

# Application Security

Tom Chothia  
Computer Security, Lecture 16

# Introduction

- This lecture takes a look at some of the protections put on software.
- It is almost always the case that this can be removed or “cracked”.
- Good protection tends to slow down this process, not stop it.

# Java Byte Code

- Java compiles to *Java Byte Code*.
  - Type: “javap -c <ClassName>” to see the byte code.
- Every computer must have its own Java Virtual Machine (JVM) which runs the byte code.
- Every different OS must have it's own JVM

Java Program  
.java

Compile Java to Byte  
Code Using "javac"

Java Byte Code  
.class

Run Byte Code On  
JVM using "java"

Windows  
JVM  
Windows  
Computer

Linux  
JVM  
Linux  
Computer

Phone  
JVM  
Mobile  
Phone

# Java Byte Code

- `iconst_0` : push 0 onto the stack
- `istore_1`: pop the top of the stack as variable 1
- `goto`: jump to line:
- `iload_1`: push variable 1 onto the stack
- `bipush`, `ldc`: push value onto stack
- `if_icmplt`: if 1<sup>st</sup> item on stack > 2<sup>nd</sup> jump to line
- `lfeq`: if 1<sup>st</sup> item on stack > 2<sup>nd</sup> jump to line

# Decompilation

- Wouldn't it be much easier to work with the source code, rather than the byte code?
- JD-GUI is a Java de-compiler, it transforms Java Byte Code into Java Code.
- Not perfect, e.g. confuses 0,1 and true,false.

# Bypassing the password check.

- De-compilation makes it much easier to understand what a program is doing.
- It also makes it easy to alter and recompile the code.
- All code that is used to protect the code can be removed.

C program

gcc on windows

gcc on linux

gcc on Mac

Window  
Assembly  
Windows  
Computer

Linux  
Assembly  
Linux  
Computer

Mac  
Assembly  
Mac  
Computer

# Binaries

- Binaries are written in assembly
- Much lower level than Java byte code,
- Assembly compiled for one type of machine won't run on another.
- But the same techniques apply.

# Hexedit

- Hex editors allow you to view and edit the binary.
- Any strings will be easily visible, and can be changed.
- Commands can also be changed.
- Once shipped, its impossible to stop someone viewing and editing the binary.

## But ...

- It is possible to make it very difficult to crack a piece of software, e.g.:
- Dynamically construct the key

# Ollydbg

- Ollydbg is an x86 debugger,
- It runs window's binaries and lets you monitor the state of the registers.
- Useful techniques include searching for strings and setting break points.

# Some x86 Commands

PUSH: add to top of stack

POP: read and remove from top of stack

CALL: execute a function

JMP: jump to some code (like writing to EIP)

RET, RETN, RETF: end a function and restart calling code.

MOV: move value between registers  
MOV r1,r2 = PUSH r1  
POP r2

# Jumps

To jump in x86 you first compare the values and then jump.

TEST: does a bitwise “and”.

CMP: subtracts 2 values

The result isn't stored but flags are set.

Following TEST:

JZ: jump if result was 0

JNZ: jump if result isn't zero

JE: jump if equal

JNE: jump if not equal

JL: jump if less than.

CPU - main thread, module ntdll

```

77970082 . E8 85760B00 CALL ntdll.77A2770C
77970087 . 85C0 TEST EAX,EAX
77970089 . 75 04 JNZ SHORT ntdll.7797008F
7797008B . 5B POP EBX
7797008C . C2 1000 RETN 10
7797008F > FF33 PUSH DWORD PTR DS:[EBX]
77970091 . 53 PUSH EBX
77970092 . 68 A0009777 PUSH ntdll.779700A0
77970097 . FF7424 18 PUSH DWORD PTR SS:[ESP+18]
7797009B . E8 A96C0300 CALL ntdll.RtlUnwind
779700A0 > 5B POP EBX
779700A1 . 50 PUSH EAX
779700A2 . 6A 00 PUSH 0
779700A4 . 6A 00 PUSH 0
779700A6 . E8 BDF70000 CALL ntdll.ZwCallbackReturn
779700AB . 8BF8 MOV EDI,EAX
779700AD > 57 PUSH EDI
779700AE . E8 026E0300 CALL ntdll.RtlRaiseStatus
779700B3 . EB F8 JMP SHORT ntdll.779700AD
779700B5 . 8D49 00 LEA ECX,DWORD PTR DS:[ECX]
779700B8 $ 64:8B00 000000 MOV ECX,DWORD PTR FS:[0]
779700BF . BA 70009777 MOV EDX,ntdll.77970070
779700C4 . 8D4424 10 LEA EAX,DWORD PTR SS:[ESP+10]
779700C8 . 894C24 10 MOV DWORD PTR SS:[ESP+10],ECX
779700CC . 895424 14 MOV DWORD PTR SS:[ESP+14],EDX
779700D0 . 64:A3 00000000 MOV DWORD PTR FS:[0],EAX
779700D6 . 83C4 04 ADD ESP,4
779700D9 . 5A POP EDX
779700DA . 64:A1 30000000 MOV EAX,DWORD PTR FS:[30]
779700E0 . 8B40 2C MOV EAX,DWORD PTR DS:[EAX+2C]
779700E3 . FF1490 CALL DWORD PTR DS:[EAX+EDX*4]
779700E6 . 50 PUSH EAX
779700E7 . 6A 00 PUSH 0
779700E9 . 6A 00 PUSH 0
779700EB . E8 78F70000 CALL ntdll.ZwCallbackReturn
779700F0 . 8BF0 MOV ESI,EAX
779700F2 > 56 PUSH ESI
779700F3 . E8 BD6D0300 CALL ntdll.RtlRaiseStatus
779700F8 . EB F8 JMP SHORT ntdll.779700F2
779700FA . C2 0C00 RETN 0C
779700FD . 8D49 00 LEA ECX,DWORD PTR DS:[ECX]
77970100 $ FC CLD
77970101 . 8B4C24 04 MOV ECX,DWORD PTR SS:[ESP+4]
77970105 . 8B1C24 MOV EBX,DWORD PTR SS:[ESP]
77970108 . 51 PUSH ECX
77970109 . 53 PUSH EBX
    
```

```

_eax_value
pExoptRec
ReturnAddr = nt
pRegistrationFr
RtlUnwind
    
```

Registers (FPU)	
EAX	C0000034
ECX	00000000
EDX	00000000
EBX	00000001
ESP	0018F9B0
EBP	0018F9E8
ESI	00525034
EDI	0000000C
EIP	7797F9CD ntdll.7797F9CD
C 0	ES 002B 32bit 0(FFFFFFFF)
P 0	CS 0023 32bit 0(FFFFFFFF)
A 1	SS 002B 32bit 0(FFFFFFFF)
Z 0	DS 002B 32bit 0(FFFFFFFF)
S 0	FS 0053 32bit 7EFD0000(FFF)
T 0	GS 002B 32bit 0(FFFFFFFF)
D 0	
O 0	LastErr ERROR_ACCESS_DENIED (00000005)
EFL	00000212 (NO,NB,NE,A,NS,PO,GE,G)
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

K Call stack of main thread

Address	Stack	Procedure / arguments	Called from
0018F9B0	779C34C0	ntdll.ZwOpenKey	ntdll.779C34BB
0018F9EC	779C343B	ntdll.779C3444	ntdll.LdrOpenImageFileOptionsKey
0018FA00	779E70AB	ntdll.LdrOpenImageFileOptionsKey	ntdll.779E70A6
0018FA04	00525034	Arg1 = 00525034	
0018FA08	00000000	Arg2 = 00000000	
0018FA0C	0018FA34	Arg3 = 0018FA34	
0018FA18	779AC160	ntdll.LdrQueryImageFileExecutionOptions	ntdll.LdrQueryImageFileExecutionOptions
0018FA3C	779D07D0	ntdll.LdrQueryImageFileExecutionOptions	ntdll.779D07D8
0018FA40	00525034	Arg1 = 00525034	
0018FA44	7798400C	Arg2 = 7798400C	
0018FA48	00000004	Arg3 = 00000004	
0018FA4C	0018FAF8	Arg4 = 0018FAF8	
0018FA50	00000004	Arg5 = 00000004	
0018FA54	00000000	Arg6 = 00000000	
0018FB38	779B37EE	? ntdll.7799DC25	ntdll.779B37E9

Address	Hex dump	ASCII
00405000	00 00 00 00 00 00 00 00	.....
00405008	00 00 00 00 00 00 00 00	.....

# Common Techniques

- Look for strings in hex.
- Identify key tests and check the values in the register using a debugger.
- Swap JEQ and JNEQ.
- Jump over the instructions that perform checks.

# Defenses

- Dynamically construct the key
  - Attacker can run code.
- Encrypt the binary,
  - Your program must include the key in plain text, so the attack can find it.
- Obfuscate the code
  - Can slow down attacks by months or years! (e.g. Skype).

# Defense

- Require online activation.
  - Activation can be completely disabled, users don't like this.
- Require online content, e.g. WoW, BlueRay
- Hardware based protection.
  - ... see Lecture 21.

# Summary

- Machine code can be inspected and edited.
- Many tools exist to inspect, debug and decompile code.
- Most software protection can be removed.
- But slowing this down by months or years can save a business.

# Exercise 3: Low Level Attacks

- Use Metasploit to gain access to a machine.
- Analyse Java Byte Code
- Analyse Window Binaries
- Due on Friday Dec 2<sup>nd</sup>.

# Next Lecture

- The most common threats to computer networks.
- And the most common defenses.