Software Theft Detection Through Program Identification

Ginger Myles
mylesg@cs.arizona.edu

University of Arizona
Department of Computer Science
and
IBM Almaden Research Center

March 31, 2006
Preventive

Forensic

Legal

Preventive

---

105,000 jobs, $5.3 billion in wages, $1.4 billion in tax revenue.

---

What is Software Watermarking?

- Embed a unique identifier in a program.

- Watermarking
  - Same identifier
  - Copyright notice
  - Discourages theft

- Fingerprinting
  - Different identifier
  - Customer identification
  - Trace illegal copies

- Discourages but does not prevent illegal copying and redistribution.
A watermarking system consists of two functions:

- \( \text{embed}(P, w, key) \rightarrow P' \)
- \( \text{recognize}(P', key) \rightarrow w \)
We want to develop an algorithm such that when we embed the watermark $W$ in the program $P$

- $W$ is resilient to various attacks.
- $W$ is stealthy.
- $W$ is large (high bit-rate).
- The overhead (space and time) is low.
Attacks on Software Watermarks

**Subtractive Attack:** The adversary examines the (disassembled/de-compiled) program in an attempt to discover the watermark and to remove all or part of it from the code.
Additive Attack: The adversary adds a new watermark in order to make it hard for the IP owner to prove that her watermark is actually the original.
Distortive Attack: A series of semantics-preserving transformations are applied to the software in an attempt to render the watermark useless.
Collusive Attack: The adversary compares two copies of the software which contain different fingerprints in order to identify the location.
Categories

- **Static**: the watermark is stored directly in the data or code sections of a native executable or class file.
Categories

- **Static:** the watermark is stored directly in the data or code sections of a native executable or class file.
- **Dynamic:** the watermark is stored in the run-time structures of the program.

![Tic-Tac-Toe Game](image)
Branch-Based Watermarking

- Dynamic execution trace technique.
- Semantics extending transformation.
- Draws from code obfuscation and tamper detection.
- Embed an authorship mark and a fingerprint mark.

International Information Hiding Workshop, 2005.
Branch-Based Watermarking

$F_{M I}$

Embed code to generate fingerprint as the program executes on a secret input sequence $I_0, \ldots, I_n$.

Fingerprint Branch Function (FBF)

Key: Link proper program execution and fingerprint generation.
Select a sequence of branch instructions.

Replace branch instructions by calls to FBF.

FBF will return execution control to original target.
Branch-Based WM — Basis

Branch Function obfuscation $^a$:

- Designed to disrupt static disassembly.
- Exploits assumption that a function call returns to the instruction immediately following the call instruction.
- Execution is rerouted through the branch function.
- The correct target is identified based on the call location.
  \[ T[h(j_i)] = t_i - j_i \]
- Return address on the stack is overwritten.


March 31, 2006
Branch-Based Watermarking

Fingerprint Branch Function

1. Performs an integrity check of the program producing the value $v_i$.
2. Generate the next key using a secure one-way function, the previous key, and the integrity check value.
   \[ k_{i+1} = SHA1(v_i, k_i) \]
3. Uses $k_{i+1}$ to identify the instruction where execution will resume.

Location for authorship mark embedding.

\[ k_{i+1} = SHA1[(k_i \oplus AM)||v_i]] \]
Implementation overview:

- Details are architecture specific.
- Java language features prohibit return address modification.
- Embedding process for x86 and Java.
BB WM — x86 Implementation Overview

Original Application

Annotated Application

Watermarked Application

March 31, 2006
Five general steps:

- FBF construction
- For each $f \in F$
  - Branch instruction selection
  - Branch instruction replacement
- Table construction
- FM calculation
Branch Function Construction

Fingerprint Branch Function

Original Application

- $f_1$
- $f_2$
- $f_3$

Modified Application

- $f_1$
- $f_2$
- $f_3$
- FBF

$\begin{align*}
    d_{i+1} &= T[h(k_{i+1})] \\
    \text{ret}_{new} &= \text{ret}_{old} + d_{i+1}
\end{align*}$
Branch Instruction Selection

Modified Application

Execution Trace

\[
f_1
f_2
f_3
f_2
\]

0
push
mov
cmp
jmp

1
mov
add
jmp

2
mov
sub
mov

3
mov
push
push
push

4
add
pop
ret
Branch Instruction Selection

Modified Application

Execution Trace

\[ f_1 \]
\[ f_2 \]
\[ f_3 \]
\[ FBF \]

\[ 0 \]
\[ \text{push} \]
\[ \text{mov} \]
\[ \text{cmp} \]
\[ \text{jge} \]

\[ 1 \]
\[ \text{mov} \]
\[ \text{add} \]
\[ \text{jmp} \]

\[ 2 \]
\[ \text{mov} \]
\[ \text{sub} \]
\[ \text{mov} \]

\[ 3 \]
\[ \text{mov} \]
\[ \text{push} \]
\[ \text{push} \]
\[ \text{push} \]
\[ \text{call} \]

\[ 4 \]
\[ \text{add} \]
\[ \text{pop} \]
\[ \text{ret} \]
Branch Instruction Selection

Modified Application

Execution Trace

\[ f_1 \]
\[ f_2 \]
\[ f_3 \]
\[ FBF \]

\[ f_2 \]
\[ f_2 \]
\[ f_2 \]

0
push
mov
cmp
jge

1
mov
add
jmp

2
mov
sub
mov

3
mov
push
push
call

4
add
pop
ret

\[ k_0 \]
\[ AM \]
\[ block 0 \]
\[ v_0 \]

\[ k_1 \]
\[ SHA-1 \]

\[ k_1 \]
\[ AM \]
\[ block 3 \]
\[ v_1' \]

\[ k_2' \]
\[ SHA-1 \]
Branch Instruction Selection

Template

March 31, 2006
Branch Instruction Replacement

Modified Application

Watermarked Application

March 31, 2006
Displacement Table Construction

Watermarked Application

\[
\begin{align*}
  f_1 & \quad \ldots \quad d_i \quad \ldots \\
  f_2 & \quad \ldots \\
  & \quad \text{FBF}() \\
  f_3 & \quad \ldots \\
  & \quad \text{FBF}() \\
  \text{FBF} & \\
\end{align*}
\]

Watermarked Application

\[
\begin{align*}
  f_1 & \quad \ldots \\
  f_2 & \quad \ldots \\
  & \quad \text{FBF}() \\
  f_3 & \quad \ldots \\
  & \quad \text{FBF}() \\
  \text{FBF} & \\
  & \quad \ldots \\
\end{align*}
\]

\[h(k_i)\]
Fingerprint Mark Calculation

\[ \Rightarrow k_2 + k_i + k_2 = \text{FM} \]
Only static variation occurs in the displacement table.
Each copy can be linked to the purchaser.
BB WM — x86 Evaluation

- Prototype for Windows executables.
- Evaluated with respect to five properties.
Credibility: Occurrence of false positives and false negatives.

Result: high credibility
Data-rate: The number of watermark bits that can be embedded.

Result: high data-rate
Stealth: Degree of similarity between the watermarked and the original code.

Result: low stealth

Result: minimal overhead
Branch-Based WM — Evaluation

Performance Overhead:

<table>
<thead>
<tr>
<th>Program</th>
<th>Slowdown</th>
</tr>
</thead>
<tbody>
<tr>
<td>gzip</td>
<td>0%</td>
</tr>
<tr>
<td>vpr</td>
<td>0%</td>
</tr>
<tr>
<td>mcf</td>
<td>0%</td>
</tr>
<tr>
<td>crafty</td>
<td>0%</td>
</tr>
<tr>
<td>parser</td>
<td>13%</td>
</tr>
<tr>
<td>gap</td>
<td>0%</td>
</tr>
<tr>
<td>vortex</td>
<td>0%</td>
</tr>
<tr>
<td>bzip2</td>
<td>-1%</td>
</tr>
<tr>
<td>twolf</td>
<td>1%</td>
</tr>
</tbody>
</table>
### Size Overhead:

<table>
<thead>
<tr>
<th>Program</th>
<th>Branches</th>
<th>Increase</th>
</tr>
</thead>
<tbody>
<tr>
<td>gzip</td>
<td>79</td>
<td>4%</td>
</tr>
<tr>
<td>vpr</td>
<td>405</td>
<td>19%</td>
</tr>
<tr>
<td>gcc</td>
<td>2124</td>
<td>62%</td>
</tr>
<tr>
<td>mcf</td>
<td>24</td>
<td>6%</td>
</tr>
<tr>
<td>crafty</td>
<td>94</td>
<td>1%</td>
</tr>
<tr>
<td>parser</td>
<td>239</td>
<td>2%</td>
</tr>
<tr>
<td>gap</td>
<td>742</td>
<td>18%</td>
</tr>
<tr>
<td>vortex</td>
<td>477</td>
<td>9%</td>
</tr>
<tr>
<td>bzip2</td>
<td>135</td>
<td>9%</td>
</tr>
<tr>
<td>twolf</td>
<td>233</td>
<td>5%</td>
</tr>
</tbody>
</table>
Robustness: The ability to withstand additive, subtractive, distortive, and collusive attacks.

Result: high resistance to transformation
# BB WM — x86 Evaluation

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Cred</th>
<th>Data-rate</th>
<th>Overhead</th>
<th>Resistance to Attack</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Size</td>
<td>Performance</td>
</tr>
<tr>
<td>DM</td>
<td>high</td>
<td>useable</td>
<td>low</td>
<td>low</td>
</tr>
<tr>
<td>MC</td>
<td>high</td>
<td>exceptional</td>
<td>high</td>
<td>low</td>
</tr>
<tr>
<td>GCB</td>
<td>low</td>
<td>useable</td>
<td>low</td>
<td>low</td>
</tr>
<tr>
<td>SS</td>
<td>medium</td>
<td>exceptional</td>
<td>low</td>
<td>low</td>
</tr>
<tr>
<td>DMMethod</td>
<td>high</td>
<td>exceptional</td>
<td>low</td>
<td>low</td>
</tr>
<tr>
<td>GTW</td>
<td>high</td>
<td>exceptional</td>
<td>extreme</td>
<td>high</td>
</tr>
<tr>
<td>A1</td>
<td>high</td>
<td>exceptional</td>
<td>low</td>
<td>moderate</td>
</tr>
<tr>
<td>A2</td>
<td>high</td>
<td>exceptional</td>
<td>moderate</td>
<td>moderate</td>
</tr>
<tr>
<td>CT</td>
<td>high</td>
<td>exceptional</td>
<td>moderate</td>
<td>moderate</td>
</tr>
<tr>
<td>PBW</td>
<td>high</td>
<td>exceptional</td>
<td>low</td>
<td>low</td>
</tr>
<tr>
<td>BB</td>
<td>high</td>
<td>exceptional</td>
<td>moderate</td>
<td>low</td>
</tr>
</tbody>
</table>
Limiting aspect is difficulty in modifying program counter register.

Maintain essence through the Java interface and explicitly thrown exceptions.
What is Software Birthmarking?

- A technique used to address the illegal distribution of all or some part of a program.
- Extract identifying characteristics from two programs to show that one is a copy of the other.
How Does Birthmarking Differ From Watermarking?

- It is often necessary to alter existing code or add code to the application in order to embed the watermark.
- Birthmarks **cannot** prove authorship or identify the source of an illegal redistribution.
- Birthmarks only **confirm** that one program is a copy of another.
A Birthmarking system consists of the following functions:

- \( \text{extract}(p) \rightarrow b_p \)
- \( \text{extract}(q) \rightarrow b_q \)
- \( \text{similarity}(b_p, b_q) \rightarrow [0, 1] \)
A Birthmarking system consists of the following functions:

- \( \text{extract}(p) \rightarrow b_p \)
- \( \text{extract}(q) \rightarrow b_q \)
- \( \text{similarity}(b_p, b_q) \rightarrow [0, 1] \)

We want to develop an algorithm such that

- \( b \) is resilient to various transformations.
- \( b \) is credible.
Definition (Software Derivative): Given two sets of modules \( p \) and \( q \) and a value \( \epsilon < 1 \), we say that \( q \) is a derivative of \( p \) if:

- same external behavior
- \( 1 - \text{similarity}(b_p, b_q) < \epsilon \)
Definition (Software Derivative): Given two sets of modules $p$ and $q$ and a value $\epsilon < 1$, we say that $q$ is a derivative of $p$ if:

- same external behavior
- $1 - \text{similarity}(b_p, b_q) < \epsilon$

Assumption:

- $1 - \text{similarity}(b_p, b_q) \leq 0.2 \Rightarrow \text{derivative}$
- $0.2 < 1 - \text{similarity}(b_p, b_q) \leq 0.4 \Rightarrow \text{inconclusive}$
- $1 - \text{similarity}(b_p, b_q) > 0.4 \Rightarrow \text{not derivative}$
Birthmarking Taxonomy

Static vs. Dynamic

- **Static**: the set of characteristics is extracted from the statically available information in a program.
  - e.g. the types or initial values of the fields
- **Dynamic**: relies on information gathered from the execution of the application to extract the set of characteristics.
Whole Program Path Birthmark

- The first dynamic birthmarking technique.
- Uniquely identifies a program based on a complete control flow trace of its execution using Whole Program Paths.
Whole Program Paths

Technique presented by Larus to represent a program’s dynamic control flow.

```c
int a;
for(int i=0; i < 5; i++){
    if (i < 3)
        a = 1;
    else
        a = 2;
}
```

---

Whole Program Paths

\[
\begin{array}{c}
1 \\
2 \\
3 \\
4 \\
5 \\
6 \\
7 \\
8
\end{array}
\Rightarrow \begin{array}{c}
123472347256725678
\end{array}
\]
Whole Program Paths

\[
\begin{array}{c}
\text{123472347256725678} \Rightarrow \text{SEQUITUR}^a \text{ algorithm} \Rightarrow \begin{align*}
R0 &\rightarrow 1 \ R1 \ R1 \ R2 \ R2 \ 8 \\
R1 &\rightarrow 2 \ 3 \ 4 \ 7 \\
R2 &\rightarrow 2 \ 5 \ 6 \ 7
\end{align*}
\end{array}
\]

---

Whole Program Paths

\[
\begin{align*}
R0 &\rightarrow 1 \ R1 \ R1 \ R2 \ R2 \ 8 \\
R1 &\rightarrow 2 \ 3 \ 4 \ 7 \\
R2 &\rightarrow 2 \ 5 \ 6 \ 7 \\
\end{align*}
\]

\[
\begin{array}{c}
R0 \\
R1 \\
R2 \\
\end{array}
\]

\[
\begin{array}{c}
1 \\
3 \\
4 \\
2 \\
7 \\
5 \\
6 \\
8 \\
\end{array}
\]

⇒
Interested in regularity.

Internal nodes are more difficult to modify through program transformation.
WPP Birthmark Similarity

- Modified version of the graph distance metric developed by Bunke and Shearer \(^a\).
- The percentage of \(G_1\) that exists in \(G_2\).
- Finding the maximal common subgraph \(G_3\).

\(^a\)H. Bunke and K. Shearer. A Graph Distance Metric Based on the Maximal Common Subgraph, 1998.
Definition (WPP Birthmark Similarity): Let $b_p = G_p$ and $b_q = G_q$ be WPP birthmarks extracted from programs $p$ and $q$ when executed with the input sequence $I$. The similarity between $b_p$ and $b_q$ is defined by:

$$\text{similarity}(b_p, b_q) = \frac{|mcs(G_p, G_q)|}{|G_p|}$$

where $|G| = |V| + |E|$. 
WPP Birthmark Similarity Example
Computing MCS for general graphs is NP-hard.
WPP Birthmark Similarity

- Computing MCS for general graphs is NP-hard.
- Our graphs are DAG.
  - Each has a single root.
  - We can construct a topological sort of the vertices.
- Reliable heuristic which runs in $O(|V_p| \times |V_q|^3)$ time and $(|V_p|^2 + |V_q|^2)$ space.
Complexity of computing similarity is major limiting factor of the technique for large programs.
WPP Birthmark Evaluation

Three experiments:

- **Independence**: ability to distinguish between two programs that are independently implemented yet accomplish the same task.
- Iterative and recursive versions of factorial and Fibonacci programs.

False positive: ability to distinguish between two programs which do not accomplish the same task.

Resistance: ability to withstand semantics-preserving transformations.

Seven Java applications and five obfuscation/optimization tools.
WPP Birthmark Evaluation

Three experiments:

- **Independence**: ability to distinguish between two programs that are independently implemented yet accomplish the same task.
  
  Iterative and recursive versions of factorial and Fibonacci programs.

- **False positive**: ability to distinguish between two programs which do not accomplish the same task.
  
  Seven Java applications which vary in size and complexity.
WPP Birthmark Evaluation

Three experiments:

- **Independence:** ability to distinguish between two programs that are independently implemented yet accomplish the same task.
  - Iterative and recursive versions of factorial and Fibonacci programs.

- **False positive:** ability to distinguish between two programs which do not accomplish the same task.
  - Seven Java applications which vary in size and complexity.

- **Resistance:** ability to withstand semantics-preserving transformations.
  - Seven Java applications and five obfuscation/optimization tools.
Independence experiment:

<table>
<thead>
<tr>
<th>Program</th>
<th>WPP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Factorial</td>
<td>±</td>
</tr>
<tr>
<td>Fibonacci</td>
<td>±</td>
</tr>
</tbody>
</table>

⊕ : derivative  
± : inconclusive  
⊖ : not derivative
### WPP Birthmark Evaluation

#### False positive experiment:

<table>
<thead>
<tr>
<th>Programs</th>
<th>WPP</th>
</tr>
</thead>
<tbody>
<tr>
<td>IterFact, IterFib</td>
<td>±</td>
</tr>
<tr>
<td>IterFact, RecurFib</td>
<td>±</td>
</tr>
<tr>
<td>IterFact, decode</td>
<td>±</td>
</tr>
<tr>
<td>IterFact, fft</td>
<td>±</td>
</tr>
<tr>
<td>IterFact, wc</td>
<td>±</td>
</tr>
<tr>
<td>RecurFact, IterFib</td>
<td>±</td>
</tr>
<tr>
<td>RecurFact, RecurFib</td>
<td>±</td>
</tr>
<tr>
<td>RecurFact, decode</td>
<td>±</td>
</tr>
<tr>
<td>RecurFact fft</td>
<td>±</td>
</tr>
<tr>
<td>RecurFact, wc</td>
<td>±</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Programs</th>
<th>WPP</th>
</tr>
</thead>
<tbody>
<tr>
<td>IterFib, decode</td>
<td>⊕</td>
</tr>
<tr>
<td>IterFib, fft</td>
<td>±</td>
</tr>
<tr>
<td>IterFib, wc</td>
<td>±</td>
</tr>
<tr>
<td>RecurFib, decode</td>
<td>±</td>
</tr>
<tr>
<td>RecurFib, fft</td>
<td>±</td>
</tr>
<tr>
<td>decode, fft</td>
<td>±</td>
</tr>
<tr>
<td>decode, wc</td>
<td>±</td>
</tr>
<tr>
<td>fft, wc</td>
<td>⊕</td>
</tr>
</tbody>
</table>

⊕ : derivative, ± : inconclusive, ⊥ : not derivative

March 31, 2006
# WPP Birthmark Evaluation

## Resistance experiment:

<table>
<thead>
<tr>
<th>Obfuscation</th>
<th>IterFact</th>
<th>IterFib</th>
<th>RecurFact</th>
<th>RecurFib</th>
<th>decode</th>
<th>fft</th>
<th>wc</th>
</tr>
</thead>
<tbody>
<tr>
<td>SM Test 1</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SM Test 1a</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SM Test 2</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SM Test 3</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SM Test 4</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SM Test 4a</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SM Test 5</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SM Test 6</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Smokescreen</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Codeshield</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Zelix Klassmaster</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>jarg</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
WPP Birthmark Evaluation

Compared with two other birthmarking techniques:


March 31, 2006
WPP Birthmark Evaluation

Compared with two other birthmarking techniques:

- TaNaMM: static technique specific to Java classfiles.
  - Is composed of four individual birthmarks \textsuperscript{a}.
  - low level of credibility
  - high resistance to transformation


March 31, 2006
WPP Birthmark Evaluation

Compared with two other birthmarking techniques:

- **TaNaMM**: static technique specific to Java classfiles.
  - Is composed of four individual birthmarks \( a \).
  - low level of credibility
  - high resistance to transformation

- **K-gram**: static technique based on instruction sequences \( b \).
  - high level of credibility
  - moderate resistance to transformation

---


March 31, 2006
WPP provides the best balance between the properties.
Birthmarks and Watermarks

- Birthmarks provide weaker evidence.
  - Can indicate one program is likely to be a copy of another.
  - Cannot determine original author.
  - Cannot determine who is guilty of piracy.

- Birthmarks can be used in instances where watermarking is not feasible.

- Birthmarks can be used in conjunction with watermarking to provide stronger evidence of theft.

- There are instances where watermarks are destroyed but birthmarks are not.
Summary

- Software piracy is widespread, decentralized problem.
- Software watermarking and birthmarking are just two of the techniques we can use to address the issue.
- A variety of algorithms have been proposed, but no perfect algorithm has yet to be developed.


void main(int argc, char *argv[])
{
    int x = atoi(argv[1]);
    printf("%d! = %d\n", x, factorial(x));
}

int factorial(int x){
    if(x == 1)
        return x;
    return (x * factorial(x-1));
}

long key = seed;

void main(int argc, char *argv[]){
    int x = branchFunction(argv[1]);
    branchFunction("%d! = %d\n", x, branchFunction(x));
}

int factorial(int x){
    if(x == 1)
        return x;
    return (x * factorial(x-1));
}

void branchFunction(void *x){
    key = evolveKey(key);
    return;
}
<table>
<thead>
<tr>
<th>Program</th>
<th>Total Branches</th>
<th>Usable excluding conditionals</th>
<th>Usable including conditionals</th>
</tr>
</thead>
<tbody>
<tr>
<td>gzip</td>
<td>2843</td>
<td>170</td>
<td>464</td>
</tr>
<tr>
<td>vpr</td>
<td>5814</td>
<td>674</td>
<td>1153</td>
</tr>
<tr>
<td>gcc</td>
<td>28136</td>
<td>3056</td>
<td>4886</td>
</tr>
<tr>
<td>mcf</td>
<td>2028</td>
<td>89</td>
<td>290</td>
</tr>
<tr>
<td>crafty</td>
<td>3340</td>
<td>178</td>
<td>496</td>
</tr>
<tr>
<td>parser</td>
<td>5628</td>
<td>522</td>
<td>864</td>
</tr>
<tr>
<td>gap</td>
<td>18999</td>
<td>1027</td>
<td>1942</td>
</tr>
<tr>
<td>vortex</td>
<td>16144</td>
<td>1049</td>
<td>3462</td>
</tr>
<tr>
<td>bzip2</td>
<td>2354</td>
<td>211</td>
<td>457</td>
</tr>
<tr>
<td>twolf</td>
<td>4397</td>
<td>429</td>
<td>729</td>
</tr>
</tbody>
</table>