# Detecting Software Theft via Whole Program Path Birthmarks

Ginger Myles

Christian Collberg

{mylesg,collberg}@cs.arizona.edu

University of Arizona

Department of Computer Science

# What is a Software Birthmark?

- A software birthmark is a unique characteristic, or set of characteristics, that a program possesses which can be used to identify the program.

# What is a Software Birthmark?

- One of the first occurrences of the term was by Derrick Grover (1989) where the term was used to mean characteristics occurring in the program by chance which could be used to aid in program identification.
  - The term differed from software watermarking/fingerprinting in that the identification characteristics are not intentionally placed in the code.

# Why use Software Birthmarking?

- It can be used to detect software theft.
- The general idea is that if two programs $p$ and $q$ have the same birthmark then it is highly likely they are copies.

# Early Use of Birthmarking

- In the 80's IBM successfully sued software pirates for copying their PC-AT ROM by showing that the order that the registers were pushed and popped was the same.

# Birthmarking Taxonomy

- A birthmark is generally classified along 2 axes:
  - Class/Module vs. Program Level
  - Static vs. Dynamic

# Birthmarking Taxonomy

- Class/Module vs. Program Level
  - The level at which the birthmark is computed.
  - A Class/Module level birthmark can be used to identify a single class/module or an entire program.
  - A program level birthmark requires an entire program to extract the birthmark and cannot be broken down into a smaller unit.

# 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.

# Related Work

- Idea similar to that of:
  - Plagiarism detection
  - Code clones
- What makes it unique is that a birthmark is computed at the machine code level and considers semantics-preserving transformations.

# Static BMs — Tamada et al.

- Specific to Java class files.
- Composed of 4 individual birthmarks
  - constant value in field variables (CVFV)
  - sequence of method calls (SMC)
  - inheritance structure (IS)
  - used classes (UC)
- IASTED International Conference on Software Engineering 2004.

# Dynamic BMs — Whole Program Path Birthmark

- The first dynamic birthmark 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 (1999) to represent a program's dynamic control flow.
- Constructed by collecting a trace of the path executed by the program.
- The trace is run through the SEQUITUR algorithm which compresses it and reveals its inherent regularity.
- The output of the SEQUITUR algorithm is a context-free grammar from which a directed acyclic graph (DAG) is produced.

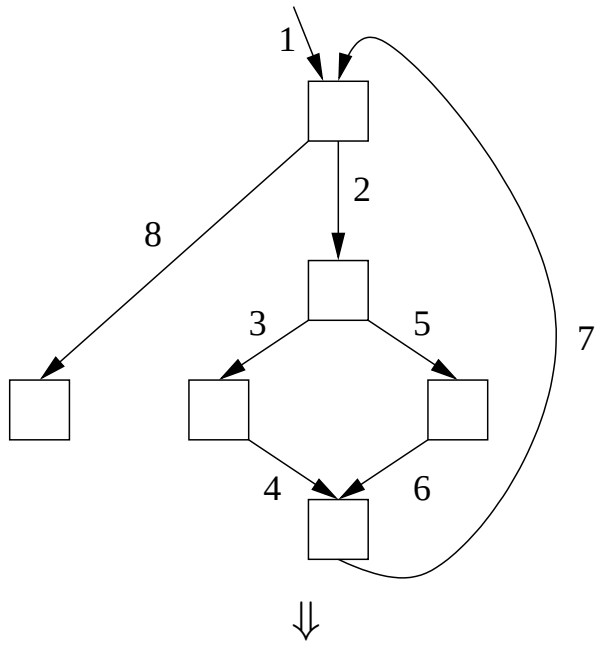
# Whole Program Paths

- Each rule of the grammar is composed of a non-terminal and a sequence of symbols which the non-terminal represents.
- To construct the DAG a node is added for each unique symbol. For each rule an edge is added from the non-terminal to each of the symbols it represents.
- The DAG is the WPP.

# Whole Program Paths Example

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

$\Rightarrow$




$\Downarrow$

```
R0 → 1 R1 R1 R2 R2 8
R1 → 2 3 4 7
R2 → 2 5 6 7
```

$\Leftarrow$

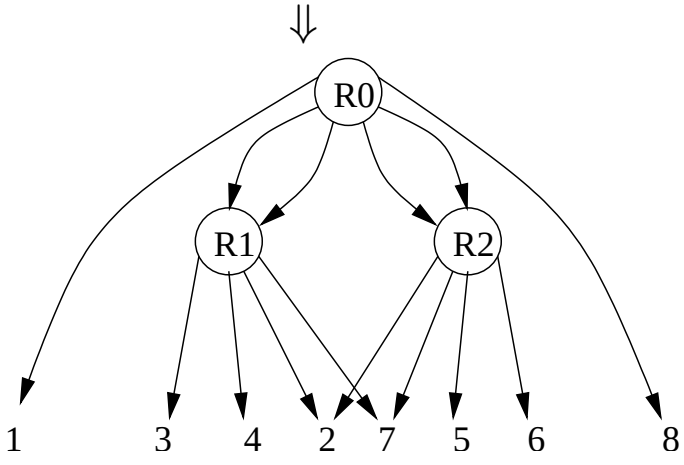
```
1234723472567256 78
```

# Whole Program Paths Example

```
R0 → 1 R1 R1 R2 R2 8
R1 → 2 3 4 7
R2 → 2 5 6 7
```

⇐

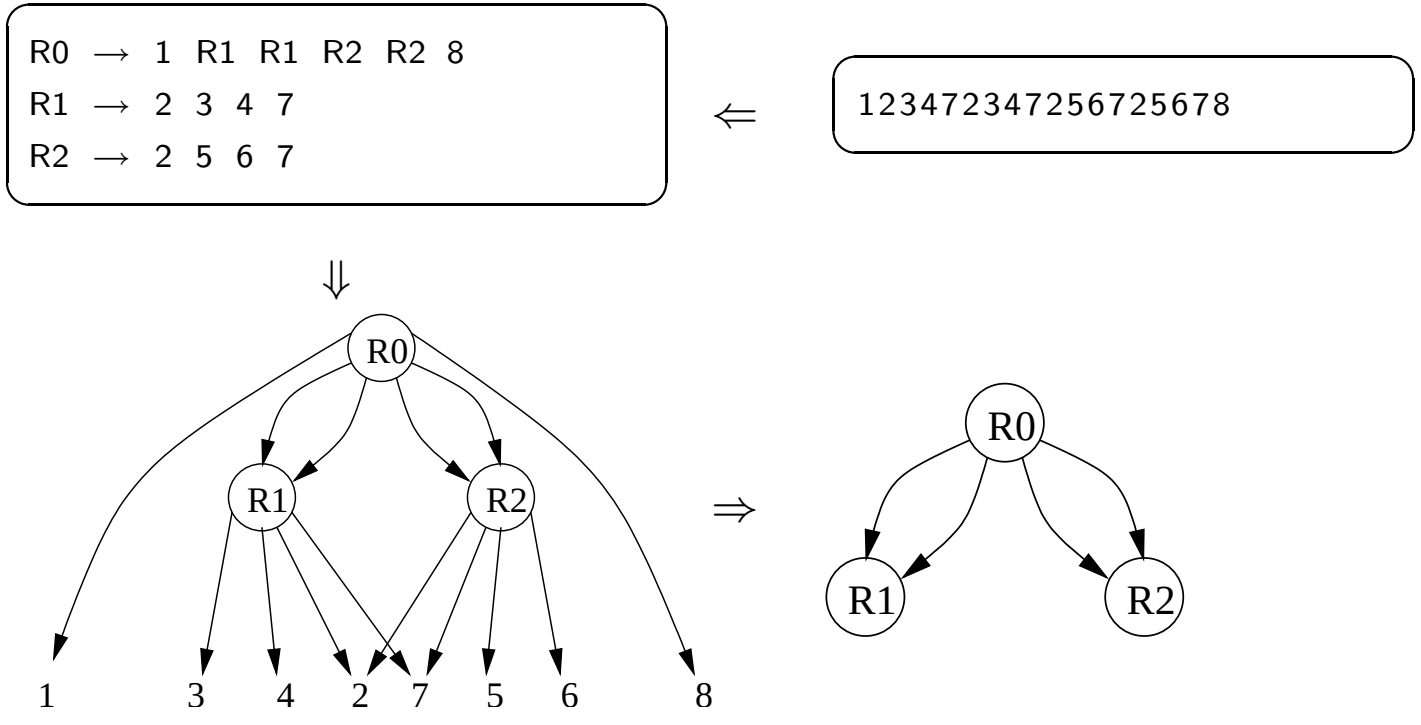
```
1234723472567256 78
```



# Whole Program Paths Birthmark

- The WPP birthmark is constructed in an identical manner as the WPP with the exception of the DAG in the final stage.
- Since we are only interested in the regularity we eliminate all terminal nodes in the DAG.
- It is the internal nodes which will be more difficult to modify through program transformation.

# Whole Program Paths Birthmark Example



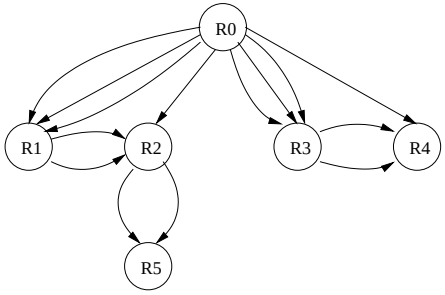
# WPP Birthmark Similarity

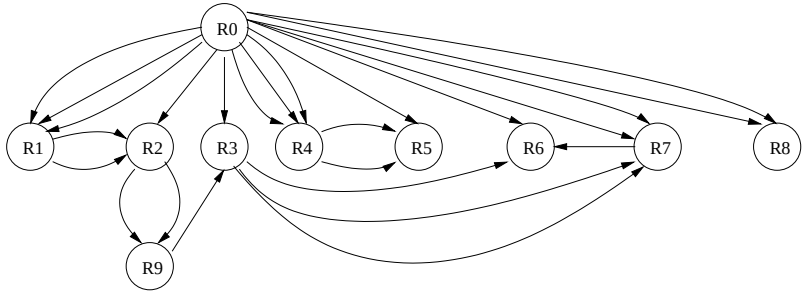
- Since it is highly unlikely that a hacker will redistribute an exact copy of the software, each birthmark technique must define a meaningful measure of similarity.

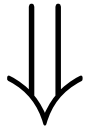
# WPP Birthmark Similarity

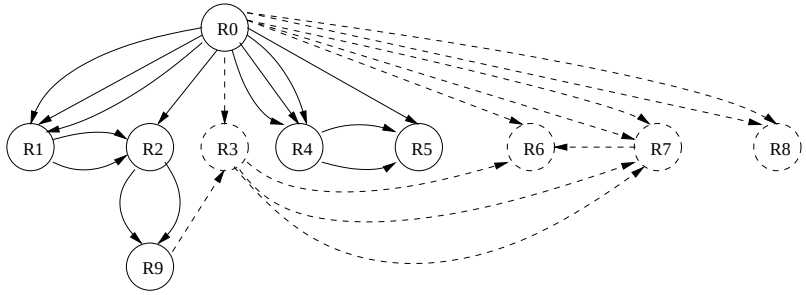
- Since it is highly unlikely that a hacker will redistribute an exact copy of the software, each birthmark technique must define a meaningful measure of similarity.
  - To compute similarity we use a slightly modified version of the graph distance metric developed by Bunke and Shearer (1998).
  - The percentage of $G_1$ that we are able to identify in $G_2$ indicates the similarity between the two programs.

# WPP Birthmark Similarity Example





# Birthmark Evaluation

- A birthmarking algorithm is evaluated based on 2 properties.
  - Credibility
  - Resilience to Transformation

# Birthmark Evaluation

- Credibility: Let $p$ and $q$ be independently written sets of modules which may accomplish the same task. We say $f$ is a credible measure if $f(p) \neq f(q)$.

# Birthmark Evaluation

- Credibility: Let $p$ and $q$ be independently written sets of modules which may accomplish the same task. We say $f$ is a credible measure if $f(p) \neq f(q)$.
- Resilience to Transformation: Let $p'$ be a set of modules obtained from $p$ by applying semantics-preserving transformations $\mathcal{T}$. We say that $f$ is resilient to $\mathcal{T}$ if $f(p) = f(p')$.

# Implementation and Evaluation

- The birthmark technique was incorporated into the SandMark framework.
  - Implemented in Java and the birthmark is calculated on Java bytecode.
- Evaluated their resilience using the obfuscations in SandMark as well as the obfuscators Codeshield and Smokescreen.

# WPP Birthmark Evaluation

- Credibility:
  - We examined the ability to distinguish between two independently written applications through iterative and recursive factorial and Fibonacci.

| Programs | Similarity |
|---|---|
| Factorial | 50% |
| Fibonacci | 7% |

# WPP Birthmark Evaluation

- Resilience:
  - The test application, wc.jar, is similar to the UNIX wc program.
  - Applied Codeshield, Smokescreen, and SandMark obfuscators.
  - For every obfuscation we were able to detect 100% similarity.

# Future Work

- Improve the implementation of the WPP birthmark.
- Conduct a more extensive evaluation.
- Modify the implementation so that the birthmark can be computed at the module level as well.
- Explore the possibility of using this technique for plagiarism detection.

Questions?