Obfuscating Transformations

Yury Lifshits

Mathematics & Mechanics Faculty
Saint Petersburg State University

Spring 2005 – SETLab

Outline

1 What is Obfuscator?

- Notion of Obfuscator
- Anatomy of Obfuscator
- Obfuscator Characteristics

Outline

1 What is Obfuscator?

- Notion of Obfuscator
- Anatomy of Obfuscator
- Obfuscator Characteristics

2 Library of Obfuscating Transformations

- Program Representation
- Data & Control : Basic Tricks
- Control Flow Obfuscation
- Even more transformations

Outline

1 What is Obfuscator?

- Notion of Obfuscator
- Anatomy of Obfuscator
- Obfuscator Characteristics

2 Library of Obfuscating Transformations

- Program Representation
- Data & Control : Basic Tricks
- Control Flow Obfuscation
- Even more transformations

3 Obfuscation vs. Deobfuscation

- Classical Program Analysis
- Deobfuscation Hardness
- Further Research

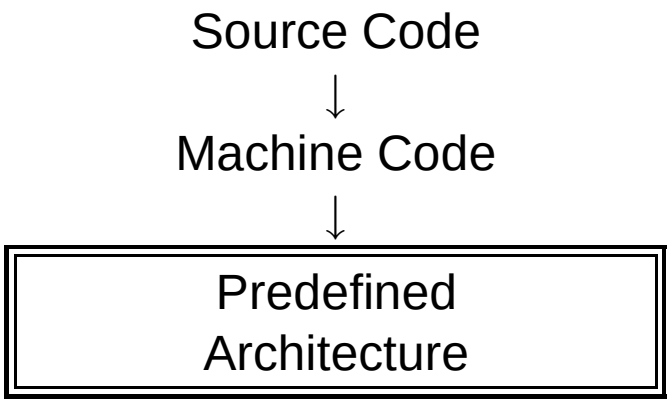
Motivation: Java Virtual Machine

What is difference between Java and others?

Motivation: Java Virtual Machine

What is difference between Java and others?

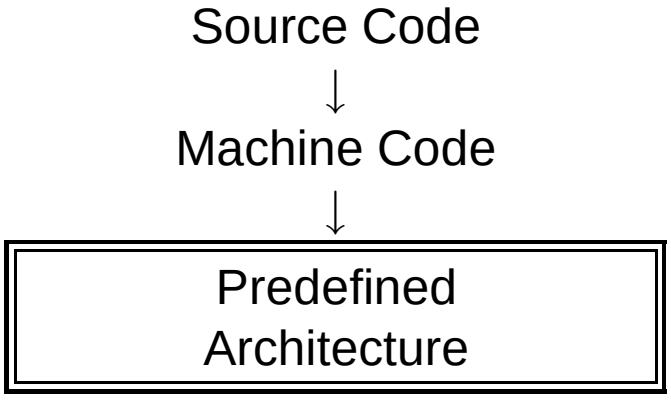
Most programming languages:



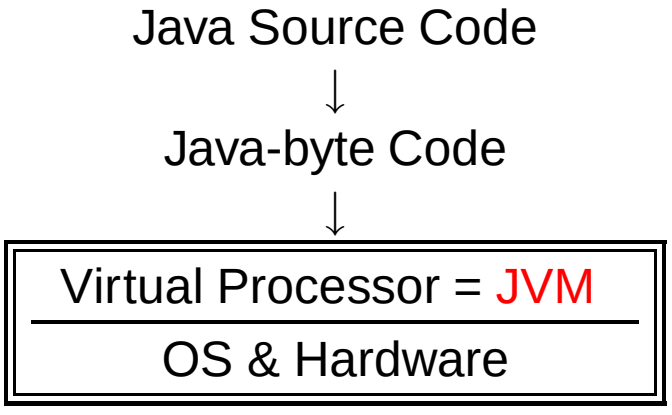
Motivation: Java Virtual Machine

What is difference between Java and others?

Most programming languages:



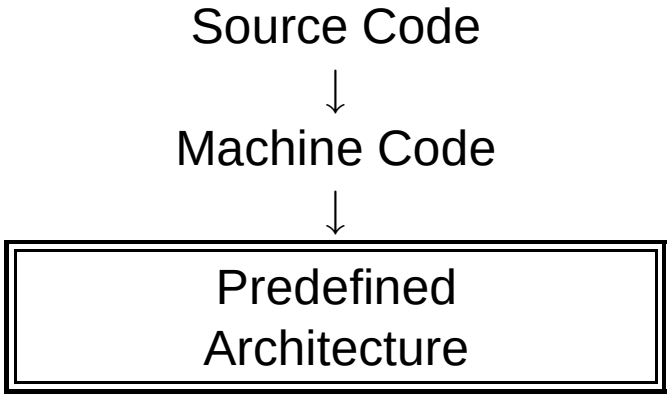
Java:



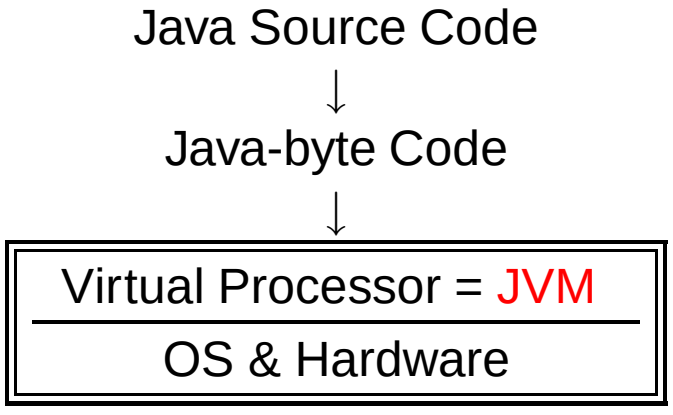
Motivation: Java Virtual Machine

What is difference between Java and others?

Most programming languages:



Java:



Java Virtual Machine: implemented on huge number of hardware architectures / operating systems

Notion of Obfuscator

Objectives:

⇨ Make code not readable by human

Notion of Obfuscator

Objectives:

- ⇨ Make code not readable by human
- ⇨ Make automated analysis difficult

Notion of Obfuscator

Objectives:

- ⇨ Make code not readable by human
- ⇨ Make automated analysis difficult
- ⇨ Make code more complicated

Notion of Obfuscator

Objectives:

- ⇨ Make code not readable by human
- ⇨ Make automated analysis difficult
- ⇨ Make code more complicated
- ⇨ Make decompilation & reverse engineering difficult

Anatomy of Obfuscator I

How real obfuscator works?

Anatomy of Obfuscator I

How real obfuscator works?

1. Prepares program to be obfuscated

Anatomy of Obfuscator I

How real obfuscator works?

1. Prepares program to be obfuscated
2. Makes a single transformation

Anatomy of Obfuscator I

How real obfuscator works?

1. Prepares program to be obfuscated
2. Makes a single transformation
3. Repeats **step 2** until task completed or constraints exceeded

Anatomy of Obfuscator II

How real obfuscator works (more precisely)?

Anatomy of Obfuscator II

How real obfuscator works (more precisely)?

The workflow is:

- ⇨ Parse input program

Anatomy of Obfuscator II

How real obfuscator works (more precisely)?

The workflow is:

- ⇨ Parse input program
  - Makes a list of obfuscation candidates: classes, variables, methods

Anatomy of Obfuscator II

How real obfuscator works (more precisely)?

The workflow is:

⇨ Parse input program

- Makes a list of obfuscation candidates: classes, variables, methods
- Constructs internal representation of the program (e.g. control flow and basic blocks)

Anatomy of Obfuscator II

How real obfuscator works (more precisely)?

The workflow is:

⇨ Parse input program

- Makes a list of obfuscation candidates: classes, variables, methods
- Constructs internal representation of the program (e.g. control flow and basic blocks)
- Makes some appropriateness suggestions

Anatomy of Obfuscator II

How real obfuscator works (more precisely)?

The workflow is:

- ⇨ Parse input program
  - Makes a list of obfuscation candidates: classes, variables, methods
  - Constructs internal representation of the program (e.g. control flow and basic blocks)
  - Makes some appropriateness suggestions
- ⇨ Main while loop (until constraints are exceeded or quality is achieved)

Anatomy of Obfuscator II

How real obfuscator works (more precisely)?

The workflow is:

- ⇨ Parse input program
  - Makes a list of obfuscation candidates: classes, variables, methods
  - Constructs internal representation of the program (e.g. control flow and basic blocks)
  - Makes some appropriateness suggestions
- ⇨ Main while loop (until constraints are exceeded or quality is achieved)
  - Choose next (by priority) element of the program to be obfuscated

Anatomy of Obfuscator II

How real obfuscator works (more precisely)?

The workflow is:

- ⇨ Parse input program
  - Makes a list of obfuscation candidates: classes, variables, methods
  - Constructs internal representation of the program (e.g. control flow and basic blocks)
  - Makes some appropriateness suggestions
- ⇨ Main while loop (until constraints are exceeded or quality is achieved)
  - Choose next (by priority) element of the program to be obfuscated
  - Implement appropriate obfuscating transformation (from obfuscator library)

Anatomy of Obfuscator II

How real obfuscator works (more precisely)?

The workflow is:

- ⇨ Parse input program
  - Makes a list of obfuscation candidates: classes, variables, methods
  - Constructs internal representation of the program (e.g. control flow and basic blocks)
  - Makes some appropriateness suggestions
- ⇨ Main while loop (until constraints are exceeded or quality is achieved)
  - Choose next (by priority) element of the program to be obfuscated
  - Implement appropriate obfuscating transformation (from obfuscator library)
  - Update internal representation

Quality of Obfuscation

Slide from Lecture 1

Quality of Obfuscation

Slide from Lecture 1

So you supposed to explain it to me...

Quality of Obfuscation

Slide from Lecture 1

So you supposed to explain it to me...

Strength can be measured by:

- ⇨ Potency

  $\frac{E(P')}{E(P)} - 1$
- ⇨ Resilience

  Trivial, weak, strong, full, one-way
- ⇨ Cost

  Free, cheap, costly, expensive
- ⇨ Stealthy

Program Complexity Metrics

We want: make program complicated

Program Complexity Metrics

We want: make program complicated

So what is program code complexity?

Program Complexity Metrics

We want: make program complicated

So what is program code complexity?

⇨ Program length

Program Complexity Metrics

We want: make program complicated

So what is program code complexity?

⇨ Program length

Number of operators and operands

Program Complexity Metrics

We want: make program complicated

So what is program code complexity?

⇨ Program length

Number of operators and operands

⇨ Data flow complexity

Program Complexity Metrics

We want: make program complicated

So what is program code complexity?

⇨ Program length

Number of operators and operands

⇨ Data flow complexity

Number of inter-block variable references

Program Complexity Metrics

We want: make program complicated

So what is program code complexity?

⇨ Program length

Number of operators and operands

⇨ Data flow complexity

Number of inter-block variable references

⇨ Cyclomatic complexity

Program Complexity Metrics

We want: make program complicated

So what is program code complexity?

⇨ Program length

Number of operators and operands

⇨ Data flow complexity

Number of inter-block variable references

⇨ Cyclomatic complexity

Number of predicates in a function

Program Complexity Metrics

We want: make program complicated

So what is program code complexity?

⇨ Program length

Number of operators and operands

⇨ Data flow complexity

Number of inter-block variable references

⇨ Cyclomatic complexity

Number of predicates in a function

⇨ Nesting complexity

Program Complexity Metrics

We want: make program complicated

So what is program code complexity?

- ⇨ Program length

  Number of operators and operands

- ⇨ Data flow complexity

  Number of inter-block variable references

- ⇨ Cyclomatic complexity

  Number of predicates in a function

- ⇨ Nesting complexity

  Number of nesting level of conditionals in a program

Program Complexity Metrics

We want: make program complicated

So what is program code complexity?

- ⇨ Program length

  Number of operators and operands
- ⇨ Data flow complexity

  Number of inter-block variable references
- ⇨ Cyclomatic complexity

  Number of predicates in a function
- ⇨ Nesting complexity

  Number of nesting level of conditionals in a program
- ⇨ Data structure complexity

Program Complexity Metrics

We want: make program complicated

So what is program code complexity?

- ⇨ Program length

  Number of operators and operands

- ⇨ Data flow complexity

  Number of inter-block variable references

- ⇨ Cyclomatic complexity

  Number of predicates in a function

- ⇨ Nesting complexity

  Number of nesting level of conditionals in a program

- ⇨ Data structure complexity

  Complexity of the static data structures in the program like variables, vectors, records

Program Complexity Metrics

We want: make program complicated

So what is program code complexity?

- ⇨ Program length

  Number of operators and operands

- ⇨ Data flow complexity

  Number of inter-block variable references

- ⇨ Cyclomatic complexity

  Number of predicates in a function

- ⇨ Nesting complexity

  Number of nesting level of conditionals in a program

- ⇨ Data structure complexity

  Complexity of the static data structures in the program like variables, vectors, records

- ⇨ OO Metrics

Program Complexity Metrics

We want: make program complicated

So what is program code complexity?

- ⇨ Program length

  Number of operators and operands

- ⇨ Data flow complexity

  Number of inter-block variable references

- ⇨ Cyclomatic complexity

  Number of predicates in a function

- ⇨ Nesting complexity

  Number of nesting level of conditionals in a program

- ⇨ Data structure complexity

  Complexity of the static data structures in the program like variables, vectors, records

- ⇨ OO Metrics

  Level of inheritance, coupling, number of methods triggered by another method, non-cohesiveness

Cost Analysis

What do we pay for security?

Cost Analysis

What do we pay for security?

⇨ Costs at creation time

Obfuscation need time!

Cost Analysis

What do we pay for security?

⇨ Costs at creation time

Obfuscation need time!

⇨ Costs at transmition time (resulting size)

Inlining library functions may increase size enormously!

Cost Analysis

What do we pay for security?

- ⇨ Costs at creation time

  Obfuscation need time!

- ⇨ Costs at transmition time (resulting size)

  Inlining library functions may increase size enormously!

- ⇨ Cost at execution time

  Checking procedures, dummy code, inlining

Cost Analysis

What do we pay for security?

- ⇨ Costs at creation time

  Obfuscation need time!

- ⇨ Costs at transmition time (resulting size)

  Inlining library functions may increase size enormously!

- ⇨ Cost at execution time

  Checking procedures, dummy code, inlining

- ⇨ Cost by not using efficiency enhancing mechanisms

  Caching is rarely possible; losing module structure

Statistics measurement

Other metrics?

Statistics measurement

Other metrics?

Statistics! What kind of?

Statistics measurement

Other metrics?

Statistics! What kind of?

⇨ Distribution of basic construstions

Statistics measurement

Other metrics?

Statistics! What kind of?

⇨ Distribution of basic construstions

- Use rare elements more often

Statistics measurement

Other metrics?

Statistics! What kind of?

- ⇨ Distribution of basic construstions
  - Use rare elements more often
- ⇨ Clustering of program elements. Uniform distribution – difficult to understand!

Statistics measurement

Other metrics?

Statistics! What kind of?

- ⇨ Distribution of basic construstions
  - Use rare elements more often
- ⇨ Clustering of program elements. Uniform distribution – difficult to understand!
  - Usage of variables

Statistics measurement

Other metrics?

Statistics! What kind of?

- ⇨ Distribution of basic construstions
  - Use rare elements more often
- ⇨ Clustering of program elements. Uniform distribution – difficult to understand!
  - Usage of variables
  - Data processing

Statistics measurement

Other metrics?

Statistics! What kind of?

- ⇨ Distribution of basic construstions
  - Use rare elements more often
- ⇨ Clustering of program elements. Uniform distribution – difficult to understand!
  - Usage of variables
  - Data processing
  - Control flow commands

Statistics measurement

Other metrics?

Statistics! What kind of?

- ⇨ Distribution of basic construstions
  - Use rare elements more often
- ⇨ Clustering of program elements. Uniform distribution – difficult to understand!
  - Usage of variables
  - Data processing
  - Control flow commands
- ⇨ Code patterns

Statistics measurement

Other metrics?

Statistics! What kind of?

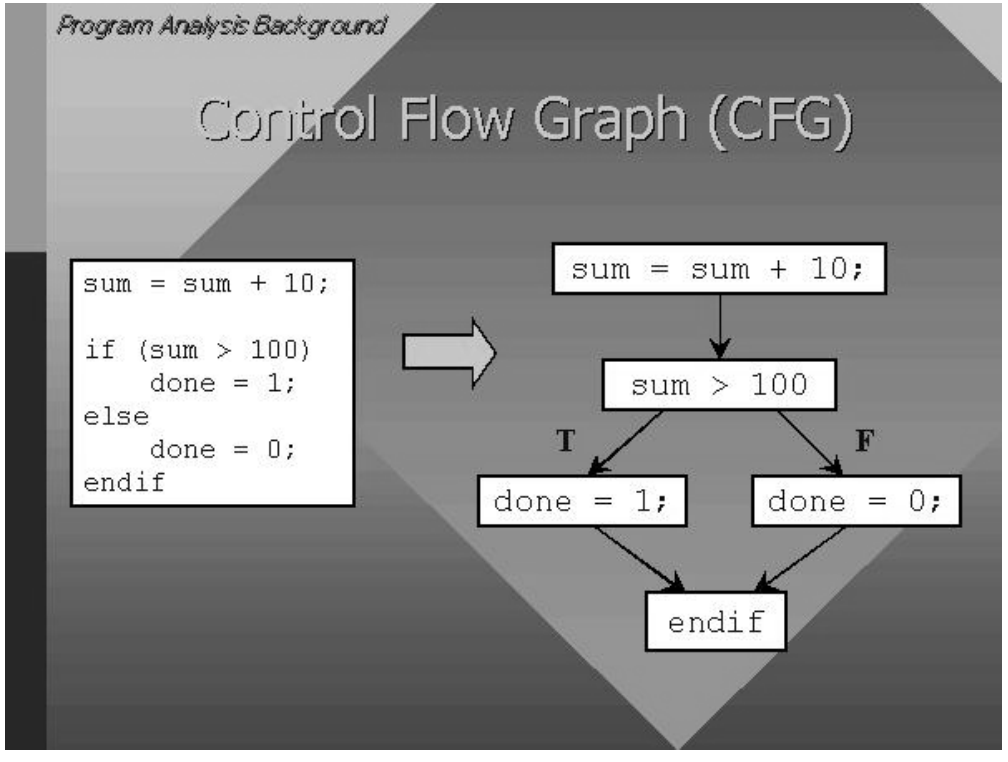
- ⇨ Distribution of basic construstions
  - Use rare elements more often
- ⇨ Clustering of program elements. Uniform distribution – difficult to understand!
  - Usage of variables
  - Data processing
  - Control flow commands
- ⇨ Code patterns
  - Destroy long patterns in program

Control Flow Model I

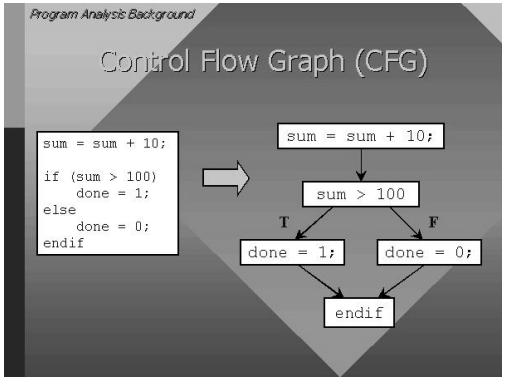
What is *program* from our point of view?

Control Flow Model I

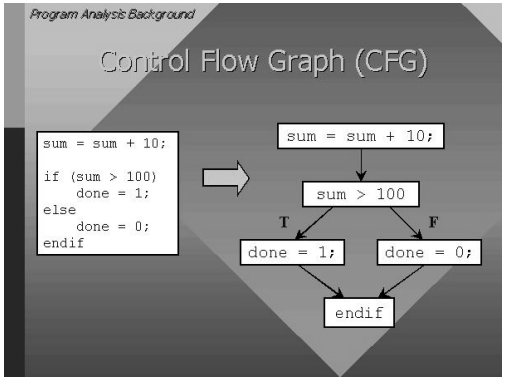
What is *program* from our point of view?



Control Flow Model II



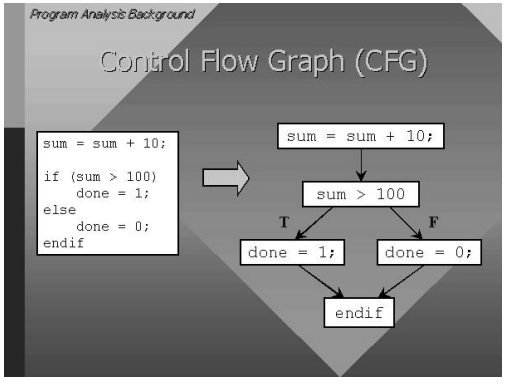
Control Flow Model II



Compilers and program optimization theory represent programs by control flow graph (CFG)

⇨ Each node in the graph represents a basic block

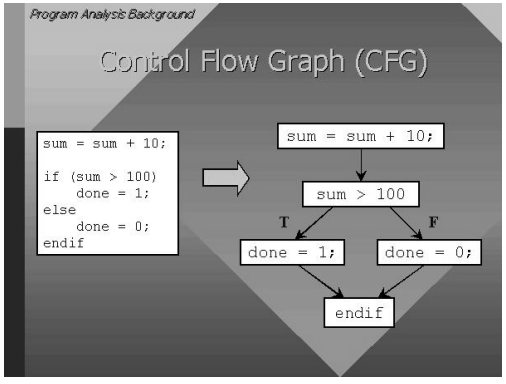
Control Flow Model II



Compilers and program optimization theory represent programs by control flow graph (CFG)

- ⇨ Each node in the graph represents a basic block
- ⇨ Basic block: straight-line piece of code without any jumps or jump targets

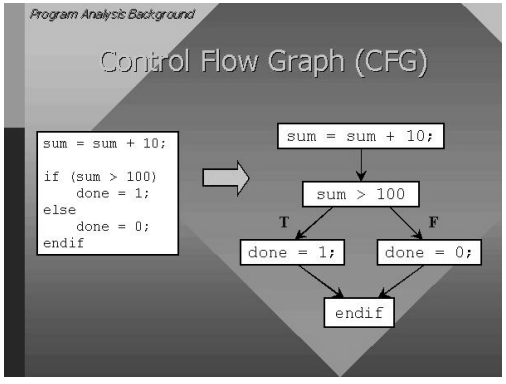
Control Flow Model II



Compilers and program optimization theory represent programs by control flow graph (CFG)

- ⇨ Each node in the graph represents a basic block
- ⇨ Basic block: straight-line piece of code without any jumps or jump targets
- ⇨ Directed edges are used to represent jumps in the control flow

Control Flow Model II



Compilers and program optimization theory represent programs by control flow graph (CFG)

- ⇨ Each node in the graph represents a basic block
- ⇨ Basic block: straight-line piece of code without any jumps or jump targets
- ⇨ Directed edges are used to represent jumps in the control flow
- ⇨ Jump targets start a block; jumps end a block

Simple transformations

So what transformations do you can suggest?

Simple transformations

So what transformations do you can suggest?

⇨ Data Obfuscation

Simple transformations

So what transformations do you can suggest?

- ⇨ Data Obfuscation
  - Variable splitting

Simple transformations

So what transformations do you can suggest?

- ⇨ Data Obfuscation
  - Variable splitting
  - Scalar/object conversion

Simple transformations

So what transformations do you can suggest?

- ⇨ Data Obfuscation
  - Variable splitting
  - Scalar/object conversion
  - Static data to procedure

Simple transformations

So what transformations do you can suggest?

- ⇨ Data Obfuscation
  - Variable splitting
  - Scalar/object conversion
  - Static data to procedure
  - Change variable lifetime

Simple transformations

So what transformations do you can suggest?

- ⇨ Data Obfuscation
  - Variable splitting
  - Scalar/object conversion
  - Static data to procedure
  - Change variable lifetime
  - Split/fold/merge arrays

Simple transformations

So what transformations do you can suggest?

- ⇨ Data Obfuscation
  - Variable splitting
  - Scalar/object conversion
  - Static data to procedure
  - Change variable lifetime
  - Split/fold/merge arrays
  - Change encoding

Simple transformations

So what transformations do you can suggest?

- ⇨ Data Obfuscation
  - Variable splitting
  - Scalar/object conversion
  - Static data to procedure
  - Change variable lifetime
  - Split/fold/merge arrays
  - Change encoding
  - Merge scalar variables

Simple transformations

So what transformations do you can suggest?

- ⇨ Data Obfuscation
  - Variable splitting
  - Scalar/object conversion
  - Static data to procedure
  - Change variable lifetime
  - Split/fold/merge arrays
  - Change encoding
  - Merge scalar variables

Simple transformations

So what transformations do you can suggest?

⇨ Data Obfuscation

- Variable splitting
- Scalar/object conversion
- Static data to procedure
- Change variable lifetime
- Split/fold/merge arrays
- Change encoding
- Merge scalar variables

⇨ Control Obfuscation

Simple transformations

So what transformations do you can suggest?

⇨ Data Obfuscation
- Variable splitting
- Scalar/object conversion
- Static data to procedure
- Change variable lifetime
- Split/fold/merge arrays
- Change encoding
- Merge scalar variables

⇨ Control Obfuscation
- Break basic blocks

Simple transformations

So what transformations do you can suggest?

⇨ Data Obfuscation
- Variable splitting
- Scalar/object conversion
- Static data to procedure
- Change variable lifetime
- Split/fold/merge arrays
- Change encoding
- Merge scalar variables

⇨ Control Obfuscation
- Break basic blocks
- Inline methods

Simple transformations

So what transformations do you can suggest?

⇨ Data Obfuscation
- Variable splitting
- Scalar/object conversion
- Static data to procedure
- Change variable lifetime
- Split/fold/merge arrays
- Change encoding
- Merge scalar variables

⇨ Control Obfuscation
- Break basic blocks
- Inline methods
- Outline statements

Simple transformations

So what transformations do you can suggest?

⇨ Data Obfuscation
- Variable splitting
- Scalar/object conversion
- Static data to procedure
- Change variable lifetime
- Split/fold/merge arrays
- Change encoding
- Merge scalar variables

⇨ Control Obfuscation
- Break basic blocks
- Inline methods
- Outline statements
- Unroll loops

Simple transformations

So what transformations do you can suggest?

⇨ Data Obfuscation
- Variable splitting
- Scalar/object conversion
- Static data to procedure
- Change variable lifetime
- Split/fold/merge arrays
- Change encoding
- Merge scalar variables

⇨ Control Obfuscation
- Break basic blocks
- Inline methods
- Outline statements
- Unroll loops
- Reorder statements

Simple transformations

So what transformations do you can suggest?

⇨ Data Obfuscation
- Variable splitting
- Scalar/object conversion
- Static data to procedure
- Change variable lifetime
- Split/fold/merge arrays
- Change encoding
- Merge scalar variables

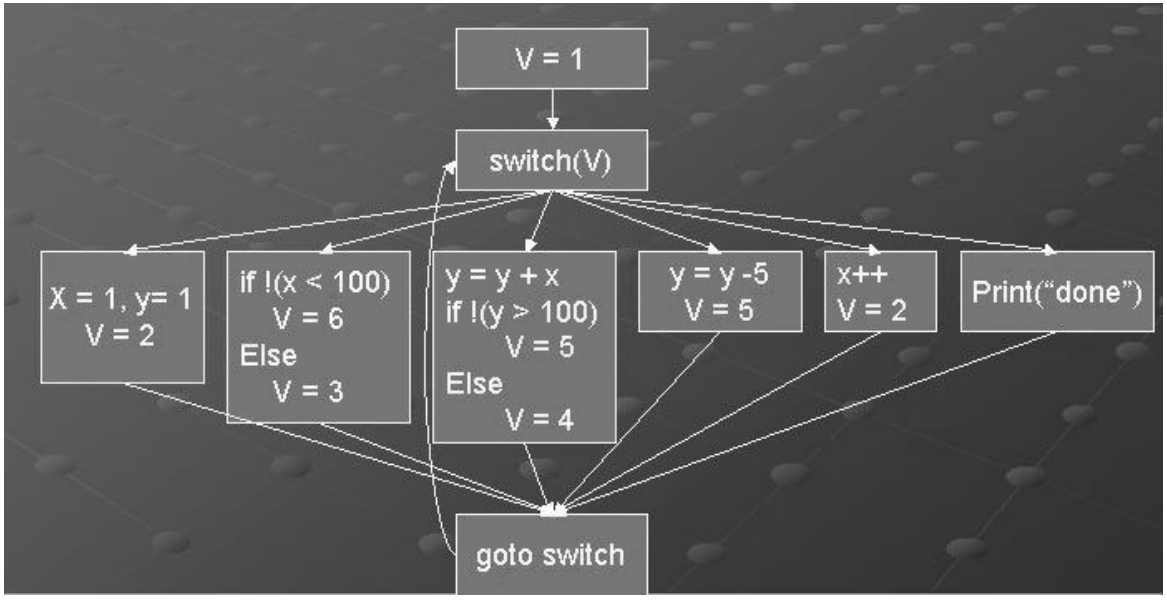
⇨ Control Obfuscation
- Break basic blocks
- Inline methods
- Outline statements
- Unroll loops
- Reorder statements
- Reorder loops

Control Flow Flattening

How to destroy control flow graph?

Control Flow Flattening

How to destroy control flow graph?

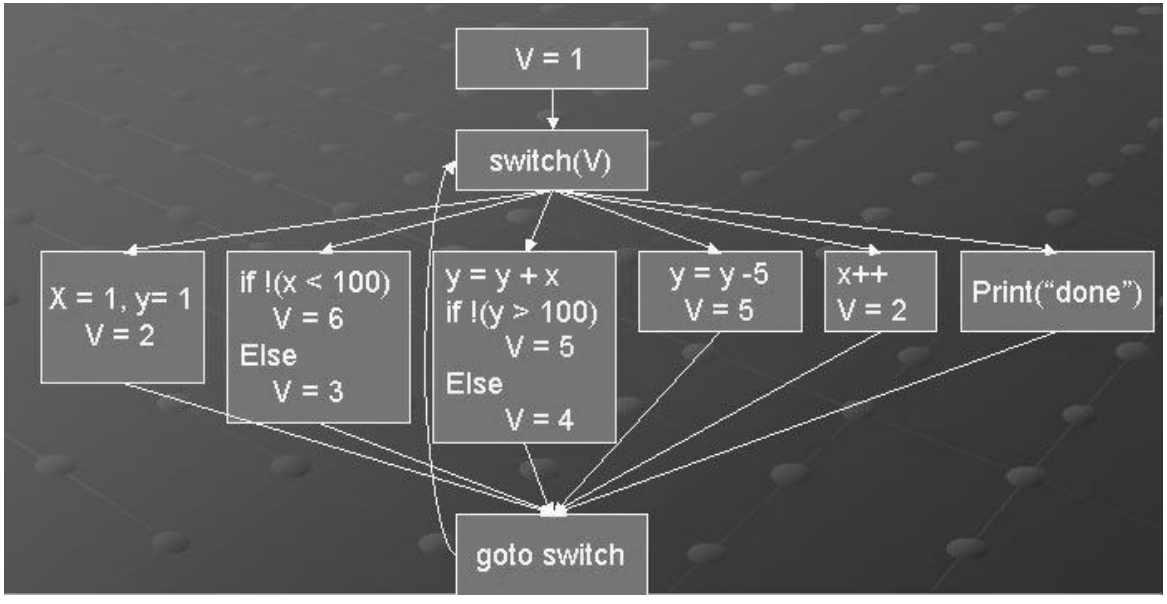


Step by step:

- ⇨ Write down a list of all basic blocks

Control Flow Flattening

How to destroy control flow graph?

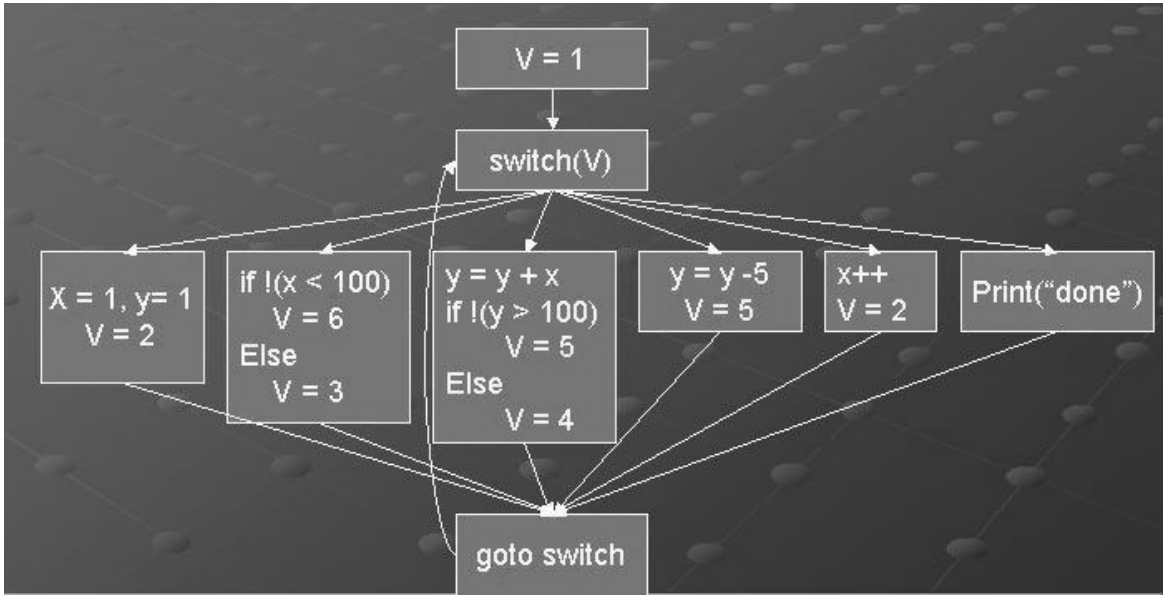


Step by step:

- ⇨ Write down a list of all basic blocks
- ⇨ Split and merge some of them

Control Flow Flattening

How to destroy control flow graph?

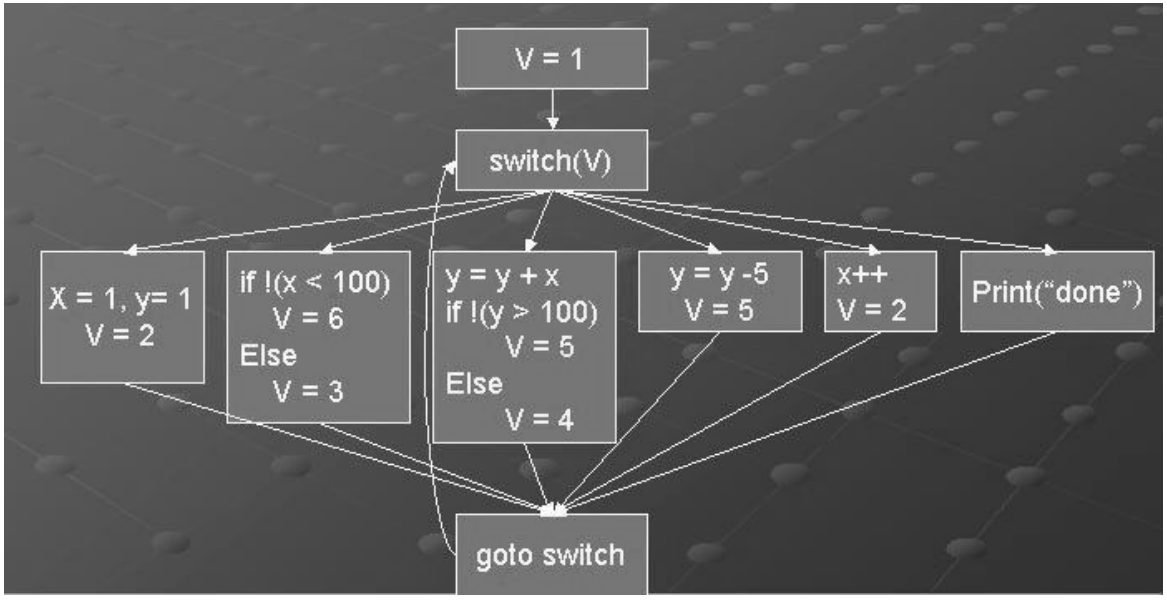


Step by step:

- ⇨ Write down a list of all basic blocks
- ⇨ Split and merge some of them
- ⇨ Enumerate them

Control Flow Flattening

How to destroy control flow graph?

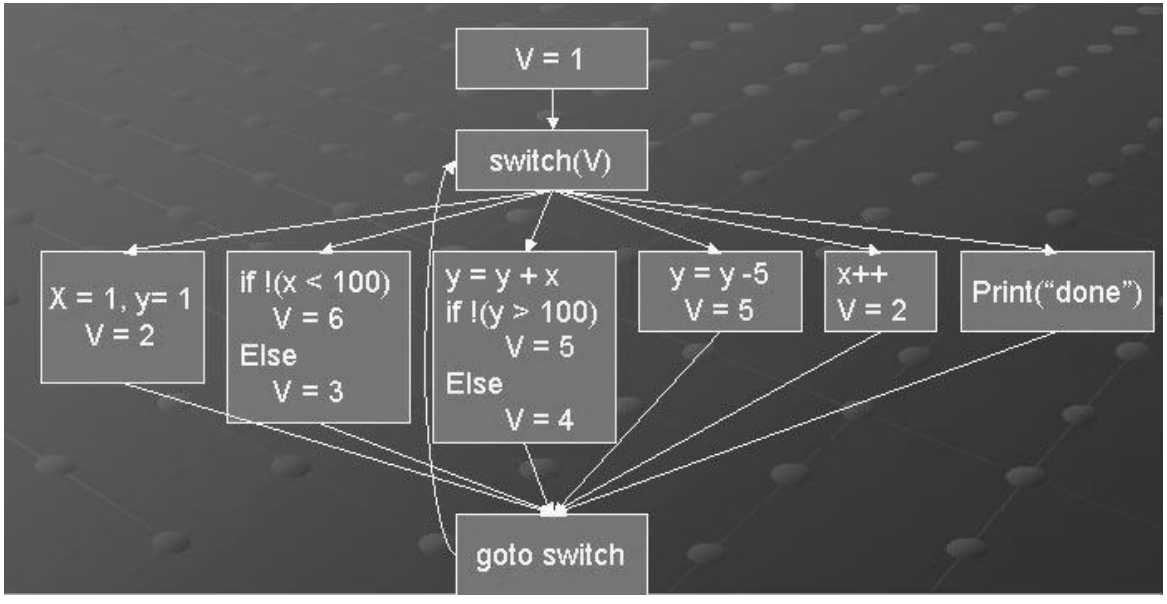


Step by step:

- ⇨ Write down a list of all basic blocks
- ⇨ Split and merge some of them
- ⇨ Enumerate them
- ⇨ Replace all calls by indirect pointing

Control Flow Flattening

How to destroy control flow graph?



Step by step:

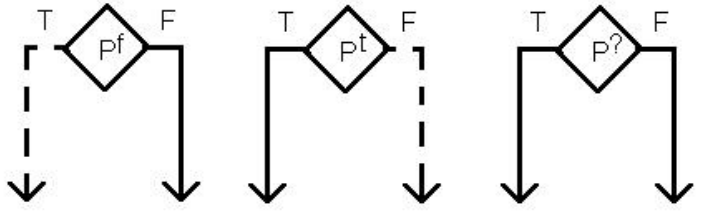
- ⇨ Write down a list of all basic blocks
- ⇨ Split and merge some of them
- ⇨ Enumerate them
- ⇨ Replace all calls by indirect pointing
- ⇨ Write a single dispatcher to maintain all control flow

Opaque predicates

How can we use IF operator for obfuscation?

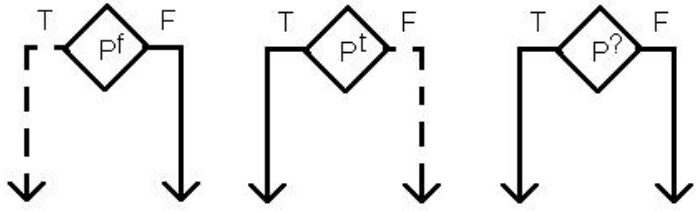
Opaque predicates

How can we use IF operator for obfuscation?



Opaque predicates

How can we use IF operator for obfuscation?

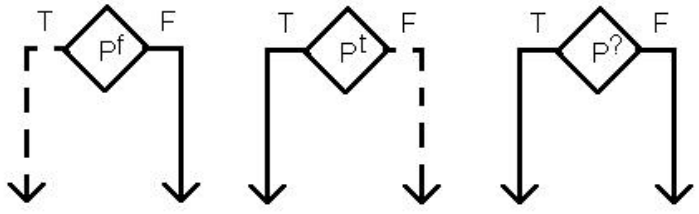


Opaque predicates: every time the same value

Difficult to discover by automatical static analysis

Opaque predicates

How can we use IF operator for obfuscation?



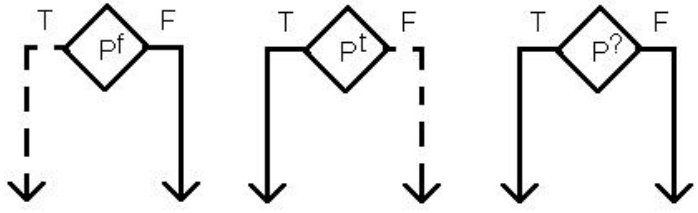
Opaque predicates: every time the same value
Difficult to discover by automatical static analysis

Examples:

$$((q + q^2) \ mod \ 2) \ = \ 0$$

Opaque predicates

How can we use IF operator for obfuscation?



Opaque predicates: every time the same value
Difficult to discover by automatical static analysis

Examples:

$$((q + q^2) \ mod \ 2) \ = \ 0$$

$$((q^3) \ mod \ 8) \ = \ 0 \ \text{OR} \ ((q^3) \ mod \ 8) \ = \ 1$$

Functions Unifying

How can we make program procedures indistinguishable?

Functions Unifying

How can we make program procedures indistinguishable?

Procedures relations are expressed by Program Call Graph: procedures are nodes and procedure calls are directed edges

Functions Unifying

How can we make program procedures indistinguishable?

Procedures relations are expressed by Program Call Graph: procedures are nodes and procedure calls are directed edges

Idea: merge functions and call universal function with additional parameter
Difficulty: different signatures (input-output specifications)

Functions Unifying

How can we make program procedures indistinguishable?

Procedures relations are expressed by Program Call Graph: procedures are nodes and procedure calls are directed edges

Idea: merge functions and call universal function with additional parameter
Difficulty: different signatures (input-output specifications)

Solution: unify signatures (in groups)

Even more transformations

Question: Can you invent more?

Even more transformations

Question: Can you invent more?

⇨ Reuse identifiers

Even more transformations

Question: Can you invent more?

- ⇨ Reuse identifiers
- ⇨ Introduce misleading comments :-)

Even more transformations

Question: Can you invent more?

- ⇨ Reuse identifiers
- ⇨ Introduce misleading comments :-)
- ⇨ Modify inheritance relations

Even more transformations

Question: Can you invent more?

- ⇨ Reuse identifiers
- ⇨ Introduce misleading comments :-)
- ⇨ Modify inheritance relations
- ⇨ Convert static data to procedural data

Even more transformations

Question: Can you invent more?

- ⇨ Reuse identifiers
- ⇨ Introduce misleading comments :-)
- ⇨ Modify inheritance relations
- ⇨ Convert static data to procedural data
- ⇨ Store part of the program as a text and interpret it only during runtime

Even more transformations

Question: Can you invent more?

- ⇨ Reuse identifiers
- ⇨ Introduce misleading comments :-)
- ⇨ Modify inheritance relations
- ⇨ Convert static data to procedural data
- ⇨ Store part of the program as a text and interpret it only during runtime
- ⇨ Remove library calls

Current Techniques: Pro and Contra

Advantages

Current Techniques: Pro and Contra

Advantages

- ✔ Easy to implement
- ✔ Universal
- ✔ Good against static analysis

Current Techniques: Pro and Contra

Advantages

- ✔ Easy to implement
- ✔ Universal
- ✔ Good against static analysis

Disadvantages

Current Techniques: Pro and Contra

Advantages

- ✔ Easy to implement
- ✔ Universal
- ✔ Good against static analysis

Disadvantages

- ✘ No guaranteed security

Current Techniques: Pro and Contra

Advantages

- ✔ Easy to implement
- ✔ Universal
- ✔ Good against static analysis

Disadvantages

- ✘ No guaranteed security
- ✘ Even no hope for that

Current Techniques: Pro and Contra

Advantages

- ✔ Easy to implement
- ✔ Universal
- ✔ Good against static analysis

Disadvantages

- ✘ No guaranteed security
- ✘ Even no hope for that
- ✘ Weak against dynamic attacks

Classical Program Analysis

What's about program analysis?

Classical Program Analysis

What's about program analysis?

Static analysis: only read code

Dynamic analysis: execute code

Classical Program Analysis

What's about program analysis?

Static analysis: only read code
Dynamic analysis: execute code

Usual tasks:

⇨ May be aliased

Classical Program Analysis

What's about program analysis?

Static analysis: only read code
Dynamic analysis: execute code

Usual tasks:

- ⇨ May be aliased
- ⇨ Must be aliased

Classical Program Analysis

What's about program analysis?

Static analysis: only read code
Dynamic analysis: execute code

Usual tasks:

- ⇨ May be aliased
- ⇨ Must be aliased
- ⇨ May be modified

Classical Program Analysis

What's about program analysis?

Static analysis: only read code
Dynamic analysis: execute code

Usual tasks:

- ⇨ May be aliased
- ⇨ Must be aliased
- ⇨ May be modified
- ⇨ Must be constant

Classical Program Analysis

What's about program analysis?

Static analysis: only read code
Dynamic analysis: execute code

Usual tasks:

- ⇨ May be aliased
- ⇨ Must be aliased
- ⇨ May be modified
- ⇨ Must be constant
- ⇨ Must be killed

Classical Program Analysis

What's about program analysis?

Static analysis: only read code
Dynamic analysis: execute code

Usual tasks:

- ⇨ May be aliased
- ⇨ Must be aliased
- ⇨ May be modified
- ⇨ Must be constant
- ⇨ Must be killed
- ⇨ Must be available

Classical Program Analysis

What's about program analysis?

Static analysis: only read code
Dynamic analysis: execute code

Usual tasks:

- ⇨ May be aliased
- ⇨ Must be aliased
- ⇨ May be modified
- ⇨ Must be constant
- ⇨ Must be killed
- ⇨ Must be available
- ⇨ May be used before kill

Classical Program Analysis

What's about program analysis?

Static analysis: only read code
Dynamic analysis: execute code

Usual tasks:

- ⇨ May be aliased
- ⇨ Must be aliased
- ⇨ May be modified
- ⇨ Must be constant
- ⇨ Must be killed
- ⇨ Must be available
- ⇨ May be used before kill
- ⇨ May be referenced

Is Deobfuscation Hard?

Can we prove the difficulty of deobfuscation?

Is Deobfuscation Hard?

Can we prove the difficulty of deobfuscation?

Is Deobfuscation Hard?

Can we prove the difficulty of deobfuscation?

Not yet. But...

Is Deobfuscation Hard?

Can we prove the difficulty of deobfuscation?

Not yet. But...

We can prove program analysis to be hard for obfuscated programs:

Is Deobfuscation Hard?

Can we prove the difficulty of deobfuscation?

Not yet. But...

We can prove program analysis to be hard for obfuscated programs:

Alias analysis of obfuscated programs is NP-hard!

Approaches to Deobfuscation

Almost all obfuscating transformations have a efficient deobfuscating method...

Approaches to Deobfuscation

Almost all obfuscating transformations have a efficient deobfuscating method...

Do you believe?

Approaches to Deobfuscation

Almost all obfuscating transformations have a efficient deobfuscating method...

Do you believe?

Deobfuscator can use:

⇨ Functions parameters catching

Approaches to Deobfuscation

Almost all obfuscating transformations have a efficient deobfuscating method...

Do you believe?

Deobfuscator can use:

- ⇨ Functions parameters catching
- ⇨ Program slicing

Approaches to Deobfuscation

Almost all obfuscating transformations have a efficient deobfuscating method...

Do you believe?

Deobfuscator can use:

- ⇨ Functions parameters catching
- ⇨ Program slicing
- ⇨ Statistical analysis

Approaches to Deobfuscation

Almost all obfuscating transformations have a efficient deobfuscating method...

Do you believe?

Deobfuscator can use:

- ⇨ Functions parameters catching
- ⇨ Program slicing
- ⇨ Statistical analysis
- ⇨ Data flow analysis

Approaches to Deobfuscation

Almost all obfuscating transformations have a efficient deobfuscating method...

Do you believe?

Deobfuscator can use:

- ⇨ Functions parameters catching
- ⇨ Program slicing
- ⇨ Statistical analysis
- ⇨ Data flow analysis
- ⇨ Pattern matching

Further Research

What can we do here?

Further Research

What can we do here?

⇨ Just new transformations

Further Research

What can we do here?

- ⇨ Just new transformations
- ⇨ Preventive transformations

Further Research

What can we do here?

- ⇨ Just new transformations
- ⇨ Preventive transformations
- ⇨ Protection against recompilation

Further Research

What can we do here?

- ⇨ Just new transformations
- ⇨ Preventive transformations
- ⇨ Protection against recompilation
- ⇨ Introducing more deobufuscation hardness results

Further Research

What can we do here?

- ⇨ Just new transformations
- ⇨ Preventive transformations
- ⇨ Protection against recompilation
- ⇨ Introducing more deobufuscation hardness results

Further Research

What can we do here?

- ⇨ Just new transformations
- ⇨ Preventive transformations
- ⇨ Protection against recompilation
- ⇨ Introducing more deobufuscation hardness results

Good Luck with this stuff!

Summary

⇨ Obfuscator analyse and modify program by series of transformations

Summary

- ⇨ Obfuscator analyse and modify program by series of transformations
- ⇨ Obfuscating transformations consist of layout, data and control tricks

Summary

- ⇨ Obfuscator analyse and modify program by series of transformations
- ⇨ Obfuscating transformations consist of layout, data and control tricks
- ⇨ Hardness of deobfuscation is not proved

Summary

- ⇨ Obfuscator analyse and modify program by series of transformations
- ⇨ Obfuscating transformations consist of layout, data and control tricks
- ⇨ Hardness of deobfuscation is not proved

Summary

- ⇨ Obfuscator analyse and modify program by series of transformations
- ⇨ Obfuscating transformations consist of layout, data and control tricks
- ⇨ Hardness of deobfuscation is not proved

Question Time!

Not covered by the talk

Obfuscation vs. watermarking
Obfuscation for watermarking
Making disassembling hard.
Decompiled – uncompilable for Java
General idea – make program dictionary as short as possible
Preventive obfuscation
Profiling in the obfuscator
Reducible and non-reducible graphs
Are obfuscating transformations comparable, e.g. one OT is every time better than another OT?
Program Analysis classification

For Further Reading

Collberg - Thomborson - Low

Series of papers

http://www.cs.arizona.edu/~collberg/research/publications/