The Maginot License:

Failed Approaches to Licensing Java Software Over the Internet

Author: Mark D. LaDue
Reviewer: Guanglun Yu (George)

Presentation Outline

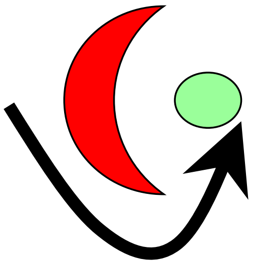
- Introduction
- Tampering Techniques
- Example 1: Hard Coding of License Data
- Example 2: Useless Encription
- Conclusion
- Questions

Introduction

Distributing software over the internet on a try-before-you-buy basis is quite popular today and such a way for Java software is named the "Maginot License".

A user will hold the whole software but without full functionality unless he makes the payment and gets the license to unlock the limitation at his side by himself.

Distributing Java software in that way is unsuccessful.



Introduction

Java programming language provides "unprecedented opportunities for who wish to purloin intellectual property".

The Java's OO brings programs into individual logic units(classes) which are easy to handle.

Its Class File Format defines a class file which contains all the original source code symbols, fields, methods,.. information which makes decompiling a Java class much easier.

Introduction

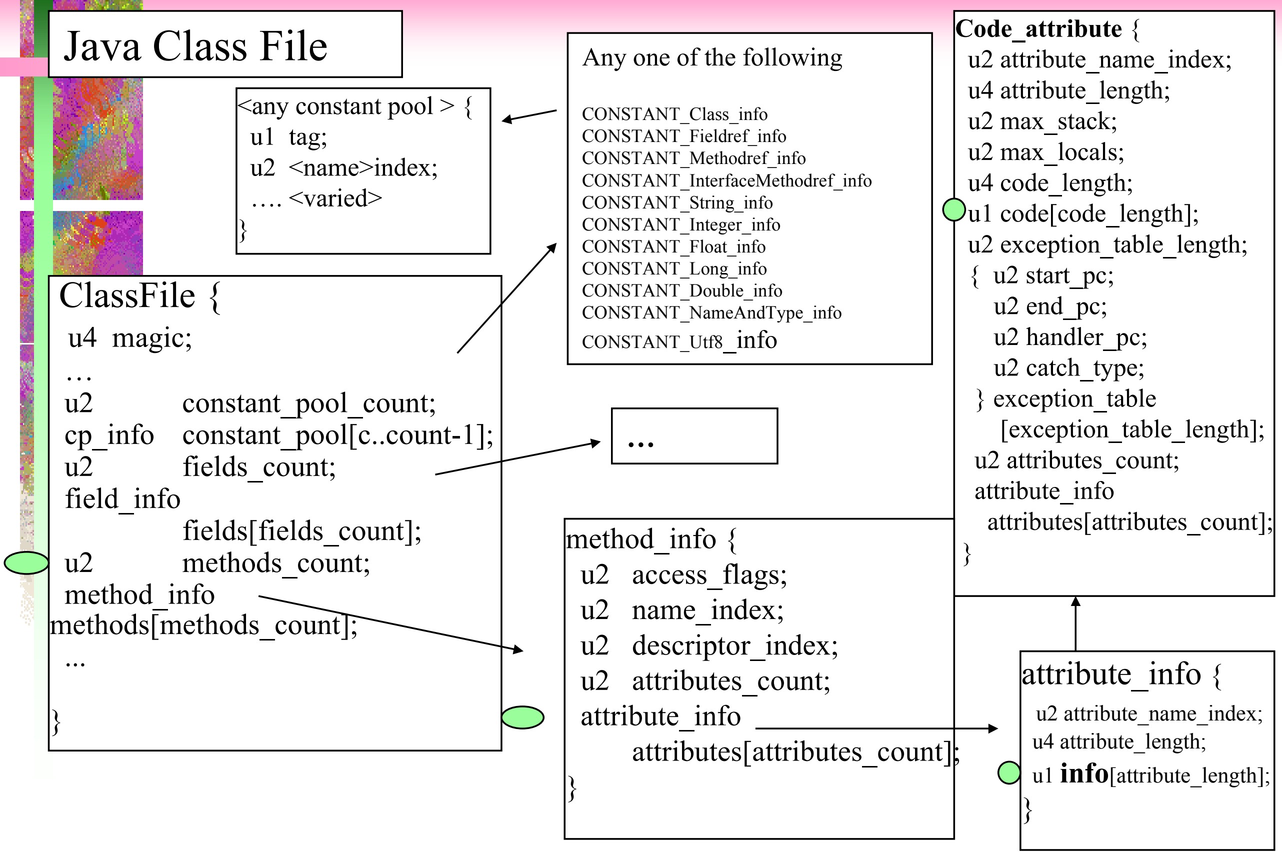
In this article, the author picked up some Java softwares which are type of "Maginot license" and then broke up these Maginot licenses.

I will explain the techniques Dr. LaDue used and present two examples to illustrate how he did them:

Hard Coding of Important Data (WingDis)

Useless Protection(JTimer).

Java Class File



```
ClassFile {
 u4  magic;
 …
 u2        constant_pool_count;
 cp_info   constant_pool[c..count-1];
 u2        fields_count;
 field_info
           fields[fields_count];
 u2        methods_count;
 method_info
methods[methods_count];
 …

}
```

```
<any constant pool > {
  u1  tag;
  u2  <name>index;
  …. <varied>
}
```

```
Any one of the following

CONSTANT_Class_info
CONSTANT_Fieldref_info
CONSTANT_Methodref_info
CONSTANT_InterfaceMethodref_info
CONSTANT_String_info
CONSTANT_Integer_info
CONSTANT_Float_info
CONSTANT_Long_info
CONSTANT_Double_info
CONSTANT_NameAndType_info
CONSTANT_Utf8_info
```

```
…
```

```
method_info {
  u2   access_flags;
  u2   name_index;
  u2   descriptor_index;
  u2   attributes_count;
  attribute_info
       attributes[attributes_count];
}
```

```
Code_attribute {
  u2 attribute_name_index;
  u4 attribute_length;
  u2 max_stack;
  u2 max_locals;
  u4 code_length;
  u1 code[code_length];
  u2 exception_table_length;
  {  u2 start_pc;
     u2 end_pc;
     u2 handler_pc;
     u2 catch_type;
  } exception_table
      [exception_table_length];
  u2 attributes_count;
  attribute_info
    attributes[attributes_count];
}
```

```
attribute_info {
  u2 attribute_name_index;
  u4 attribute_length;
  u1 info[attribute_length];
}
```

Tampering Techniques

Tools

Decompilers: Sun’s javap, Mocha

Assistant tools: Inspector(for getting method offset)
BotI, ItoB(hex editors).

Step 1: From various outside information which the software presents, locate possible classes which we need to examine.

Step 2: Use decompiler to examine the located classes and find the positions (index/line number from the output of the decompiler) of the bytecodes which relate to the outside information that we have already gained from the behavior of the software.

Tampering Techniques

Step 3: The javap, mocha give you the inner offset in one method of the bytecode which we have already located at previous step.

Step 4: To get, in that class file, exact position index of that bytecode which we want to change, use the output of Inspector which gives us the method’s starting offset.

Step 5: Then add the above two offsets to get the absolute offset in that class file for that bytecode and change it(normally to goto instruction).

Step 6: Update the original program with tampered classes.

The above are general for the tampering, but they may contain some difference in each individual case.

Hard Coding of License Data

Some companies do hard-code the licensing data and restrictions in software which in Java, would be the class files

Finjan Software's SurfinShield gives a user the evaluation of 30 days license.

But they hard-coded the user's installation date as the license starting date into their program when the user installs SurfinShield.

LaDue found the fatal class is SFped which is much simpler enough to find the clue of the weakness of the licensing by using javap.

Hard Coding of License Data

Output of Decompiled SFped.class by javap:

```
public class SFped extends
java.lang.Object {
    static final int year;
    static final int month;
    static final int day;
    public java.util.Date ped;
    public SFped();
}
Method SFped()
 0 aload_0
   1 invokespecial #9 <Method
java.lang.Object()>
   4 aload_0
   5 new #6 <Class java.util.Date>
   8 dup
   9 ldc #3 <Integer 97>
  11 ldc #2 <Integer 3>
  13 ldc #1 <Integer 15>
  15 invokespecial #8 <Method
java.util.Date(int,int,int)>
  18 putfield #7 <Field java.util.Date
ped>
  21 return
```

Hard Coding of License Data

The new SFped class source code rewritten by the author

```
public class SFped {
   public Date ped;
   public SFped() {
      ped = new Date();
      ped.setDate(ped.getDate()    - 1);
      }
}      // use this SFped, …
```

Compile it and update the SurfinShield with this new SFped class, then every time when you start SurfinShield, it always says 29 days left.

Useless Encription

Some software companies claimed that their product solved the Java software's licensing problem by providing encription or obfuscation.

InetSoft's JTimer is a good example of encription type. As it provides a complicated process:

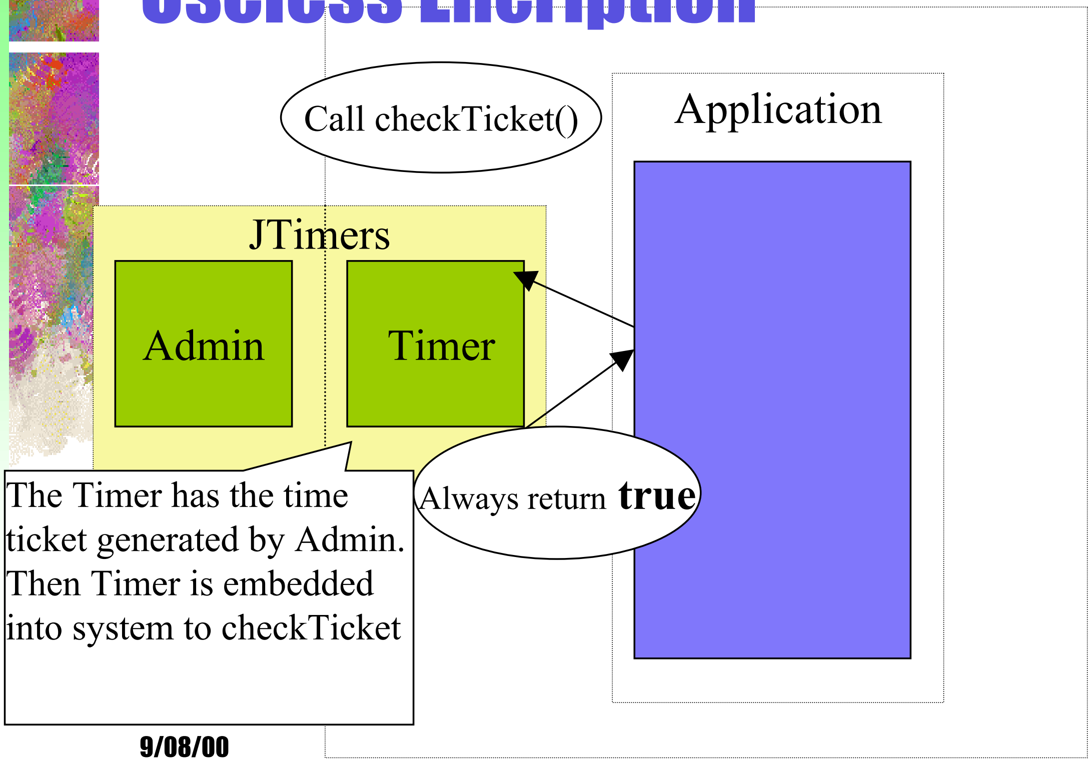
Admin ---> pub/private key pair, vendor ID;

(vendor)privatekey,vendor ID,expiryDate-> (feed)Admin --> (generate) Timer class object with a time ticket;

(Application) includes-->that Timer class -->(when run)

checkTicket() -->return true or false to application

Useless Encription



The Timer has the time ticket generated by Admin. Then Timer is embedded into system to checkTicket

Conclusion

The first example shows there is a serious problem for distributing Java software over the internet on a try-before-you-buy basis; the second shows there are no simple solutions; **perhaps no solution at all**.

Code obfuscation "does nothing to thwart the disassembly of Java class files." The class file format allows free and easy disassembly of classes by anyone who cares to inspect and tamper with them.

"Including encryption to protect Java application is equally futile."

Questions

How did Dr. LaDue realize that the license starting date was hard coded into SFped class?

Instead of the two passive solutions the author mentioned in this article, the toy version and the traditional shareware concept, is there any other better one?