

Application Security

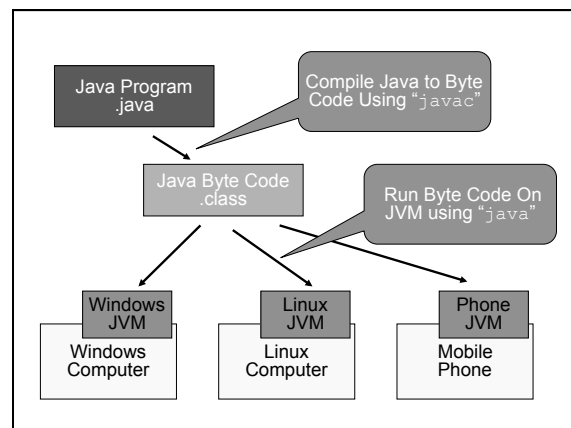
Tom Chothia
Computer Security, Lecture 16

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 its own JVM



Java Byte Code

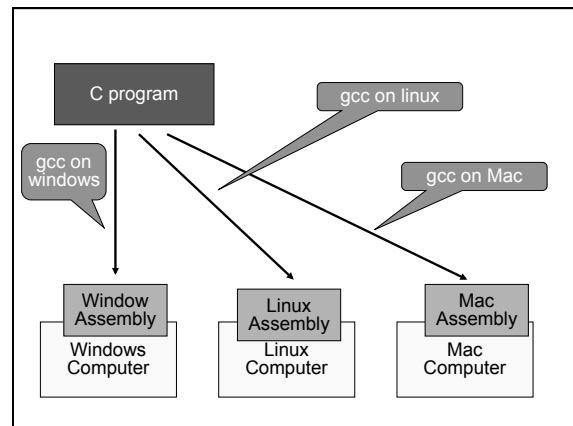
- `iconst_0` : push 0 onto the stack
- `istore_1`: pop the top of the stack as variable 1
- `goto`: jump to line:
- `iload_1`: push variable 1 onto the stack
- `bipush, ldc`: push value onto stack
- `if_icmplt`: if 1st item on stack > 2nd jump to line
- `lfeq`: if 1st item on stack > 2nd jump to line

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.



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

Olllydbg

- Olllydbg 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.

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 2nd.

Next Lecture

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