Software Watermarking: Model and Dynamic Embeddings

Author: Christian Collberg Clark Thomborson

POPL'99 SAN ANTONIO, TEXAS, USA, 20-22 JAN 1999

"We will be constructing new techniques which are resilient to a variety of semantics-preserving de-watermarking attacks."

Presented by: Nian Zhou
21 September 2000

Overview of paper and my focus

- Introduction
- Static Software Watermarking
- Dynamic Software Watermarking
- A Formal Model of Software Watermarking
- Dynamic Graph Watermarking
  - Overview and Working Principles
  - Embedding the Watermarking
  - Recognizing the Watermarking
  - Attacking Against the Watermarking (my focus)
  - Tamperproofing the Watermarking (my focus)
- Conclusion

What is semantics-preserving transformation ?

- Semantics-preserving transformations is one kind of distortive attacks.
- The definition:

$T_{sem}=\{t:t \mid P \in P, I \in dom(p), dom(p)=dom(t(p)), out(p,i)=out(t(p),i) \}$

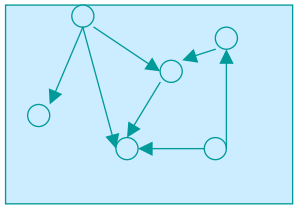
(In here, P is the set of programs. T is the set of transformations
Dom(p) is the input sequence accepted by P.
Out(p,i) is output of P on input I)

- Most of software watermarking techniques are susceptible to distortive attacks by semantics-preserving transformations.

Overview of Dynamic Graph Watermarking

- The central Idea is to embed a watermark in the topology of a dynamically built graph structure.
- Our technique:

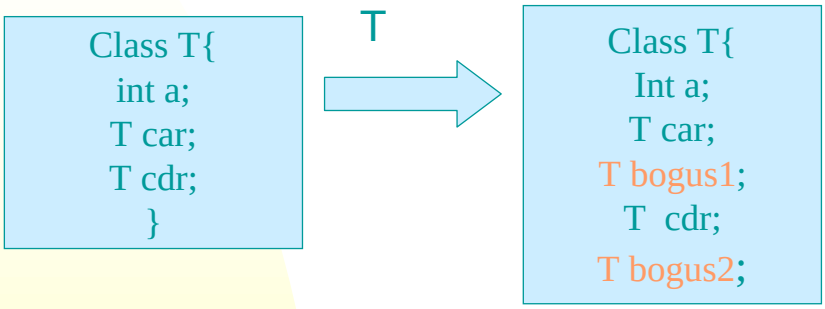
P←prime()
Q←prime()
N ← P×Q



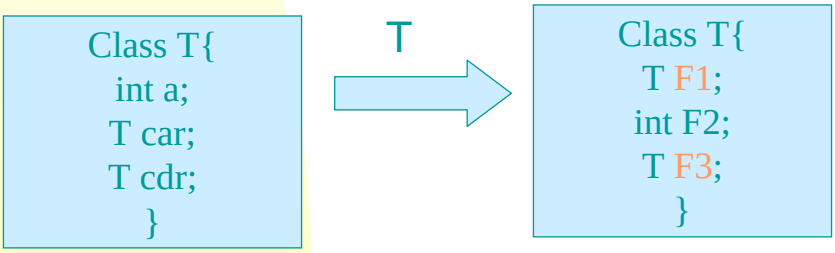
```
p=new node();
q=new node();
addEdge(p,q);
……W
```

Attacks Against the Watermark

- Adding(extra pointers) attacks:

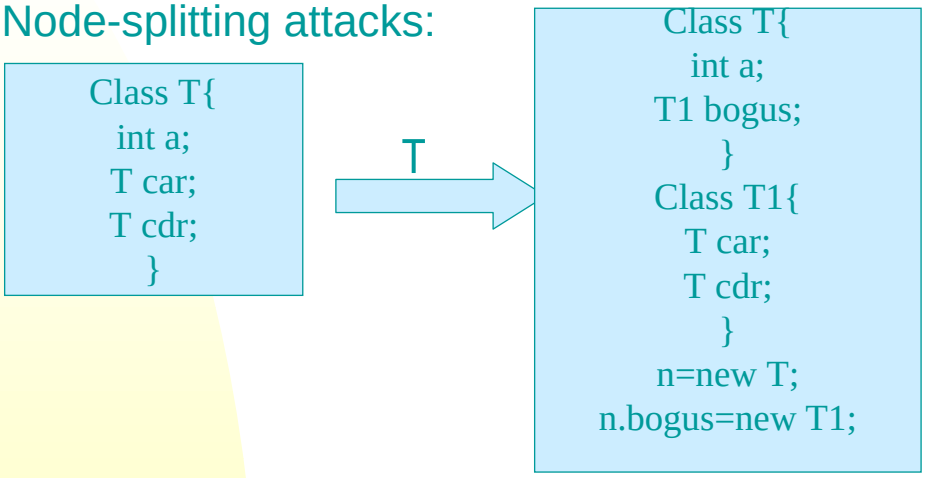


- Reordering and renaming attacks:



Attacks Against the Watermark(continued)

- Node-splitting attacks:

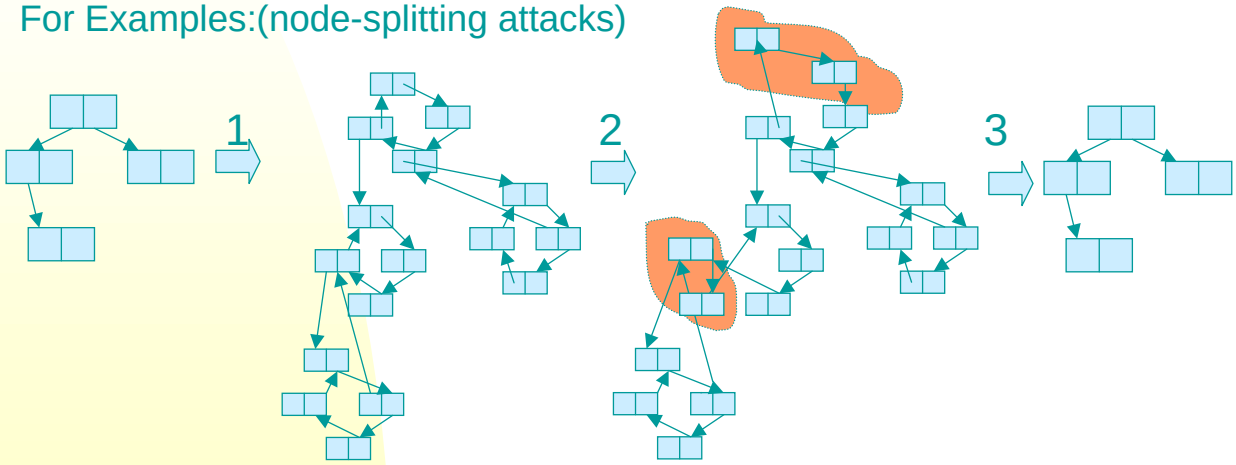


Tamperproofing the Watermarking

- Tamperproofing by the structure of graph:

The most attractive method makes certain types of attacks ineffective.

For Examples:(node-splitting attacks)



Tamperproofing by Reflection

- The reflection capabilities of Java give us a simple way of tamperproofing a graph watermark.
- For a given graph node Node:

```
class Node{public int a; public Node car,cdr}
```

The Java reflection class enable us check the intergrity of this type at runtime.

```
Field[] F=Node.class.getFields();
If(F.length !=3) die();
If(f[1].getType() != Node.class) die();
```

To prevent reordering and renaming attacks,we can access watermark pointers through reflection.(let car represented by the first relevant pointer)

Cropping Attacks

- :

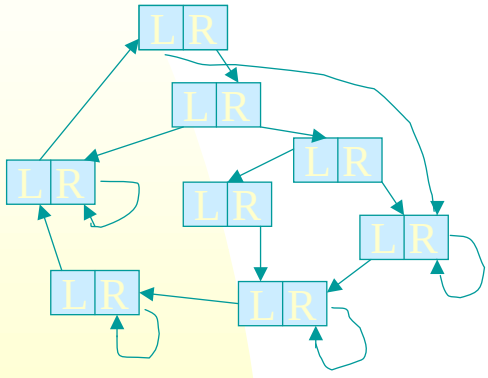
If the adversary can locate the code that build the watermark graph G,
And launch the adding (extra nodes) attacks.

What can We do?
Solution:Occasionally check the Integrity of G.

For Example:

Tamperproofing the Watermarking(continued)

Planted plane cubic tree on 2m=8 nodes:



1)A leaf node is recognized by its self-loop.

2)The root node can be found from any leaf node by following l-links.

3)left-most child of each internal node'right subtree is l-linked to the right-most child of its left subtree.

Conclusion

- A new family of software watermarking techniques embed marks into the topology of dynamic heap data structures.
- It makes the semantics-preserving transformations which make fundamental changes to a graph will be hard to construct.

Q1: If the adversary can locate the watermark in a graph and not just adding extra pointers(for example,remove the watermark totally if possible !) What should we do? That is the end of the day?

Q2: Does anybody has the experience of Java reflection? Can you should me an example of that?