Empirical Studies: Analysis of Obfuscation Effectiveness

Paolo Falcarin, Marco Torchiano

Mariano Ceccato, Paolo Tonella

Jasvir Nagra

Massimiliano Di Penta

Filippo Ricca

FONDAZIONE BRUNO KESSLER

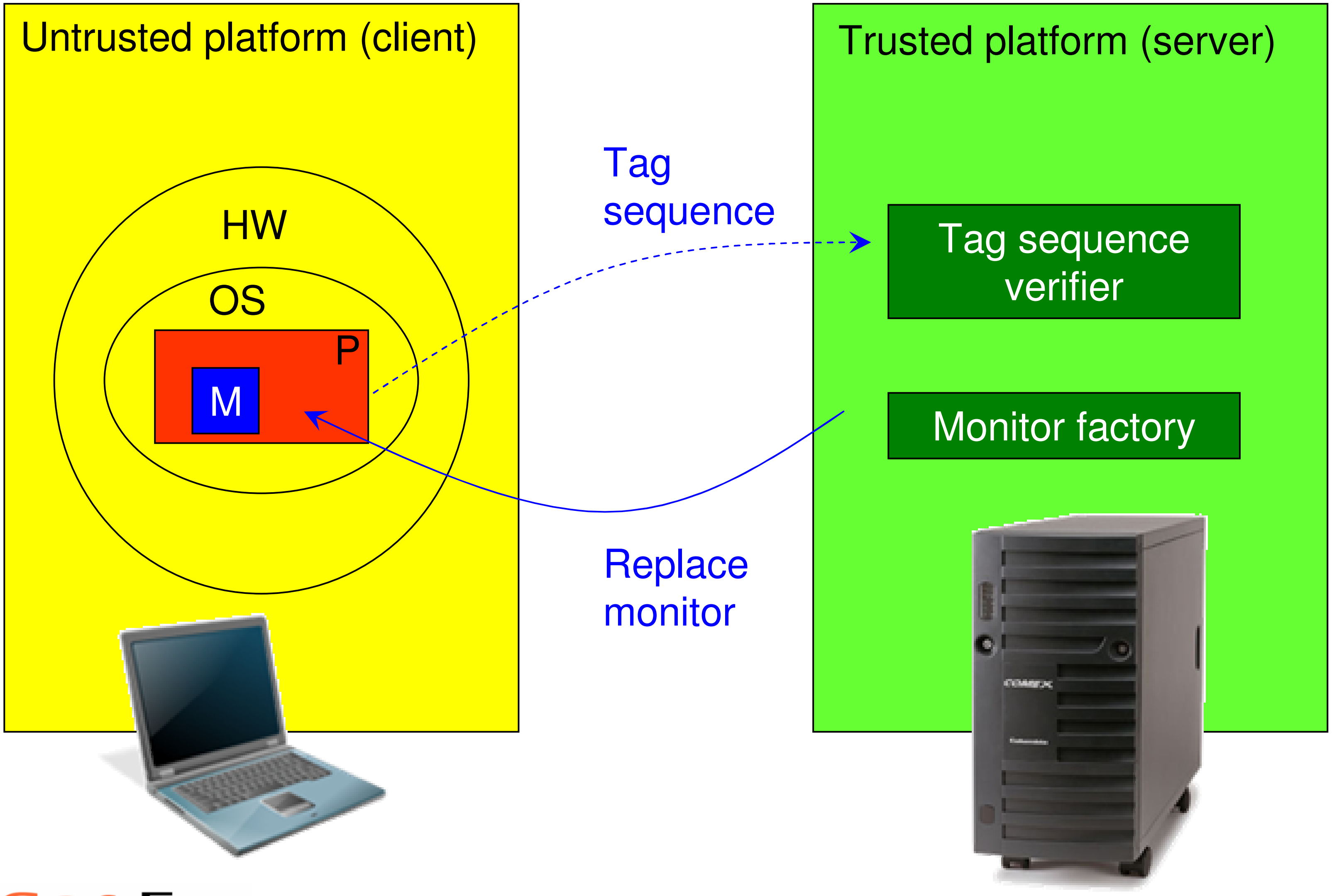
UNIVERSITY OF TRENTO - Italy

SoftEng http://softeng.polito.it

DISI

Università degli Studi del Sannio

RE-TRUST architecture



Untrusted platform (client)

Trusted platform (server)

Tag sequence

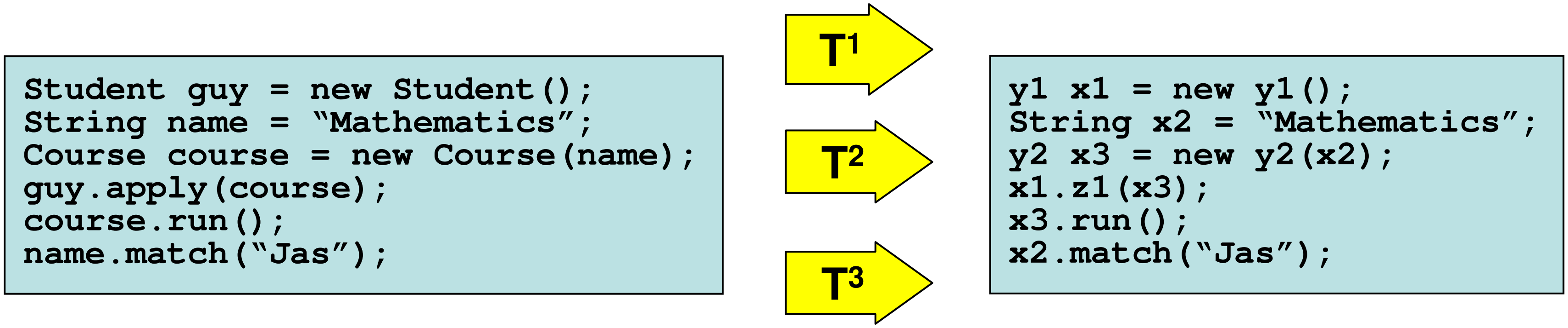
Tag sequence verifier

Monitor factory

Replace monitor

Obfuscation

- Transforming a program into an equivalent one...
- ...But Harder to reverse engineer
- How Much Harder ?



```
Student guy = new Student();
String name = "Mathematics";
Course course = new Course(name);
guy.apply(course);
course.run();
name.match("Jas");
```

T^1

T^2

T^3

```
y1 x1 = new y1();
String x2 = "Mathematics";
y2 x3 = new y2(x2);
x1.z1(x3);
x3.run();
x2.match("Jas");
```

Research Questions

RQ1: To what extent the obfuscation reduces the capability of subjects to comprehend decompiled source code?

RQ2: To what extent the obfuscation increases the time needed to perform a comprehension task?

RQ3: To what extent the obfuscation reduces the capability of subjects to perform an attack?

RQ4: To what extent the obfuscation increases the time needed to perform an attack?

Experiment Definition

Goal

To analyze the effect of source code obfuscation to evaluate its effectiveness

Quality focus

- Capability of understanding the obfuscated code.
- Capability to perform attacks on the obfuscated code

Experiment Definition

Treatments

1. Decompiled obfuscated code
2. Decompiled clear code

Dependent variables

1. Ability to perform comprehension tasks
2. Time required for comprehension
3. Ability to correctly perform an attack
4. Time required to perform an attack

Null hypotheses

H01 The obfuscation does not significantly reduce source code comprehensibility.

H02 The obfuscation does not significantly increase the time needed to perform code comprehension tasks.

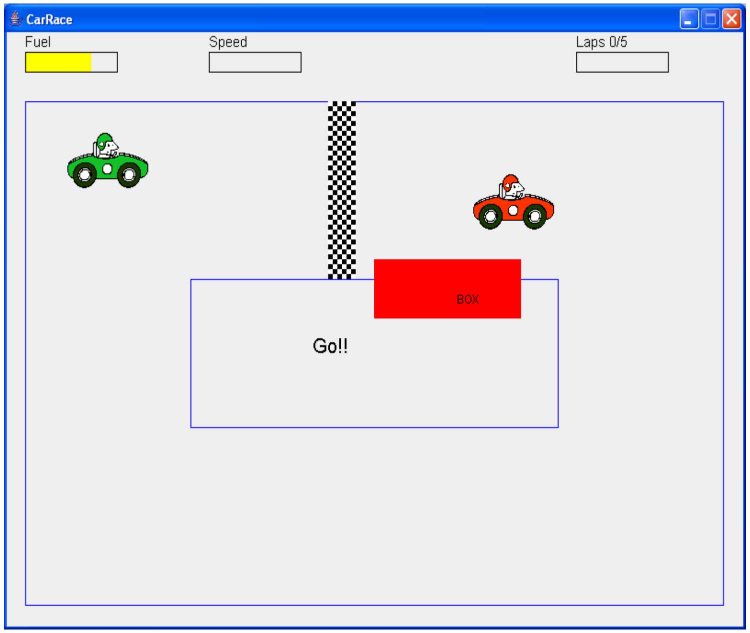
H03 The obfuscation does not significantly reduce the capability of subjects to correctly perform an attack.

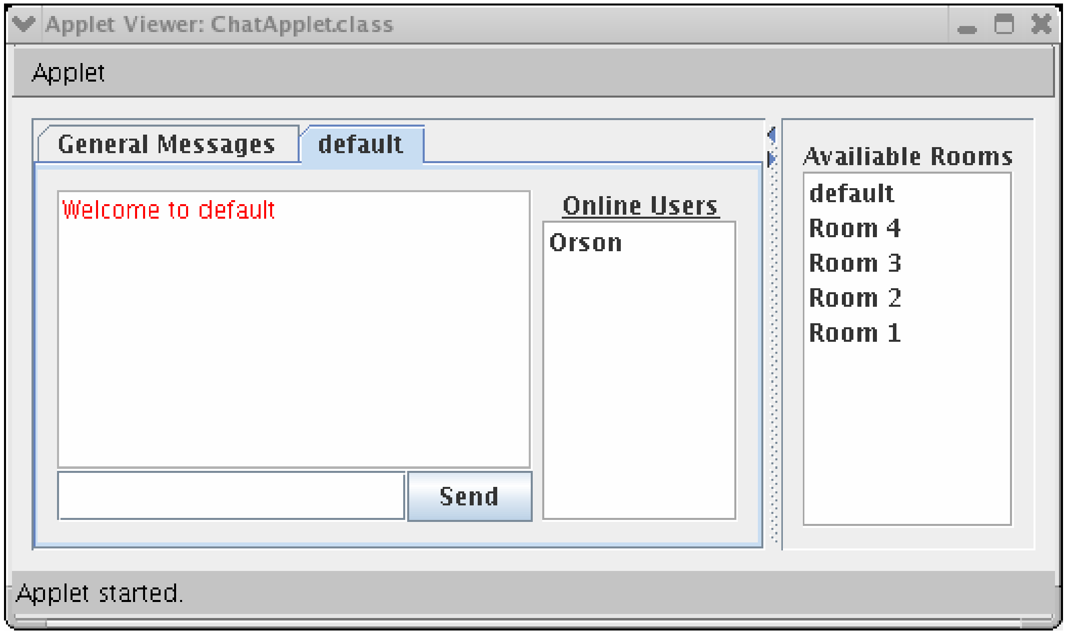
H04 The obfuscation does not significantly increase the time needed to perform an attack.

Subjects of the Experiment

- 8 Master students form the University of Trento (computer science)
- Good knowledge of Java programming
- Knowledge of software engineering topics
  - Design
  - Testing
  - Software evolution
  - Code analysis

Objects of the Experiment





Welcome to default

Orson

default
Room 4
Room 3
Room 2
Room 1

Applet started.

- Chat App: 14 classes, 1215 LOC
- Car Game: 13 classes, 1030 LOC

Experiment Balanced design

What they have

- Decompiled code
- Code browsing tools
- Debuggers
- API documentation
- Possibility to run the (modified) code

What they have to do

- Understanding tasks
- Change tasks

What to measure

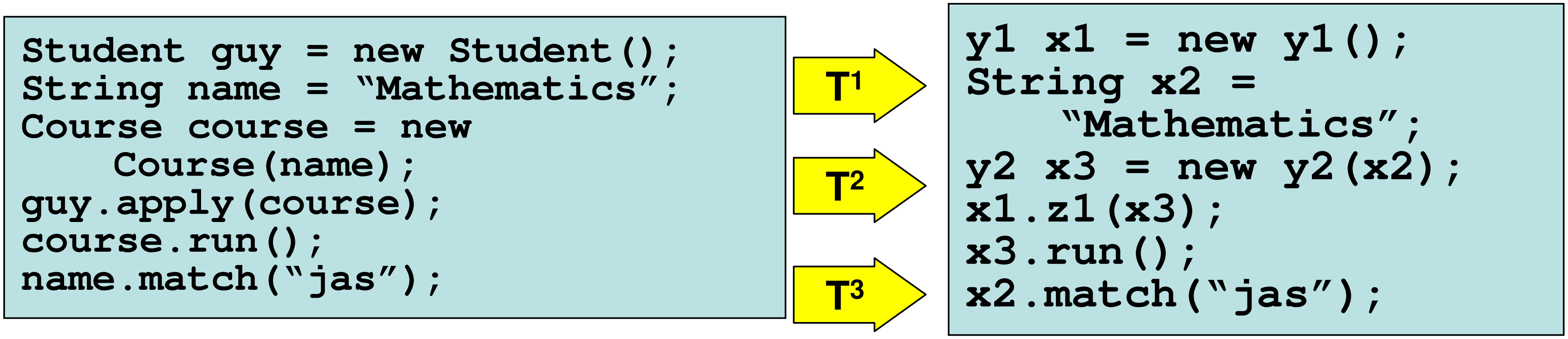
- Time/accuracy

1st session	Clear	Obfuscated
App1	**G1**	**G2**
App2	**G4**	**G3**

2nd session	Clear	Obfuscated
App1	**G3**	**G4**
App2	**G2**	**G1**

Treatment

- Identifier Renaming obfuscation
- Decompiled code
- Typical attack scenario



```
Student guy = new Student();
String name = “Mathematics”;
Course course = new
    Course(name);
guy.apply(course);
course.run();
name.match(“jas”);
```

T^1

T^2

T^3

```
y1 x1 = new y1();
String x2 =
    “Mathematics”;
y2 x3 = new y2(x2);
x1.z1(x3);
x3.run();
x2.match(“jas”);
```

Preliminary lecture

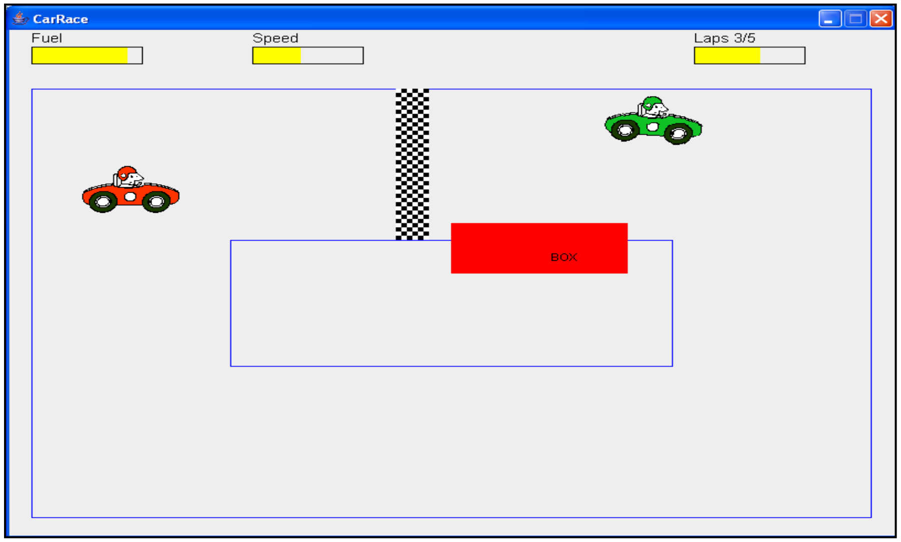
- Preliminary lecture to make the subjects aware of the experimental environment
  - IDE
  - Obfuscation
  - Debugging facilities
  - Pre questionnaire to classify expertise
  - Informed consent
  - Exercise on an application
    - To practice with the environment and mitigate the learning effect.

Experimental sessions

- Two experimental sessions
  - Description of the application
  - Either clear or obfuscated source code
  - Possibility to run the (modified) code
  - Four paper sheets (each one contains a task)
  - A post questionnaire

Kinds of attacks

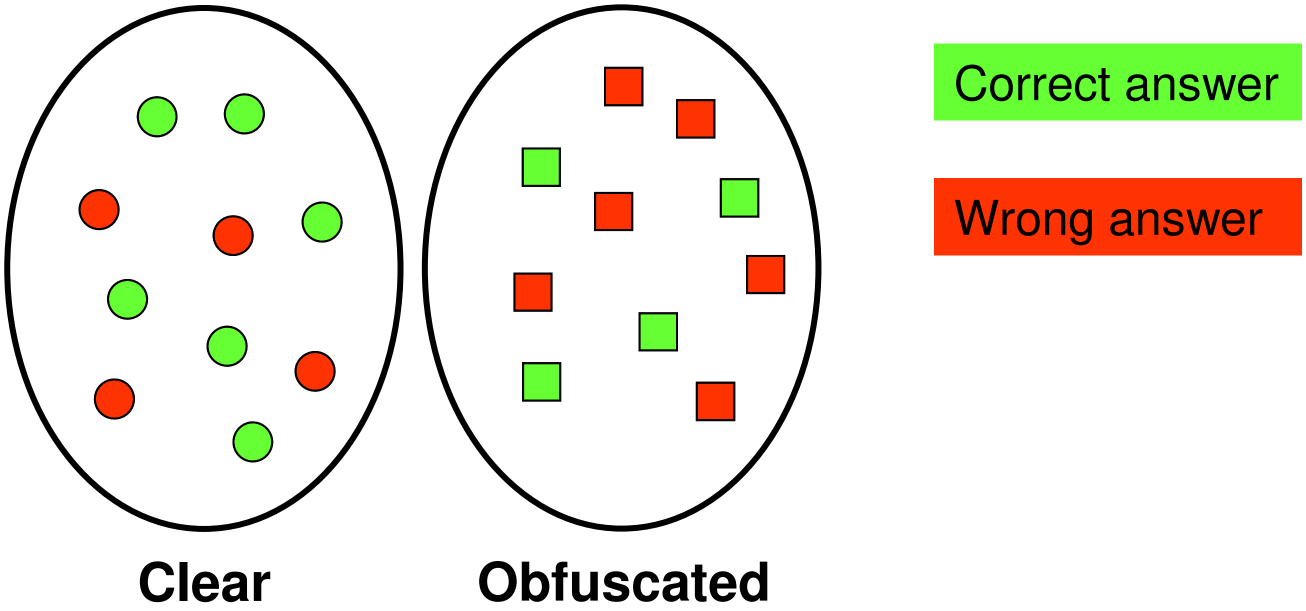
- Spotting specific functionalities
  - Observable features
- Tampering with the application
  - Make the application do something that is not available is the original code



Survey questionnaire

- Clarity of task and objective
- Difficulties experienced when performing the tasks
- Confidence in using the development environment and the debugger
- Percentage of time spent looking at the code or executing the system

Descriptive statistics



Is the proportion of correct and wrong answers statistically correlated with the treatment (obfuscation) ?

Accuracy

	Comprehension		Attack		Overall	
Treatment	Wrong	Correct	Wrong	Correct	Wrong	Correct
Clear	7	11	3	15	10	26
Obfuscated	12	8	12	8	24	16
P-value (Fisher test)	0.33		**0.009**		**0.006**	
Effect Size (Odds Ratio)	2.3		**7.1**		**3.8**	

P-value < 5% => Causal Effect

Effect > 1 => Relevant Effect

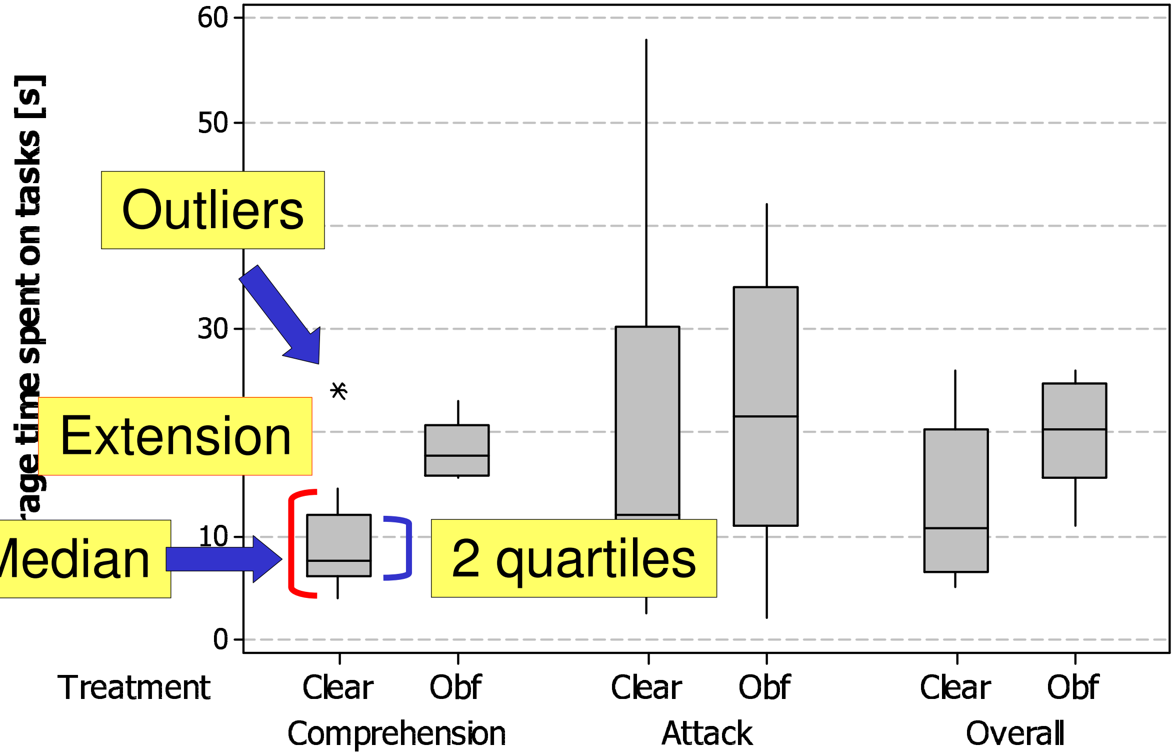
$$OR = \frac{\frac{p}{1-p}}{\frac{q}{1-q}}$$

Odd = indicate how much likely is that an event will occur as opposed to it not occurring

Time

$$d = \frac{\left(M_{obf} - M_{clear}\right)}{\sigma}$$

The Cohen *d* effect size indicates the magnitude of a main factor treatment effect on the dependent variables



Outliers

Extension

Median

2 quartiles

	Comprehension	Attack	Overall
P-value (Mann-Whitney)	**0.002**	0.19	**0.02**
Effect Size (Cohen d)	**1.8**	0.2	**1.03**

Null hypotheses

- H01 The obfuscation does not significantly reduce source code comprehensibility.

HA2 The obfuscation significantly increases the time needed to perform code comprehension tasks

Effect size = 1.8

HA3 The obfuscation significantly reduces the capability of subjects to correctly perform an attack.

Effect size = 7.1

- H04 The obfuscation does not significantly increase the time needed to perform an attack.

Threat to validity

Construct validity

- Measurements were as objective as possible
  - Comprehension tasks had only one correct solution
  - Change tasks evaluated with test cases

Internal validity

- Full factorial design with random assignments to balance individual factors and to limit learning effect

Conclusion validity

- Non parametric tests are used, we do not assume data normality

External validity

- The subject are students, only further studies can confirm that our results can be generalized to professional developers

Conclusions

- Obfuscation (Id-Renaming) thwarts reverse engineering by reducing success factor of attacks
- However it is not enough:
  - One can make a crack and spread it on the net
- Obfuscation slightly delays Comprehension
  - In RE-TRUST context we can use this Time-To-Break to define the time interval for mobile code update

Ongoing work

Consider the impact of other factors

- Subjects’ ability
- System
- Lab

Evaluate feedback after the experiment

- Clarity of objectives/tasks
- Difficulties
- Confidence with the environment
- Allocation of time code browsing/execution

Ongoing work

Torino:

- 22 PhD students
- Same obfuscation

Benevento:

- 16 master students,
- Different obfuscation techniques

What with multiple obfuscations?

References

- M. Ceccato, M. Di Penta, J. Nagra, P. Falcarin, F. Ricca, M. Torchiano, and P. Tonella.

  Towards experimental evaluation of code obfuscation techniques. In *Proc. of the 4th Workshop on Quality of Protection. ACM, Oct 2008 (to appear).*

- M. Ceccato, M. Di Penta, J. Nagra, P. Falcarin, F. Ricca, M. Torchiano, and P. Tonella.

  The effectiveness of source code obfuscation: an experimental assessment. Technical report, University of Sannio, Dept. of Engineering— sep 2008. *http://www.rcost.unisannio.it/mdipenta/icse09-tr.pdf*

Questions

- How to plan next experiments?
- Which other factors to take into account?
- Which Sw Metrics to use as parameters to estimate Sw intrinsic complexity?

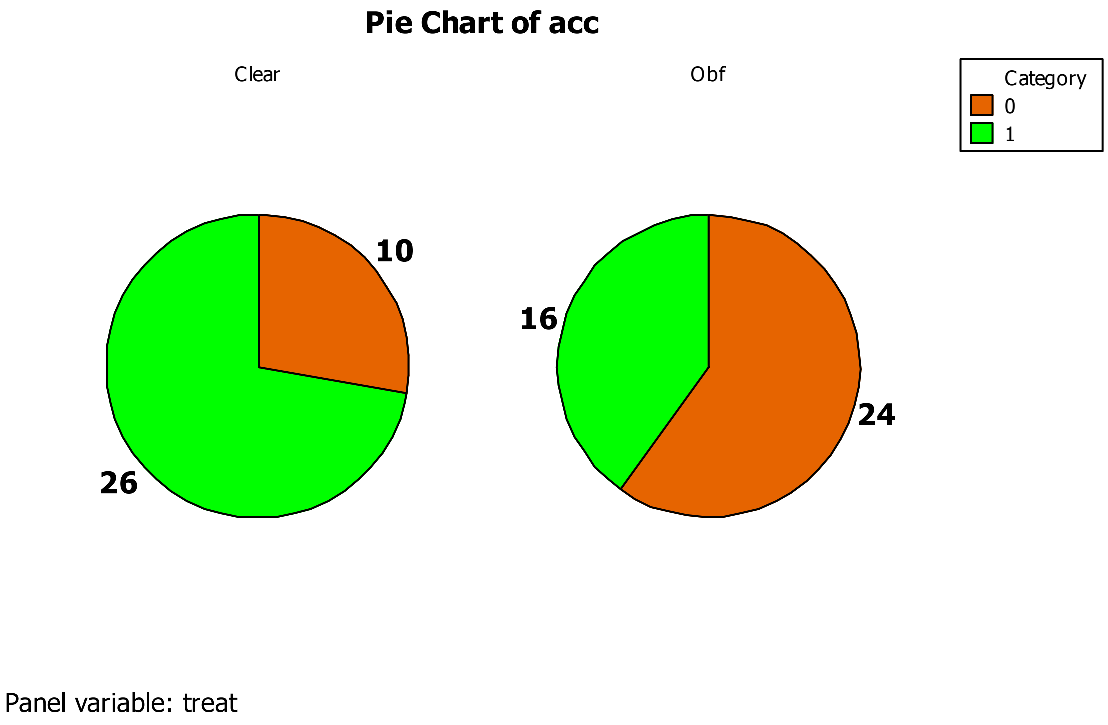
Metrics on Obfuscation

- Collberg et al. proposed the use of complexity measures (e.g. **potency**) in obfuscator tools to help developers choosing among different obfuscation transformations.
- Udupa et al. used the amount of time required to perform automatic de-obfuscation to evaluate the usefulness of **control-flow flattening** obfuscation
- Goto et al. proposed the **depth of parse tree** to measure source code complexity;
- Linn: **confusion factor** = percentage of instructions not correctly disassembled (binary obfuscation)
- Anckaert et al. attempted at quantifying and comparing the level of protection of different obfuscation techniques.
  - Provide a series of metrics based on code, control flow, data and data flow: clear and obfuscated source code.

Related Work

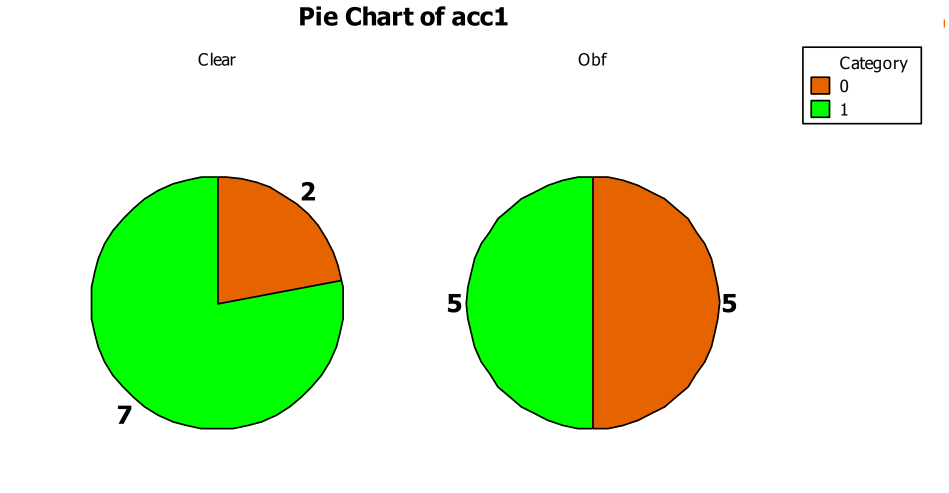
- I. Sutherland, G. E. Kalb, A. Blyth, and G. Mulley. An empirical examination of the reverse engineering process for binary files. *Computers & Security, 25(3):221-228, 2006.*
- They evaluate complexity of reverse engineering binary code by asking a group of 10 students (of heterogeneous level of experience) to perform static analysis, dynamic analysis and change tasks on several C (compiled) programs.
- They found that the subjects' ability was significantly correlated with the success of reverse engineering tasks they had to perform.

Accuracy



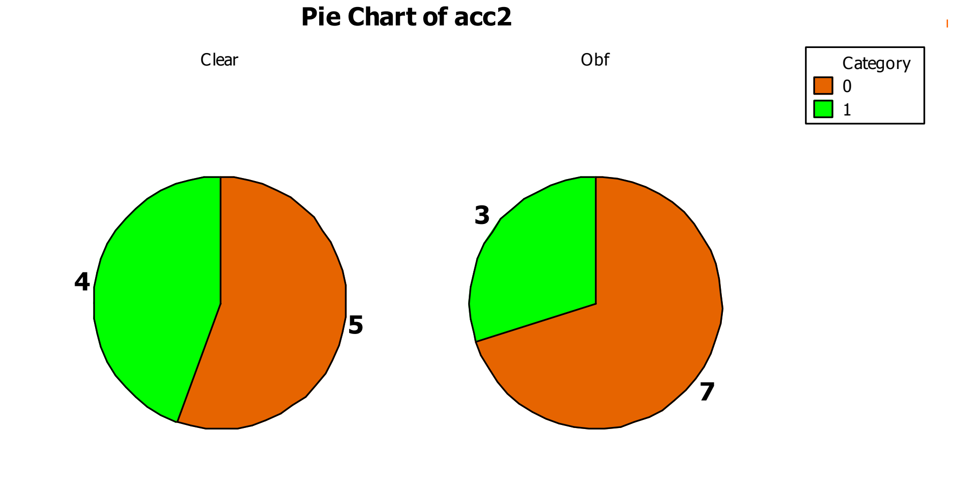
Fisher test
p-value = 0.005977
odds ratio = 0.2613782

Accuracy by Task



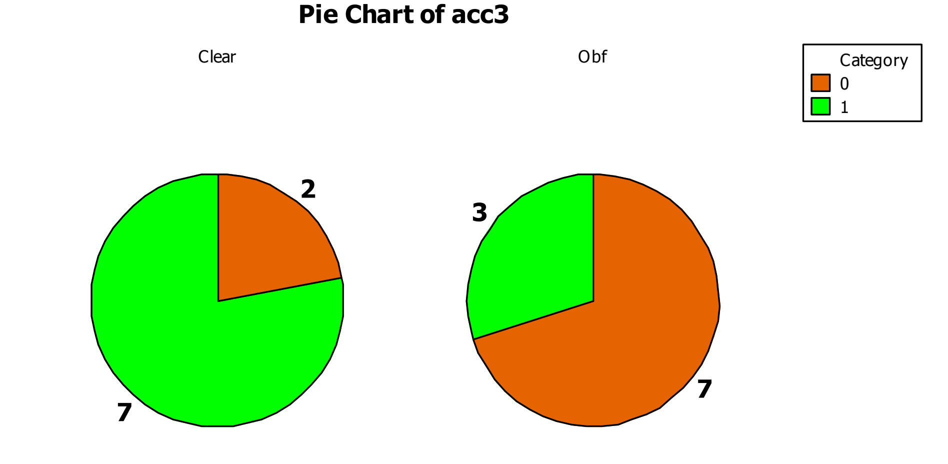
Panel variable: treat

Fisher test
p-value = 0.3498
odds ratio = 0.3059173



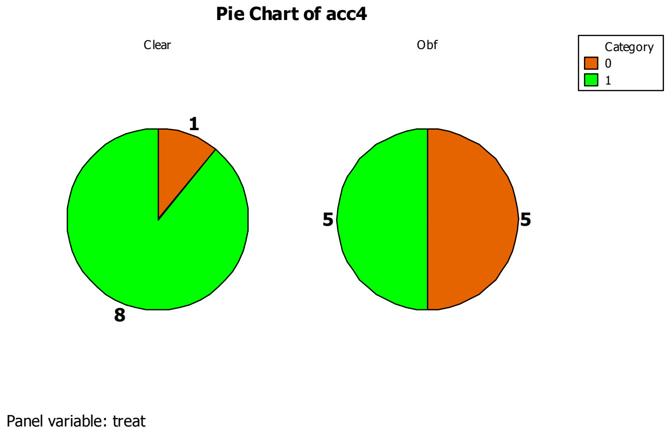
Panel variable: treat

Fisher test
p-value = 0.6499
odds ratio = 0.5539091



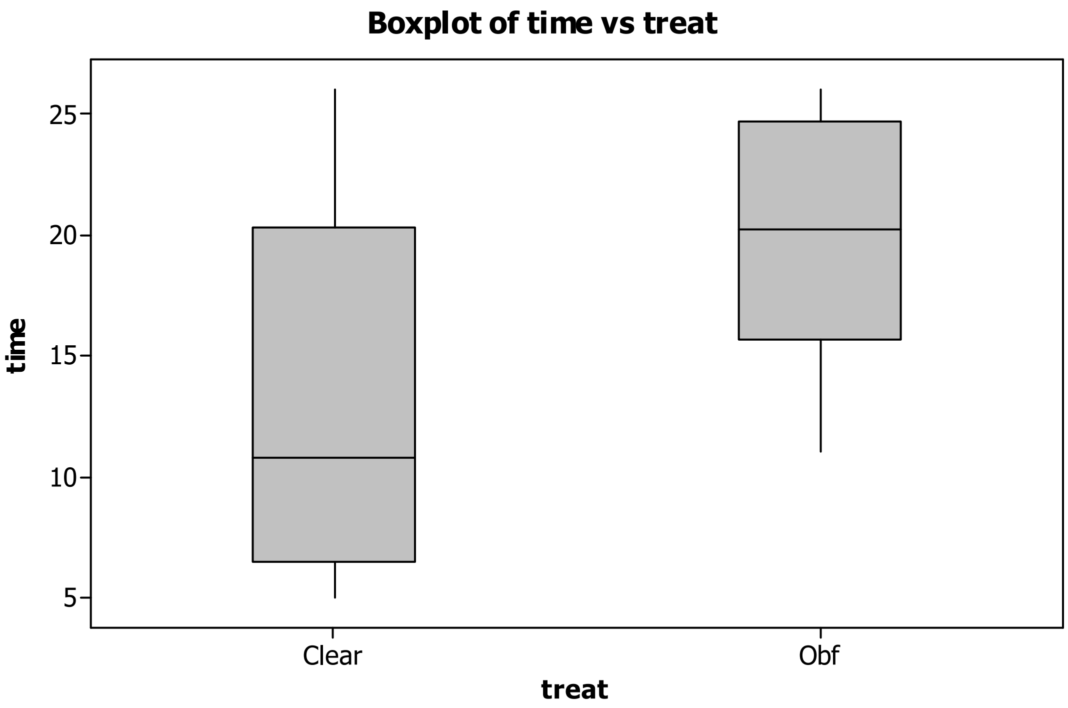
Panel variable: treat

Fisher test
p-value = 0.06978
odds ratio = 0.1395424



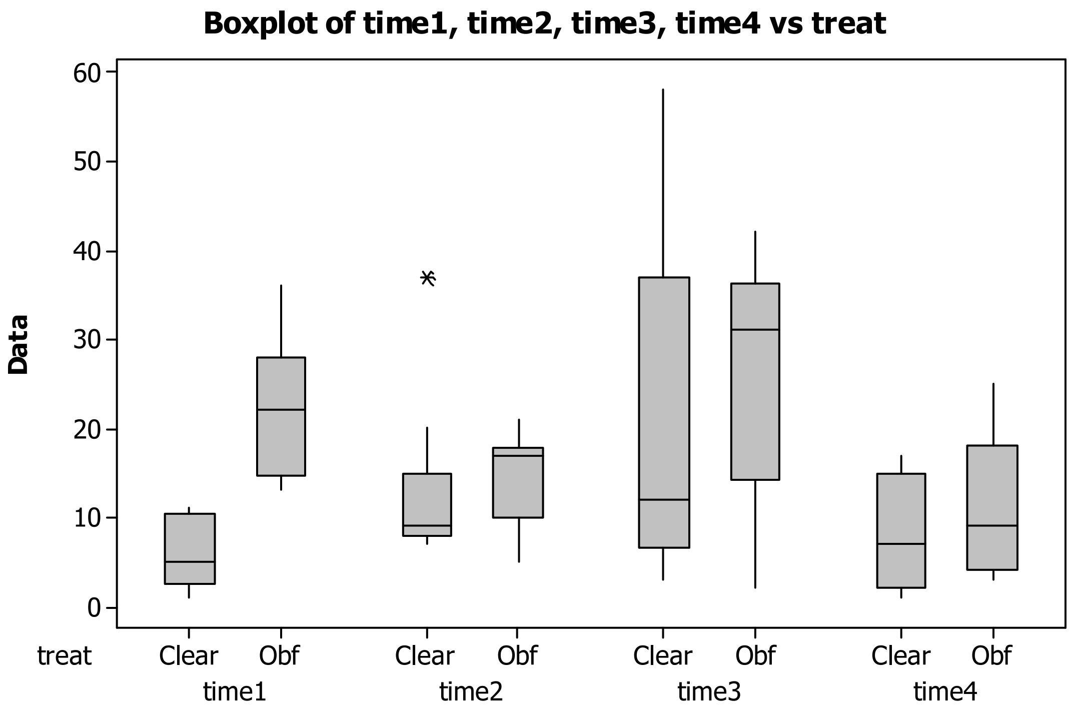
Fisher test
p-value = 0.1409
odds ratio = 0.1399176

Time



Wilcox test unpaired one-tailed
P-value 0.02487

Time by task



Wilcox test unpaired one-tailed
P-value t_1:0.0001373 t_2:0.1421 t_3:0.1733 t_4:0.3418

Obfuscation

- Preserve same functionality of original program
- Maximize obscurity
  - More time consuming to reverse engineer
  - More difficult to use automated tools
  - Minimum overhead
- Obfuscation makes reverse engineering difficult

Authors Affiliations

P. Falcarin, M. Torchiano

POLITECNICO DI TORINO 1859 1906

SOftEng http://softeng.polito.it

M. Ceccato, P. Tonella

FONDAZIONE BRUNO KESSLER

J. Nagra

UNIVERSITY OF TRENTO

M. Di Penta

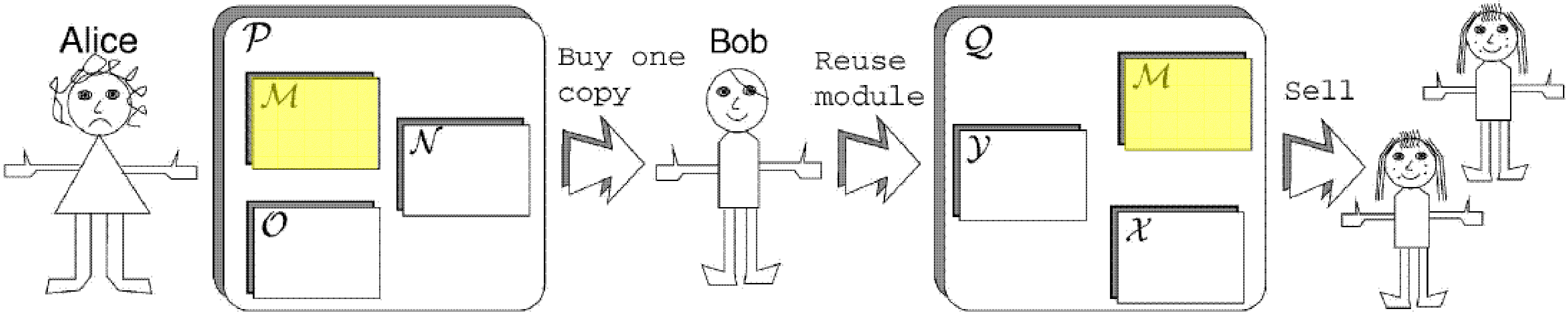
Università degli Studi del Sannio

F. Ricca

DISI

Malicious reverse engineering

- Valuable piece of code is extracted from an application and incorporated into competitor’s code.
- Illegal Software Reuse (a.k.a. piracy)


Alice
𝒫
ℳ
𝒩
𝒪
Buy one copy
Bob
Reuse module
𝒬
ℳ
𝒴
𝒳
Sell