030000 CALL <JMP.&MSVCR100.?terminate@@YAXXZ>
0040132E  33C0        XOR EAX,EAX
00401330  5D          POP EBP
00401331  C2 0400     RETN 4
00401334  68 F2124000 PUSH test.004012F2
00401339  FF15 24204000 CALL DWORD PTR DS:[<&KERNEL32.SetUnhandl kernel32.SetUnhandledExceptionFilt
0040133F  33C0        XOR EAX,EAX
00401341  C3          RETN
00401342 - FF25 9C204000 JMP DWORD PTR DS:[<&MSVCR100._amsg_exit> MSVCR100._amsg_exit
00401348  6A 14       PUSH 14
0040134A  68 F8214000 PUSH test.004021F8
00401669=test.00401669

crtexe.c:361.

Registers (FPU)
EAX 77483665 kernel32.BaseThreadInitThunk
ECX 00000000
EDX 004012E8 OFFSET test.<ModuleEntryPoint>
EBX 7EFDE000
ESP 0018FF8C ASCII "w6Hw"
EBP 0018FF94
ESI 00000000
EDI 00000000

EIP 004012E8 test.<ModuleEntryPoint>

C 0  ES 002B 32bit 0(FFFFFFFF)
P 1  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 1  DS 002B 32bit 0(FFFFFFFF)
S 0  FS 0053 32bit 7EFDD000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr ERROR_SUCCESS (00000000)

EFL 00010246 (NO,NB,E,BE,NS,PE,GE,LE)

ST0 empty 0.0
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 0.0
ST7 empty 0.0
               3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0  (GT)
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1

Address   Hex dump                                         ASCII
00403000  FF FF FF FF FF FF FF FF 00 00 00 00 00 00 00 00  . . .
00403010  FE FF FF FF 01 00 00 00 4E E6 40 BB B1 19 BF 44  . . .
00403020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ......
00403030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ......
00403040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ......
00403050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ......
00403060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0018FF8C  77483677  RETURN to kernel32.77483677
0018FF90  7EFDE000
0018FF94  0018FFD4
0018FF98  77E99D42  RETURN to ntdll.77E99D42
0018FF9C  7EFDE000
0018FFA0  767E2333
0018FFA4  00000000
0018FFA8  00000000
```

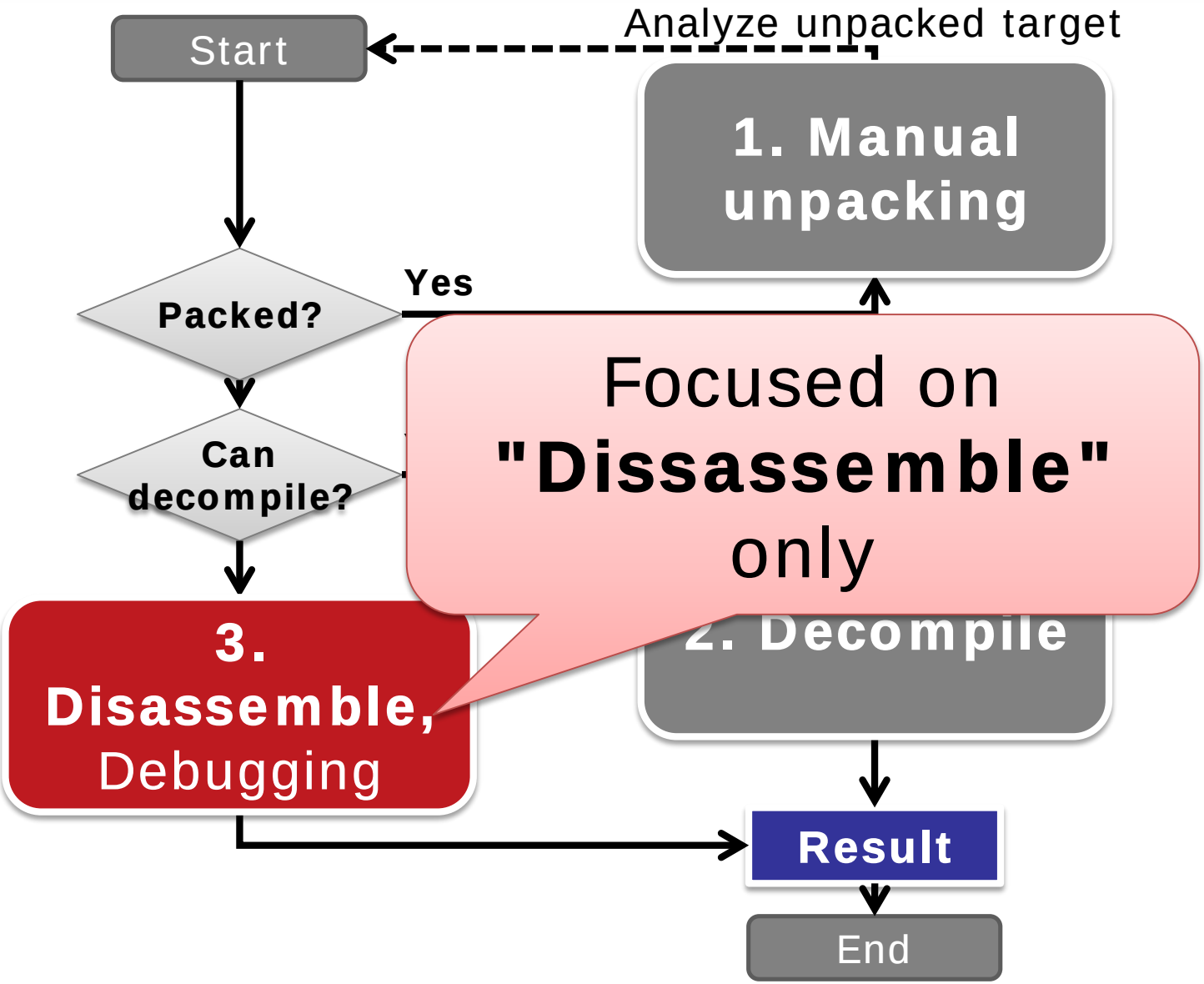
Static Analysis Flow



Static Analysis Tools

Category	Name	Description
Disassembler	IDA	Disassembles more than 50 architectures
Decompiler	Hex-rays Decompiler	x86/ARM binary to C source code
	VB Decompiler	Visual Basic binary to Visual Basic source code
	.NET Reflector	.NET binary to .NET source code
Debugger	OllyDbg	World famous x86 debugger
	Immunity Debugger	Python familiar x86 debugger

Static Analysis Basics



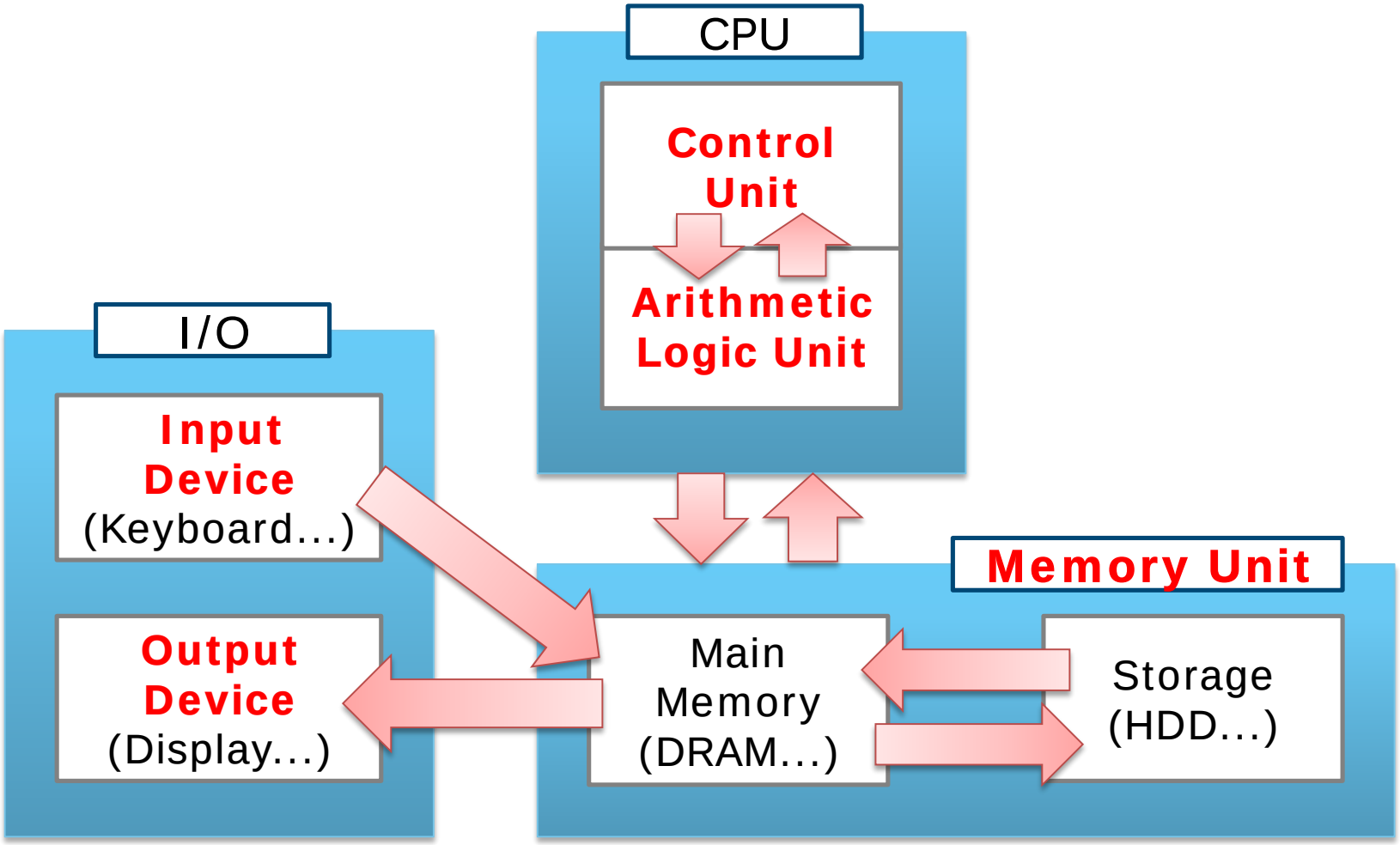
Interactive DisAssembler

- http://www.hex-rays.com/idapro/
  - 3 versions are available

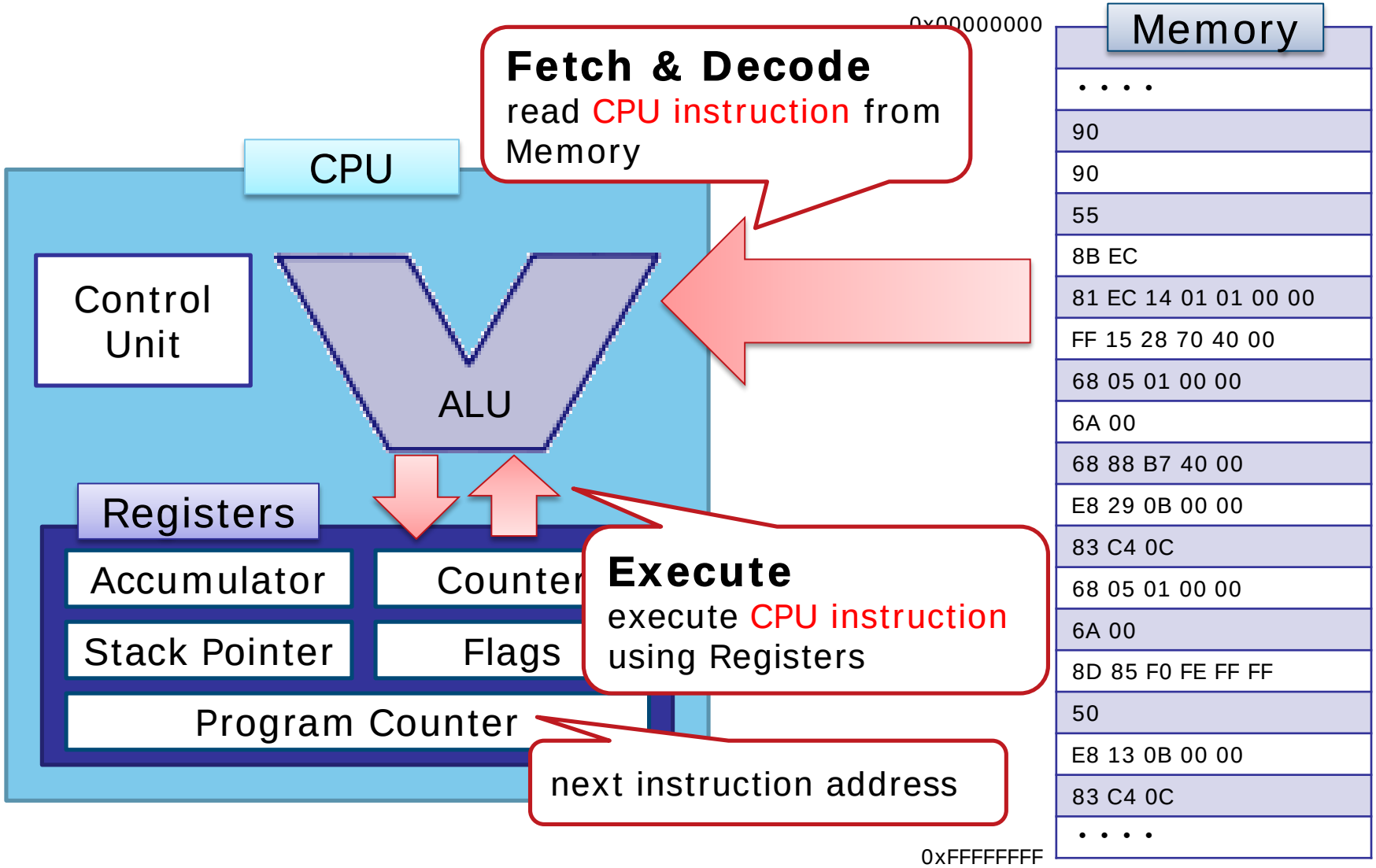
	Free 	Starter 	Pro 
Version	Ver. 5.0 Ver. 6.8 demo	Ver. 6.8	Ver. 6.8
Cost	Free	• 589USD/User • 879USD/Computer	• 1129USD/User • 1689USD/Computer
Features	• Old or limited	• Supports up to 20 processes	• Supports up to 50 processes • Can analyse files for 64 bit platforms

ASSEMBLY BASICS

Components of Computer System



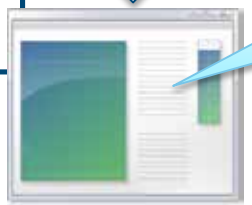
CPU Operation



Compiling Source Code

```
int main(){
  int a,b,c;

  a = 1;
  b = 2;
  c = a + b;

  printf("Answer is %d¥n", c);

  return 0;
}
```

Binary file includes CPU instructions (Machine Code)



Executable file

```
C:¥Test>add.exe
Answer is 3.

C:¥Test>_
```

Execute CPU instructions

Disassemble

- Machine code to assembly code (human readable)

```
55
8B EC
81 EC 04 01 00 00
83 7D 0C 01
74 06


33 C0
C9
C2 0C 00



68 04 01 00 00
8D 85 FC FE FF FF
50
6A 00
FF 15 2C 60 00 10
8D 85 FC FE FF FF
6A 5C
50
E8 83 01 00 00
```

```
; BOOL __stdcall DllMain(HINSTANCE hinstDLL, DWORD fdwR
_DllMain@12     proc near               ; CODE XREF: __
                                        ; ___DllMainCRT


var_108         = dword ptr -108h
Filename        = byte ptr -104h
hinstDLL        = dword ptr  8
fdwReason       = dword ptr  0Ch
lpvReserved     = dword ptr  10h

                push    ebp
                mov     ebp, esp
                sub     esp, 104h
                cmp     [ebp+fdwReason], 1
                jz      short loc_100014E9

loc_100014E3:                           ; CODE XREF: Dl
                xor     eax, eax
                leave
                retn    0Ch
; ---------------------------------------------------------

loc_100014E9:                           ; CODE XREF: Dl
                push    104h            ; nSize
                lea     eax, [ebp+Filename]
                push    eax             ; lpFilename
                push    0               ; hModule
                call    ds:GetModuleFileNameA
```

Format of Assembly Code

OpCode

```
push    ebp
mov     ebp, esp
sub     esp, 0Ch
mov     [ebp-4], 1
mov     [ebp-8], 2
mov     eax, [ebp-4]
add     eax, [ebp-8]
mov     [ebp-0Ch], eax
mov     ecx, [ebp-0Ch]
push    ecx
push    0040C000h
call    00401034h
add     esp, 8
xor     eax, eax
mov     esp, ebp
pop     ebp
retn
```

Operand (arguments)

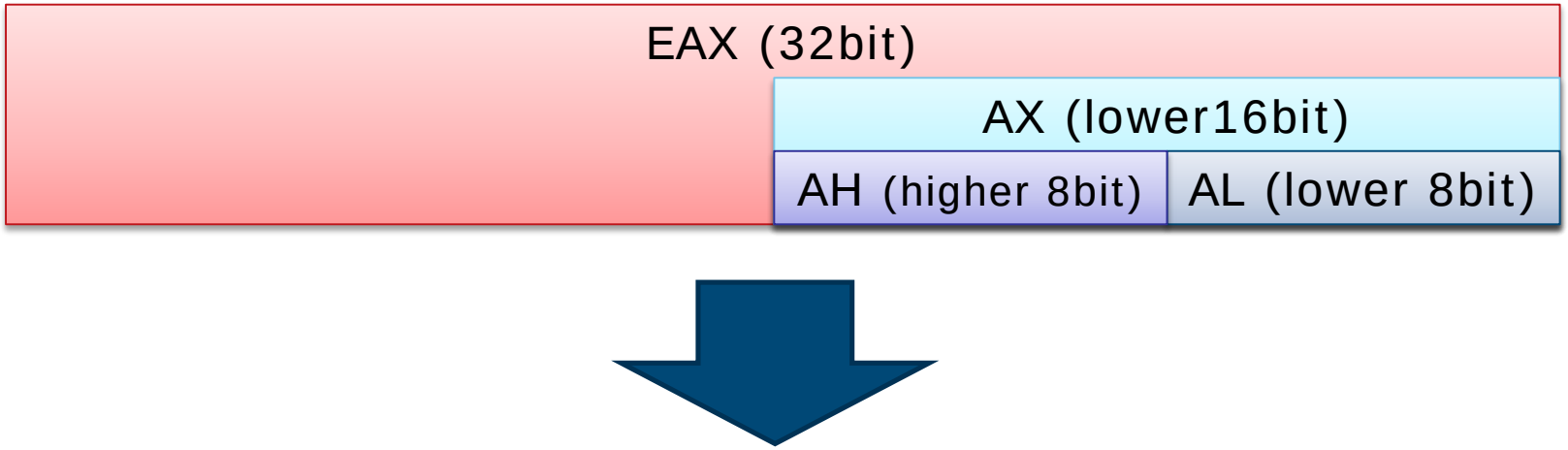
Register

- Memory inside CPU
  - Can use them as variables for calculations
  - Address that indicates next instruction (Program Counter)
  - Pointers related stack

Register name	Description
EAX, EBX, EDX	General purpose register
ECX	General purpose register especially used for counter
ESI, EDI	General purpose register especially used for "source" and "destination"
EIP (Instruction Pointer)	Address that indicates next instruction
ESP (Stack Pointer)	Current stack address
EBP (Base Pointer)	Bottom of stack for current function

Register Size

- Several registers' names are changed according to the data size
  - EAX, EBX, ECX...



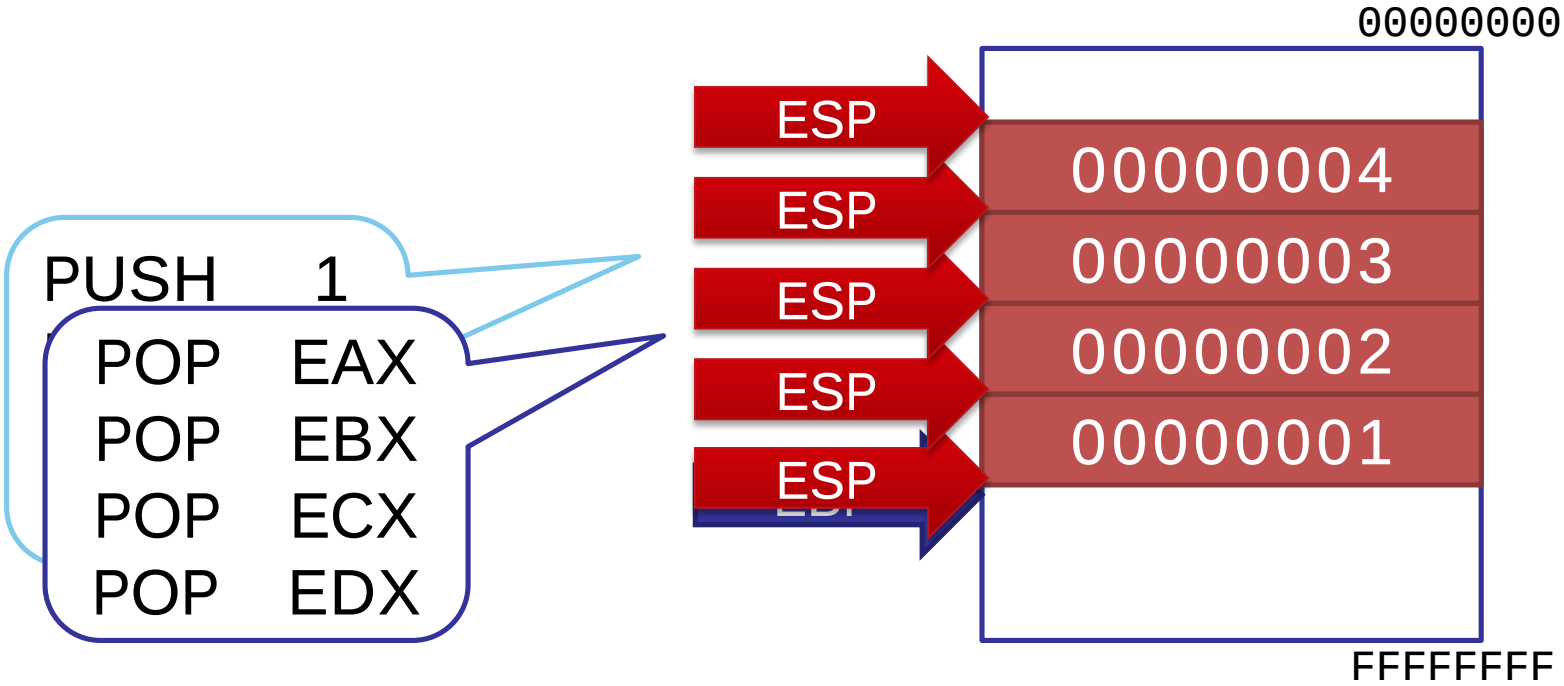
"mov eax, 0" ≠ "mov ax, 0"

Major Instructions

Assignment	mov	Copy value
	lea	Load address
Calculation	add & sub	+ / -
	and & or & xor & not	Logical operation
	inc & dec	++1 / --1
Jump	jmp	Jump to specified address
	jz, jnz, ja, ...	For branch on condition
	call	Call subroutine (function)

Stack

- Store temporary values to "stack" on memory
  - Due to the limits of registers
- Stack management
  - Use PUSH/POP
  - Stack related addresses are stored in EBP & ESP

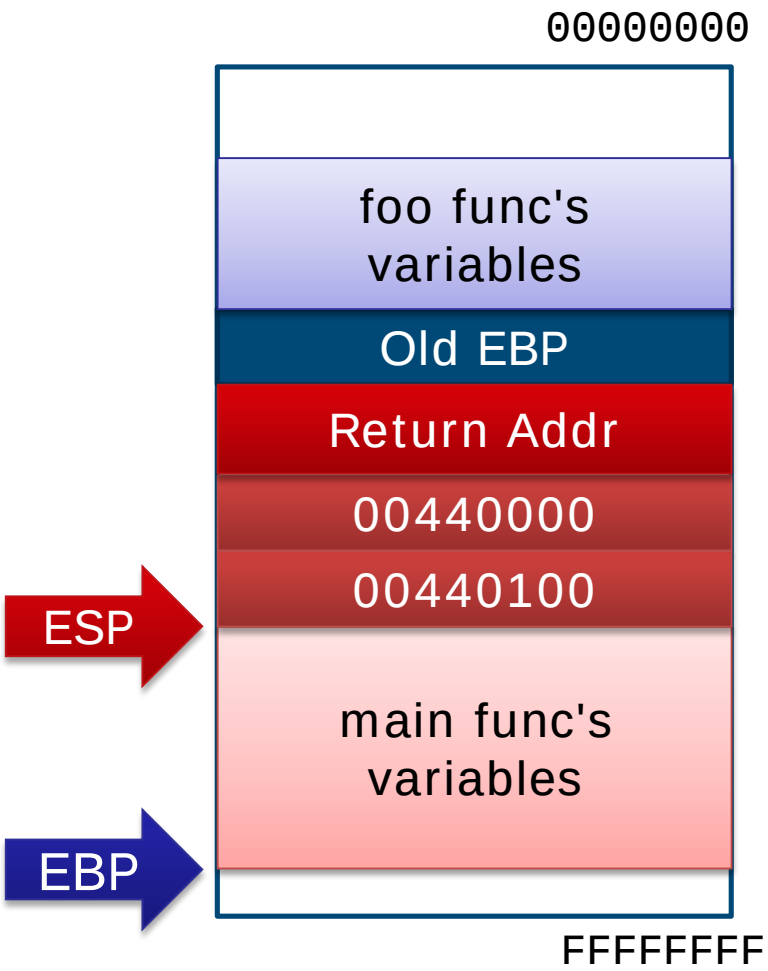


Function Call using Stack

- call = push + jmp
- retn = pop + jmp

```
main
  :
  push   440100h   ;arg1
  push   440000h   ;arg2
  call   foo
  mov    [ebp+result],eax
  :
```

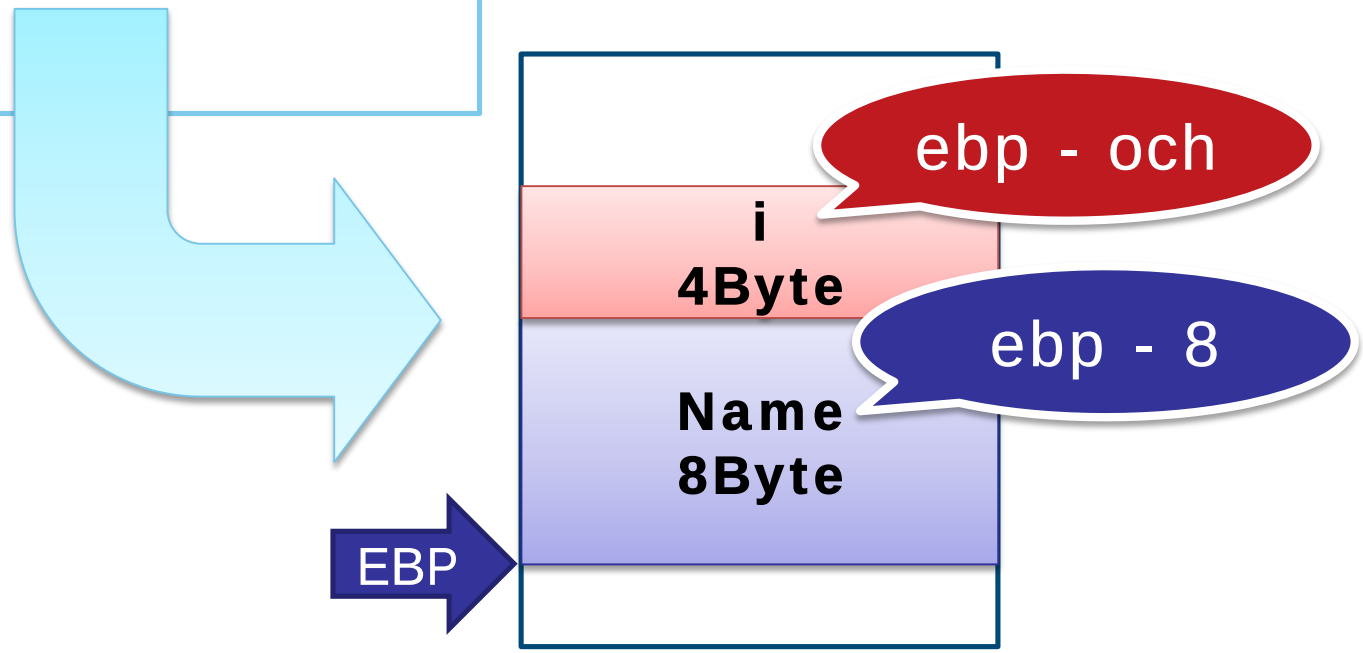
```
foo
  push   ebp
  mov    ebp, esp
  xor    eax, eax
  :
  mov    esp, ebp
  pop    ebp
  retn   10h
```



Local Variables

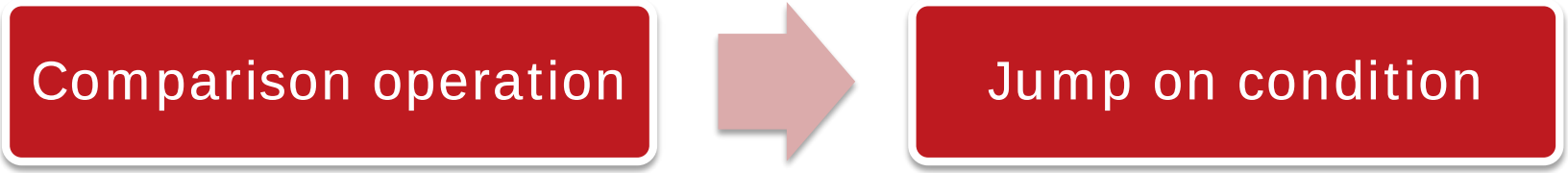
- Local variables are allocated on stack
  - Normally referred using offset from ebp

```
void func(){
    char Name[] = “abcdefg”;
    int i;
    :
}
```



Branch on Conditions

- Basic flow



1. Comparison operation
   - cmp a b
     - "sub a b" and discard result
   - test a b
     - "and a b" and discard result test
2. Jump on condition
   - jz
   - jnb
   - ja
   - etc.

Char	Description	
n	not	
z / e	zero/equal	Previous result is 0 (Both values are same)
a	above	Operand 1 is higher than operand 2
b	below	Operand 1 is smaller than operand 2

Exercise 1. Static Analysis Basic

i. Analyze the following function and explain what it does

```
sub_401000      proc near               ; CODE XREF: _main+8↓p

var_8           = dword ptr -8
arg_0           = dword ptr  8
arg_4           = dword ptr  0Ch

                push    ebp
                mov     ebp, esp
                sub     esp, 8
                mov     eax, [ebp+arg_0]
                add     eax, [ebp+arg_4]
                mov     [ebp+var_8], eax
                mov     eax, [ebp+var_8]
                mov     esp, ebp
                pop     ebp
                retn
sub_401000      endp
```

Exercise 1. Answer

i. Analyze the following function and explain what it does

```
sub_401000      proc near               ; CODE XREF: _main+8↓p

var_8           = dword ptr -8
arg_0           = dword ptr  8
arg_4           = dword

                push    ebp
                mov     ebp, esp
                sub     esp, 8
                mov     eax, [ebp+arg_0]
                add     eax, [ebp+arg_4]
                mov     [ebp+var_8], eax
                mov     eax, [ebp+var_8]
                mov     esp, ebp
                pop     ebp
                retn
sub_401000      endp
```

arg_0 + arg_4

Exercise 1. Static Analysis Basic

ii. Find “branch on condition” and explain each condition and corresponding result

```
_main           proc near               ; CODE XREF: ___tmainCRTStartup+15A↓p

var_4           = dword ptr -4

                push    ebp
                mov     ebp, esp
                push    ecx
                push    5
                push    4               ; int
                call    sub_401000
                add     esp, 8
                mov     [ebp+var_4], eax
                cmp     [ebp+var_4], 0
                jz      short loc_40104C
                mov     eax, [ebp+var_4]
                push    eax
                push    offset aAnswerIsD_ ; "Answer is %d.\n"
                call    _printf
                add     esp, 8
                jmp     short loc_401059
```

sub_401000 is "add_value" function

Exercise 1. Answer

ii. Find “branch on condition” and explain each condition and corresponding result

```
_main           proc near               ; CODE XREF: ___tmainCRTStartup+15A↓p

var_4           = dword ptr -4

                push    ebp
                mov     ebp, esp
                push    ecx
                push    5               ;
                push    4               ;
                call    sub_401000
                add     esp, 8
                mov     [ebp+var_4], eax
                cmp     [ebp+var_4], 0
                jz      short loc_40104C
                mov     eax, [ebp+var_4
                push    eax
                push    offset aAnswerI
                call    _printf
                add     esp, 8
                jmp     short loc_401059
```

stored result of "add_value" func

result is 0 or not?

Deep Understanding x86

- IA-32 Architectures Software Developer Manuals
  - http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html

MOV—Move

Opcode	Instruction	Op/En	64-Bit Mode	Compat/Leg Mode	Description
88 /r	MOV *r/m8,r8*	MR	Valid	Valid	Move *r8* to *r/m8*.
REX + 88 /r	MOV *r/m8****,*r8****	MR	Valid	N.E.	Move *r8* to *r/m8*.
89 /r	MOV *r/m16,r16*	MR	Valid	Valid	Move *r16* to *r/m16*.
89 /r	MOV *r/m32,r32*	MR	Valid	Valid	Move *r32* to *r/m32*.
REX.W + 89 /r	MOV *r/m64,r64*	MR	Valid	N.E.	Move *r64* to *r/m64*.
8A /r	MOV *r8,r/m8*	RM	Valid	Valid	Move *r/m8* to *r8*.
REX + 8A /r	MOV *r8***,r/m8****	RM	Valid	N.E.	Move *r/m8* to *r8*.
8B /r	MOV *r16,r/m16*	RM	Valid	Valid	Move *r/m16* to *r16*.
8B /r	MOV *r32,r/m32*	RM	Valid	Valid	Move *r/m32* to *r32*.
REX.W + 8B /r	MOV *r64,r/m64*	RM	Valid	N.E.	Move *r/m64* to *r64*.
8C /r	MOV *r/m16,Sreg***	MR	Valid	Valid	Move segment register to *r/m16*.
REX.W + 8C /r	MOV *r/m64,Sreg***	MR	Valid	Valid	Move zero extended 16-bit segment register

EFFICIENT CODE ANALYSIS

Understanding Source Code

- In which order should be read the following source code?

```
int send_data(){
    HANDLE hInternet, hConnect, hRequest;

    hInternet = InternetOpen(NULL, INTERNET_OPEN_TYPE_PRECONFIG,
                             NULL, NULL, 0);
    if(hInternet == NULL)
        return 1;
    hConnect = InternetConnect(hInternet, SERVER_NAME,
                               INTERNET_DEFAULT_HTTP_PORT,...
    if(hConnect == NULL){
        InternetCloseHandle(hInternet);
        return 2;
    }
    hRequest = HttpOpenRequest(hConnect, __T("GET"), NULL, NULL, ...
```

Reading Steps

1. Check Windows API

2. Check arguments

3. Check brunch on condition

```
lea     edx, [esp+4D0h+FileName]
push    edx                     ; lpBuffer
push    104h                    ; nBufferLength
call    ds:GetTempPathW
test    eax, eax
jnz     short loc_4010B1
```

Learning Windows API

- Use MSDN Library
  - http://msdn.microsoft.com/library

	Online	Offline
Cost	Free	MSDN Subscription needed to download
Features	• Always up to date • Need Internet connection • Can download necessary sections for offline use • Depends on connection speed	• Can be used in offline environment • Fast

Reading Arguments

- Assembly code

```
push    0           ;bFailIfExists
mov     eax,[ebp+C]
push    eax         ;lpNewFileName
mov     ecx,[ebp+8]
push    ecx         ;lpExistingFileName
call    ds:CopyFileA
```

- C++ syntax from MSDN Library

```
BOOL CopyFile(
  LPCTSTR lpExistingFileName,
  LPCTSTR lpNewFileName,
  BOOL bFailIfExists
);
```

Reverse order

- Return value is stored in EAX

Exercise 2. Efficient Code Analysis

- Read the following function efficiently and guess what it does

```
                and     [ebp+hKey], 0
                lea     eax, [ebp+hKey]
                push    eax             ; phkResult
                push    0F003Fh         ; samDesired
                push    0               ; ulOptions
                push    offset SubKey   ; "SOFTWARE\\Microsoft\\Windows\\CurrentVe"
                push    80000001h       ; hKey
                call    ds:RegOpenKeyExA
                test    eax, eax
                jz      short loc_401382
                xor     eax, eax
                jmp     short locret_4013BB
; ---------------------------------------------------------------------------
loc_401382:                             ; CODE XREF: sub_401357+25↑j
                push    105h            ; cbData
                push    offset Data     ; lpData
                push    1               ; dwType
                push    0               ; Reserved
                push    offset ValueName ; "TRAINING"
                push    [ebp+hKey]      ; hKey
                call    ds:RegSetValueExA
```

ersion¥¥Run

Exercise 2. Answer

- Read the following function efficiently and guess what it does

```
                and     [ebp+hKey], 0
                lea     eax, [ebp+hKey]
                push    eax
                push    0F003Fh
                push    0                       ; ulOptions
                push    offset SubKey           ; "SOFTWARE\\Microsoft\\Windows\\CurrentVe"
                push    80000001h               ; hKey
```

Target entry is "Run" key

Run

Register itself to be executed automatically after rebooting

```
; ---------------------------------------------------------------------------
loc_401382:                                     ; CODE XREF: sub_401357+25↑j
                push    105h                    ; cbData
                push    offset Data             ; lpData
                push    4                       ; dwType
                push    0                       ; Reserved
                push    offset ValueName        ; "TRAINING"
                push    [ebp+hKey]              ; hKey
                call    ds:RegSetValueExA
```

Write value in registry entry

USING IDA

(recap) Interactive DisAssembler

- http://www.hex-rays.com/idapro/
  - —3 versions are available

	Free 	Starter 	Pro 
Version	Ver. 5.0 Ver. 6.5 demo	Ver. 6.5	Ver. 6.5
Cost	Free	• 539USD/User • 819USD/Computer	• 1059USD/User • 1589USD/Computer
Features	• Old or limited	• Supports up to 20 processes	• Supports up to 50 processes • Can analyze files for 64 bit platforms

You Have to Talk with

Madame de Maintenon
(see Wikipedia)

Important Points for IDA

1. Make it right

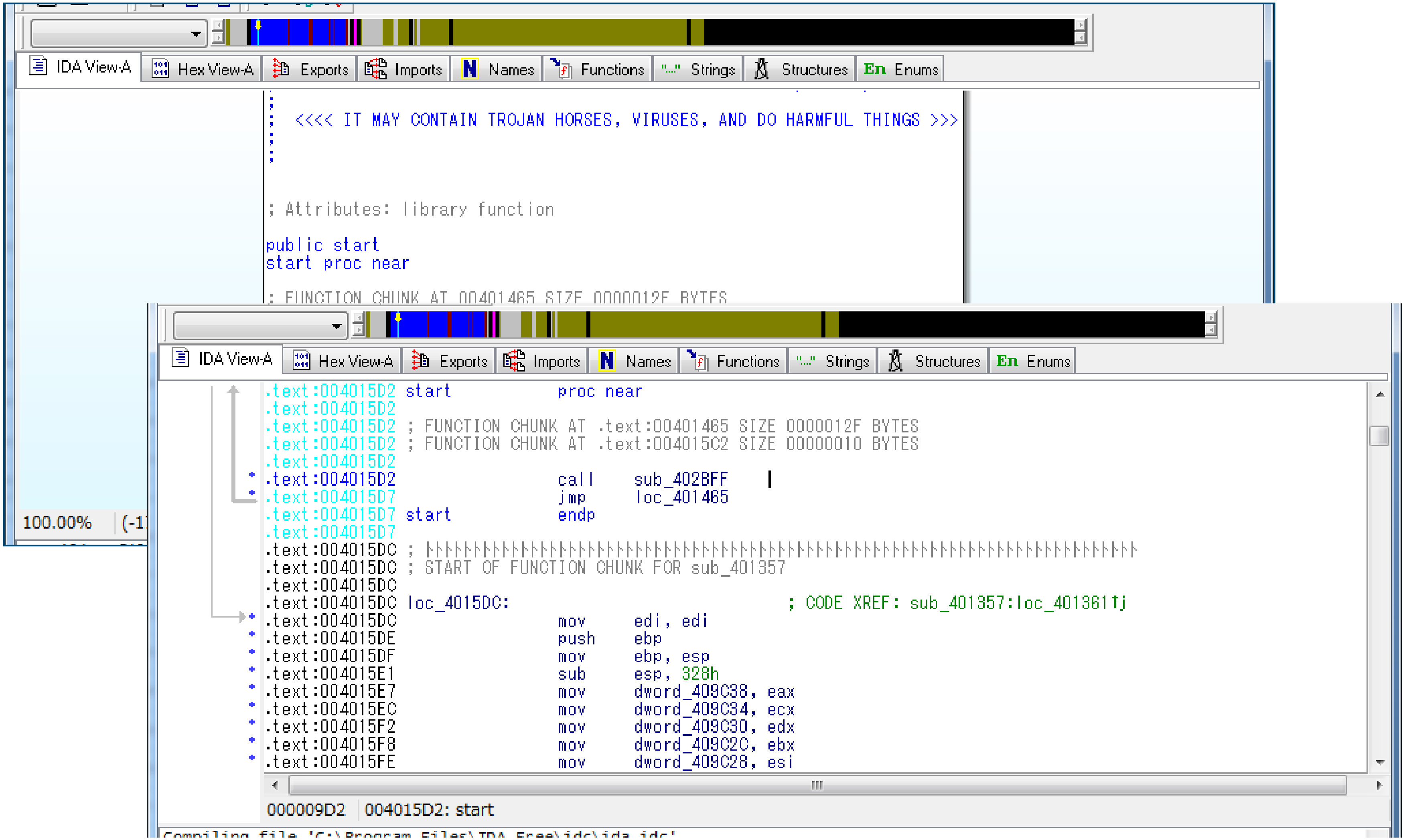
- **Instruction** or **data**?
- **Malicious function** or **library function**?

2. Use as a high functionality notepad

- Name analyzed function / variables
- Write your comments
- Put function type-declaration
- Change display format for easier reading
  - Hex / binary / ASCII / offset

Main Windows (IDA view)

- Graph view / Text view



```
;
;   <<<< IT MAY CONTAIN TROJAN HORSES, VIRUSES, AND DO HARMFUL THINGS >>>
;
;

; Attributes: library function

public start
start proc near

; FUNCTION CHUNK AT 00401465 SIZE 0000012F BYTES
```

```
.text:004015D2 start           proc near
.text:004015D2
.text:004015D2 ; FUNCTION CHUNK AT .text:00401465 SIZE 0000012F BYTES
.text:004015D2 ; FUNCTION CHUNK AT .text:004015C2 SIZE 00000010 BYTES
.text:004015D2
.text:004015D2                 call    sub_402BFF
.text:004015D7                 jmp     loc_401465
.text:004015D7 start           endp
.text:004015D7
.text:004015DC ; ├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├├
.text:004015DC ; START OF FUNCTION CHUNK FOR sub_401357
.text:004015DC
.text:004015DC loc_4015DC:                             ; CODE XREF: sub_401357:loc_401361↑j
.text:004015DC                 mov     edi, edi
.text:004015DE                 push    ebp
.text:004015DF                 mov     ebp, esp
.text:004015E1                 sub     esp, 328h
.text:004015E7                 mov     dword_409C38, eax
.text:004015EC                 mov     dword_409C34, ecx
.text:004015F2                 mov     dword_409C30, edx
.text:004015F8                 mov     dword_409C2C, ebx
.text:004015FE                 mov     dword_409C28, esi
```

```
000009D2  004015D2: start
```

```
Compiling file 'C:\Program Files\IDA Free\idc\ida.idc'
```

Main Windows (IDA view)

Reading Code

```
00000000 55 MYUNK   db  55h ; U                          ; Regular unknown name
00000001 02 MYDATA  db offset MYCODE+1                   ; Regular data name
00000001    MYCODE:                                      ; CODE XREF: MyFunc::MyClass(int,ulong
00000001 1D+                                             ; MyFunc::MyClass(int,ulong)+C↓p
00000001 2B+                                             ; DATA XREF: seg000:0000002C↓o
00000001 02+                                             ; seg000:00000000↑o
00000001 00+        sbb     eax, 555                     ; Regular code name
00000007 53+        unicode 0, <String constant>         ; Macro name & string data
00000025 9A+        call    PostMessage                  ; A repeatable comment
0000002C 55 unk_0_2B db 55h ; U                          ; Dummy unknown name
0000002D 01 HIDNAME db offset MYCODE                     ; Hidden name
0000002E 49 off_1   d                                    ; Dummy data name and libfunc name
0000002F 63         d                                    ; Character data constant
00000030 55         db 55h                               ; Numeric data constant
00000031    ; Below are an assembler directive and a segment nam
00000031            assume ds:seg000
00000031    ; =============== S U B R O U T I N E =======================================
00000031    ; MyFunc::MyClass(int,unsigned long)
00000031    ; A demangled name is above
00000031    @MyFunc@MyClass$qiul proc near
00000031    arg_4   = dword ptr  8
00000031 B0+        mov     al, 'U'                      ; Character constant in instruction
00000033 BB+        mov     ebx, 88h ; ' ; <void>        ; A void operand
00000038 E8+        call    MYCODE                       ; Call Procedure
0000003D E8+        call    near ptr MYCODE+1            ; Call Procedure
00000042 C7+        mov     [ebp+arg_4], ALTOP           ; Stack var and alternative operand
00000049    locret_2:                                    ; Dummy code name
```

Name

XREF

Calling API

Comments

Strings Window

- Like BinText

IDA - C:¥Users¥Administrator¥Desktop¥svn¥malware_analyst¥trunk¥sample_code¥runtime_sample_basic¥Release¥dropper.exe -...

File Edit Jump Search View Debugger Options Windows Help

IDA View-A | Hex View-A | Exports | Imports | Names | Functions | Strings | Structures | Enums

Address	Length	Type	String
HEADE...	00000027	C	!This program cannot be run in DOS mode
HEADE...	00000005	C	RichR
HEADE...	00000006	C	.text
HEADE...	00000008	C	`.rdata
HEADE...	00000007	C	@.data
HEADE...	00000006	C	.rsrc
HEADE...	00000008	C	@.reloc
.rdata:0...	0000000F	C	CorExitProcess
.rdata:0...	00000008	C	FlsFree
.rdata:0...	0000000C	C	FlsSetValue
.rdata:0...	0000000C	C	FlsGetValue
.rdata:0...	00000009	C	FlsAlloc
.rdata:0...	00000018	C	GetProcessWindowStation
.rdata:0...	0000001A	C	GetUserObjectInformationW
.rdata:0...	00000013	C	GetLastActivePopup
.rdata:0...	00000010	C	GetActiveWindow

Functions Window

- A list of functions

IDA - C:¥Users¥Administrator¥Desktop¥svn¥malware_analyst¥trunk¥sample_code¥runtime_sample_basic¥Release¥dropper.exe -...

File Edit Jump Search View Debugger Options Windows Help

IDA View-A | Hex View-A | Exports | Imports | Names | Functions | Strings | Structures | Enums

Function name	Segment	Start	Length	R	F	L	S	B	T	=
sub_401000	.text	00401000	00000202	R	.	.	.	B	.	.
sub_401210	.text	00401210	00000147	R	.	.	.	B	T	.
sub_401357	.text	00401357	0000000F	R	.	.	.	.	.	.
sub_401366	.text	00401366	000000D6	R	.	.	.	B	.	.
sub_40143C	.text	0040143C	00000029	R	.	.	.	B	.	.
sub_401594	.text	00401594	00000014	R	.	.	.	.	.	.
sub_4015A8	.text	004015A8	0000001A	R	.	.	.	.	.	.
start	.text	004015D2	0000000A	R	.	L	.	.	.	.
sub_4016E2	.text	004016E2	0000000F	R	.	.	.	B	.	.
sub_4016F1	.text	004016F1	00000129	R	.	.	.	B	.	.
sub_40181A	.text	0040181A	00000025	R	.	.	.	.	.	.
sub_40183F	.text	0040183F	0000002D	R	.	.	.	B	.	.
sub_40186C	.text	0040186C	00000010	R	.	.	.	.	.	.
sub_40187C	.text	0040187C	00000042	R	.	.	.	B	.	.
sub_4018BE	.text	004018BE	00000013	R	.	.	.	.	.	.
TopLevelExceptionFilter	.text	004018D1	00000042	R	.	.	.	B	T	.

Line 1 of 147

Imports Windows

- A list of APIs required by the target

IDA - C:¥Users¥Administrator¥Desktop¥svn¥malware_analyst¥trunk¥sample_code¥runtime_sample_basic¥Release¥dropper.exe -...

File Edit Jump Search View Debugger Options Windows Help

IDA View-A | Hex View-A | Exports | Imports | Names | Functions | Strings | Structures | Enums

Address	Ordinal	Name	Library
00406000		ExitProcess	KERNEL32
00406004		GetEnvironmentVariableW	KERNEL32
00406008		FindResourceW	KERNEL32
0040600C		LoadResource	KERNEL32
00406010		CreateProcessW	KERNEL32
00406014		HeapAlloc	KERNEL32
00406018		HeapFree	KERNEL32
0040601C		GetProcessHeap	KERNEL32
00406020		WriteFile	KERNEL32
00406024		LoadLibraryW	KERNEL32
00406028		SizeofResource	KERNEL32
0040602C		GetModuleFileNameW	KERNEL32
00406030		CreateFileW	KERNEL32
00406034		GetTempPathW	KERNEL32
00406038		GetLastError	KERNEL32
0040603C		GetProcAddress	KERNEL32

Line 1 of 65

Hex View Window

- Imports (list of APIs required by the target)

IDA - C:¥Users¥Administrator¥Desktop¥svn¥malware_analyst¥trunk¥sample_code¥runtime_sample_basic¥Release¥dropper.exe -...

File Edit Jump Search View Debugger Options Windows Help

IDA View-A | Hex View-A | Exports | Imports | Names | Functions | Strings | Structures | Enums

```
.text:00401590  00 00 EB 2E 8B 45 EC 8B  08 8B 09 89 4D DC 50 51  ..・畿・  ・窺ワPQ
.text:004015A0  E8 75 08 00 00 59 59 C3  8B 65 E8 8B 45 DC 89 45  鑾 ..YYテ菊闇Eワ右
.text:004015B0  E0 83 7D E4 00 75 06 50  E8 FD 05 00 00 E8 1D 06  煩}・u P・  ..・
.text:004015C0  00 00 C7 45 FC FE FF FF  FF 8B 45 E0 E8 84 14 00  ..ヌE・・・・畿琲
.text:004015D0  00 C3 E8 28 16 00 00 E9  89 FE FF FF 8B FF 55 8B  .テ・( ..驩・・・・
.text:004015E0  EC 81 EC 28 03 00 00 A3  38 9C 40 00 89 0D 34 9C  ・・  ..]8廖.・4・
.text:004015F0  40 00 89 15 30 9C 40 00  89 1D 2C 9C 40 00 89 35  @.・0廖.・,廖.・
.text:00401600  28 9C 40 00 89 3D 24 9C  40 00 66 8C 15 50 9C 40  (廖.・$廖.f・P廖
.text:00401610  00 66 8C 0D 44 9C 40 00  66 8C 1D 20 9C 40 00 66  .f・D廖.f・  廖.f
.text:00401620  8C 05 1C 9C 40 00 66 8C  25 18 9C 40 00 66 8C 2D  ・廖.f・  ↑廖.f・
.text:00401630  14 9C 40 00 9C 8F 05 48  9C 40 00 8B 45 00 A3 3C   廖.恂 H廖.畿.」<
.text:00401640  9C 40 00 8B 45 04 A3 40  9C 40 00 8D 45 08 A3 4C  廖.畿 」@廖.孔 」L
.text:00401650  9C 40 00 8B 85 E0 FC FF  FF C7 05 88 9B 40 00 01  廖.球珱・・ヌ ・@
.text:00401660  00 01 00 A1 40 9C 40 00  A3 3C 9B 40 00 C7 05 30  . ..@廖.」<奸.ヌ 0
.text:00401670  9B 40 00 09 04 00 C0 C7  05 34 9B 40 00 01 00 00  奸. .匁 4奸. ..
.text:00401680  00 A1 04 90 40 00 89 85  D8 FC FF FF A1 08 90 40  .。拭.怨リ・・。お
.text:00401690  00 89 85 DC FC FF FF FF  15 6C 60 40 00 A3 80 9B  .怨ワ・・・ |`@.」
.text:004016A0  40 00 6A 01 E8 F1 15 00  00 59 6A 00 FF 15 68 60  @.j 頌 ..Yj.・ h
.text:004016B0  40 00 68 40 61 40 00 FF  15 64 60 40 00 83 3D 80  @.h@a@.・ d`@.・
.text:004016C0  9B 40 00 00 75 08 6A 01  E8 CD 15 00 00 59 68 09  奸..u j 鬚 ..Yh
.text:004016D0  04 00 C0 FF 15 60 60 40  00 50 FF 15 5C 60 40 00   .タ・ ``@.P・ ¥`
.text:004016E0  C9 C3 8B FF 55 8B EC 8B  45 08 A3 54 9E 40 00 5D  ノテ・U駆畿 」T曄.]
```

000009D2 | 004015D2: start

Recommended Configuration

- IDA config files you can edit
  - C:¥Program Files (x86)¥IDA Free¥cfg¥
- You can create user settings file
  - idauser.cfg, idauserg.cfg

Config File	Name	Meaning	Recommended Value
ida.cfg / idauser.cfg	OPCODE_BYTES	Display binary data	8
	SHOW_SP	Display stack pointer	YES
	SHOW_XREFS	Display cross references	8
idagui.cfg / idauserg.cfg	DISPLAY_PATCH_SUBMENU	Display patch submenu	YES

Basic Instruction

- Move

Key assign	Description
G	Jump to address
Esc	Back

- Changing data type

U	Change selection to "Unknown"
C	Change selection to "Code"
D	Change selection to "Data" • Byte, Word, Double Word
P	Change selection to "Function"
A	Change selection to "ASCII string"
*****	Change selection to "Array"

Basic Instruction

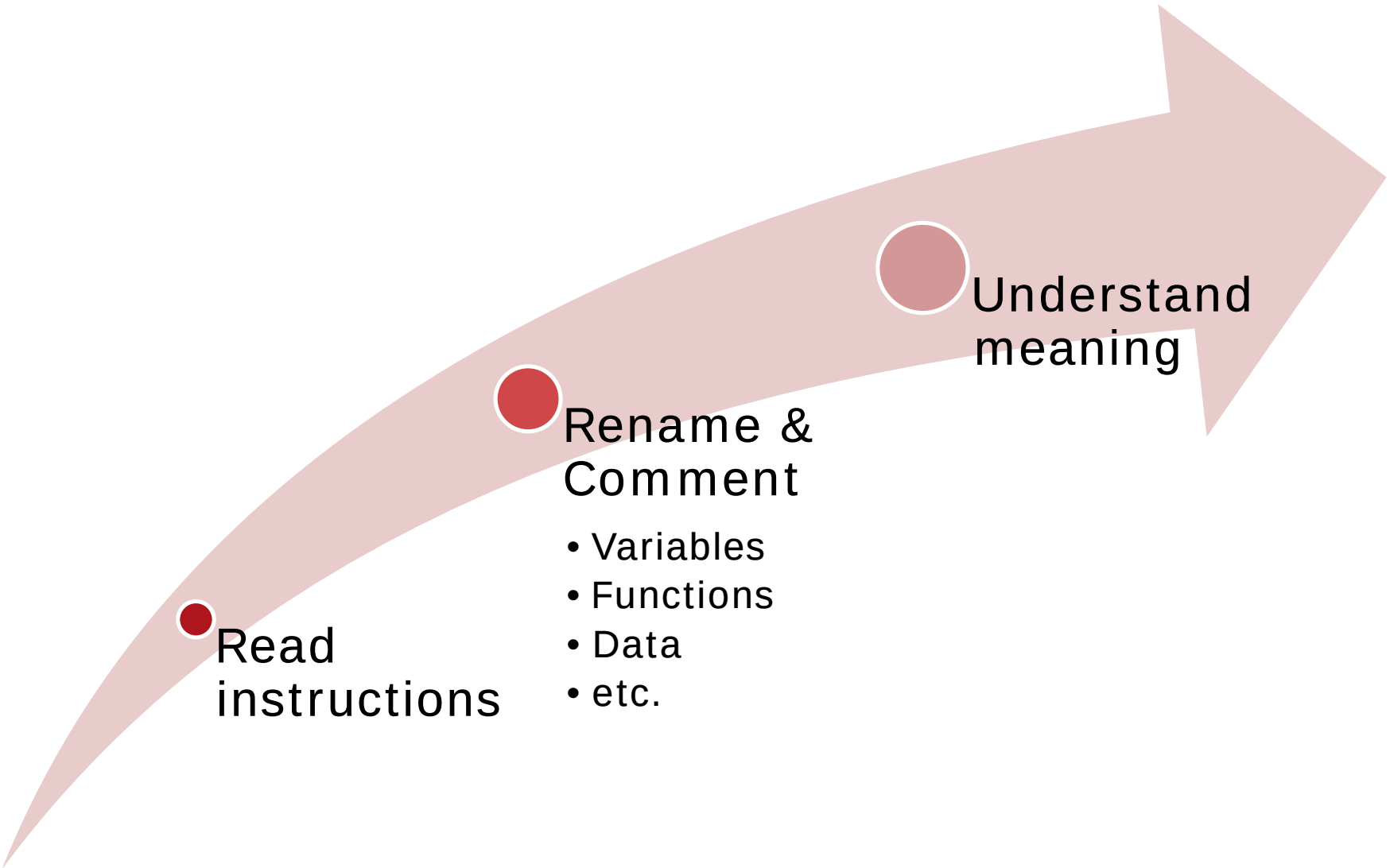
Note

Key assign	Description
N	Name function/variables/etc.
:	Insert comment
;	Insert repeatable comment
Y	Put type-declaration • void __cdecl Func(int num1, int num2);

Display format

H	Decimal <-> Hexadecimal
R	ASCII <-> Hexadecimal
Right-click -> Symbolic constant	Symbolic constant <-> Hexadecimal • ERROR_ALREADY_EXISTS • ACCESS_ALL • KEY_WRITE

Basic Analysis Process in IDA



Example of Renaming

- Exercise 1

```
sub_401000      proc near               ; COD

var_8           = dword ptr -8
arg_0           = dword ptr  8
arg_4           = dword ptr  0Ch

                push    ebp
                mov     ebp, esp
                sub     esp, 8
                mov     eax, [ebp+arg_
                add     eax, [ebp+arg_
                mov     [ebp+var_8], e
                mov     eax, [ebp+var_
                mov     esp, ebp
                pop     e
                retn
sub_401000      endp
```

```
; int __cdecl add_func(int arg1, int arg2)
add_func        proc near               ; COD

answer          = dword ptr -8
arg1            = dword ptr  8
arg2            = dword ptr  0Ch

                push    ebp
                mov     ebp, esp
                sub     esp, 8
                mov     eax, [ebp+arg1]
                add     eax, [ebp+arg2]
                mov     [ebp+answer], eax
                mov     eax, [ebp+answer]
                mov     esp, ebp
                pop     ebp
                retn
add_func        endp
```

Example of Analysis

- Analyzed IDB sample

```
push    offset rclsid   ; rclsid
call    ds:CoCreateInstance
mov     esi, eax
cmp     esi, edi
jl      loc_10016896
mov     eax, [ebp+ppv]
mov     ecx, [eax]
lea     edx, [ebp+pProxy]
push    edx
push    edi
push    edi
push    edi
push    edi
push    edi
push    edi
push    offset aRootCimv2 ; "ROOT\\CIMV2"
push    eax
mov     eax, [ecx+0Ch]
call    eax
mov     esi, eax
cmp     esi, edi
jl      loc_10016896
mov     ecx, [ebp+pProxy]
push    edi             ; dwCapabilities
push    edi             ; pAuthInfo
push    3               ; dwImpLevel
push    3               ; dwAuthnLevel
push    edi             ; pServerPrincName
push    edi             ; dwAuthzSvc
push    0Ah             ; dwAuthnSvc
push    ecx             ; pProxy
call    ds:CoSetProxyBlanket
```

```
push    offset IID_WbemLocator ; rclsid
call    ds:CoCreateInstance
mov     esi, eax
cmp     esi, edi
jl      loc_10016896
mov     eax, [ebp+ppv]
mov     ecx, [eax+IWbemLocator.lpVtbl]
lea     edx, [ebp+pProxy]
push    edx             ; ppNamespace
push    edi             ; pCtx
push    edi             ; strAuthority
push    edi             ; ISecurityFlags
push    edi             ; strLocale
push    edi             ; strPassword
push    edi             ; strUser
push    offset aRootCimv2 ; "ROOT\\CIMV2"
push    eax             ; strNetworkResource
mov     eax, [ecx+IWbemLocatorVtbl.ConnectServer]
call    eax             ; IWbemLocatorVtbl.ConnectServer
mov     esi, eax
cmp     esi, edi
jl      loc_10016896
mov     ecx, [ebp+pProxy]
push    edi             ; dwCapabilities
push    edi             ; pAuthInfo
push    RPC_C_IMP_LEVEL_IMPERSONATE ; dwImpLevel
push    RPC_C_AUTHN_LEVEL_CALL ; dwAuthnLevel
push    edi             ; pServerPrincName
push    edi             ; dwAuthzSvc
push    RPC_C_AUTHN_WINNT ; dwAuthnSvc
push    ecx             ; pProxy
call    ds:CoSetProxyBlanket
```

Exercise 3. Using IDA

i. Analyze the following functions in "static_sample3.idb" and rename functions/variables or insert your comments

- sub_4012DD
- sub_401303
- sub_401357

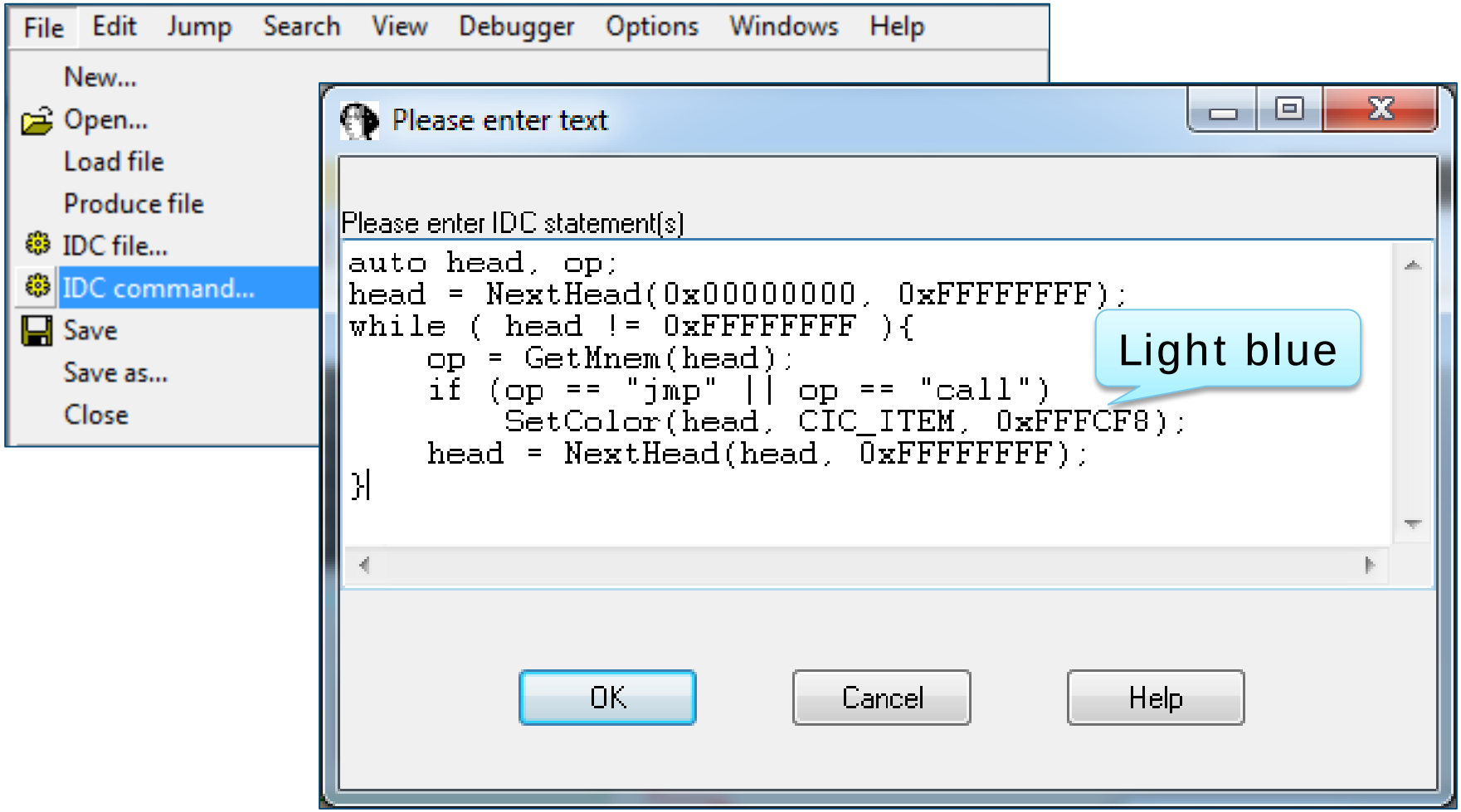
Exercise 3. Answer

i. Analyze the following functions in "static_sample3.idb" and rename functions/variables or insert your comments

- sub_4012DD
- sub_401303
- sub_401357
- **See static_sample3_ans.idb**

FYI: IDC Scripting

- If you want to change background color on "jmp" & "call" instructions



Questions?