Research Group

- Sowmya Krishnaswamy, Minseok Kwon, Di Ma, Quiyun Shao, Yi Zhang.
- Work supported by an NSF CAREER award and by Lilly Endowment Inc.
Secret Keys

\[ \text{encrypt}_k(m) \]

\[ \text{watermark}_k(P,W) \]
Overview

- ≥ 4 U.S. patents: software watermarking
- Collberg and Thomborsen (1999): embed a watermark in data structures.
- Our contribution: defense against "open-source" attacks.
- We have an implementation for Java: efficient, moderate overhead.
The Need to Prove Software Ownership

- Alice copyrights her software and sells it for profit.
- Bob manages to make a pirate copy.
- Bob wants the software because:
  - private use
  - industrial e-spionage
  - further selling for his own profit.
The Need to Prove Software Ownership

- **Question:** how does Alice protect her copyright?
- **Answer:** She must be able to prove ownership, possibly in a court of law, of a given copy of the software.
Approaches to Anti-piracy

- Keep a certified list of customers
- Link the software to the hardware.
- Link the software to a movable piece of hardware that cannot easily be copied
- **Software watermarking**: embed a secret into the software which can be retrieved on demand.
Attacks on Software Watermarks

- Locate, distort, remove the watermark.
- **Transformation** attacks: compilation, optimization, obfuscation, decompilation, dead-code removal.
- Defense: embed the watermark in data structures, not the structure of the program.
A comment:
/* My software, version 1.0 */

A data string:
printf("My software, version 1.0");

A particular ordering of the instructions, e.g., the ordering of the branches of an n-branch switch-statement.
Attacks on Software Watermarks

- **Subtractive** attacks: Bob attempts to locate and remove the watermark.
- **Additive** attacks: Bob inserts his own watermark to make it plausible that his watermark came before Alice’s.
- **Collusion** attacks: two attackers have two copies of the same watermarked program -- with different watermarks.
Open-Source Attack on Software Watermarks

- The attacker has access to the software for embedding and extracting watermarks.
- **Brute-force**: run embedding + extraction to learn how to locate a watermark.
Secret Keys

\[ \text{encrypt}_{k}(m) \]

\[ \text{watermark}_{k}(P, W) \]
Watermarks as Graphs

Each graph **encodes** a number (the watermark,) which can contain the serial number, customer number, date, etc.
The Design of our Watermarking System

original data structures

built by

original code

obfuscated by

a watermark PPCT

built by

code that builds wm PPCT
The Design of our Watermarking System

- **W**: the watermark
- **P**: the Java 1.2 program
- Three steps to watermark(P,W):
  1. generate(W)
  2. merge(P,generate(W))
  3. obfuscate the outcome of (2).
## Benchmark Programs

<table>
<thead>
<tr>
<th>Program</th>
<th>Description</th>
<th>Test input</th>
</tr>
</thead>
<tbody>
<tr>
<td>javac</td>
<td>a compiler for Java</td>
<td>the JavaCup source code</td>
</tr>
<tr>
<td>javadoc</td>
<td>a Java API documentation generator</td>
<td>the JavaCup source code</td>
</tr>
<tr>
<td>JavaCup</td>
<td>an LALR parser-generator for Java</td>
<td>the CORBA grammar</td>
</tr>
<tr>
<td>JTB</td>
<td>a frontend for Sun's JavaCC</td>
<td>the Java 1.2 grammar</td>
</tr>
<tr>
<td>JavaWiz</td>
<td>our watermarking system</td>
<td>the JTB source code</td>
</tr>
<tr>
<td>JAX</td>
<td>a Java packaging tool from IBM</td>
<td>Hanoi demo shipped with JAX</td>
</tr>
<tr>
<td>BLOAT</td>
<td>a Java bytecode optimization tool</td>
<td>the JavaWiz source code</td>
</tr>
</tbody>
</table>
## Experimental Results

<table>
<thead>
<tr>
<th>Program</th>
<th>Lines of Code</th>
<th>Time to Watermark</th>
<th>Running Time</th>
<th>Needed Heap Space</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>before</td>
<td>after</td>
<td>before</td>
<td>after</td>
</tr>
<tr>
<td>javac</td>
<td>6,053</td>
<td>6,946</td>
<td>3.4s</td>
<td>2.8s</td>
</tr>
<tr>
<td>javadoc</td>
<td>7,812</td>
<td>8,407</td>
<td>3.7s</td>
<td>11.8s</td>
</tr>
<tr>
<td>JavaCup</td>
<td>11,029</td>
<td>11,799</td>
<td>2.8s</td>
<td>5.3s</td>
</tr>
<tr>
<td>JTB</td>
<td>13,125</td>
<td>14,059</td>
<td>3.0s</td>
<td>5.6s</td>
</tr>
<tr>
<td>JavaWiz</td>
<td>22,397</td>
<td>23,152</td>
<td>3.4s</td>
<td>3.0s</td>
</tr>
<tr>
<td>JAX</td>
<td>22,705</td>
<td>23,537</td>
<td>2.7s</td>
<td>12.3s</td>
</tr>
<tr>
<td>BLOAT</td>
<td>55,518</td>
<td>56,052</td>
<td>4.1s</td>
<td>3.6s</td>
</tr>
</tbody>
</table>
Conclusion

- Our watermarking system is efficient
- The watermarked programs use only moderately more time and space
- The watermarked programs are resilient to attacks.
- Coming soon: measurements of the extraction process.