Programming Experience Might Not Help in Comprehending Obfuscated Source Code Efficiently

Norman Hänsch[1], Andrea Schankin[2], Mykolai Protsenko[3]
Felix Freiling[1], Zinaida Benenson[1]

[1]Friedrich-Alexander-Universität Erlangen-Nürnberg
[2]Karlsruhe Institute of Technology
[3]Fraunhofer Institute for Applied and Integrated Security

August 14, 2018

FAU

KIT
Karlsruher Institut für Technologie

Fraunhofer
AISEC

Background - Software Obfuscation

- Software obfuscation: protect programs by making them harder to understand

Background - Software Obfuscation

- Software obfuscation: protect programs by making them harder to understand

  Who obfuscates? Why?

-

Background - Software Obfuscation

- Software obfuscation: protect programs by making them harder to understand
-

Who obfuscates?	Why?
Software vendors	secure their intellectual property
Hackers	make their malicious code harder to understand

Background - Software Obfuscation

- Software obfuscation: protect programs by making them harder to understand
-
Who obfuscates?	Why?
Software vendors	secure their intellectual property
Hackers	make their malicious code harder to understand

- Collberg et al. (Technical Report, 1997):
  - Potency: To what degree is a human reader confused?
  - Resilience: How well an obfuscation methods up under an attack from an automatic deobfuscator?

Background - Software Obfuscation

- Software obfuscation: protect programs by making them harder to understand

Who obfuscates?	Why?
Software vendors	secure their intellectual property
Hackers	make their malicious code harder to understand

- Collberg et al. (Technical Report, 1997):
  - Potency: To what degree is a human reader confused?
  - Resilience: How well an obfuscation methods up under an attack from an automatic deobfuscator?
  - Evaluation based on software metrics

Background - Software Obfuscation

- Software obfuscation: protect programs by making them harder to understand

Who obfuscates?	Why?
Software vendors	secure their intellectual property
Hackers	make their malicious code harder to understand

- Collberg et al. (Technical Report, 1997):
  - Potency: To what degree is a human reader confused?
  - Resilience: How well an obfuscation methods up under an attack from an automatic deobfuscator?
  - Evaluation based on software metrics
- Ceccato et al. (Empirical Software Engineering, 2014): user studies

The Study - Replication and Novelty

Replication
- Study materials
- Questionnaire
- Code understanding (correctness, time, efficiency)
- Obfuscation methods: Name Overloading (NO), Opaque Predicates (OP)

The Study - Replication and Novelty

Replication	Study materials Questionnaire Code understanding (correctness, time, efficiency) Obfuscation methods: Name Overloading (NO), Opaque Predicates (OP)
Novelty	Code analysis behavior (actions & time spent on them) The influence of experience Evaluation of correctness of the answers Study design

The Study - The Programs

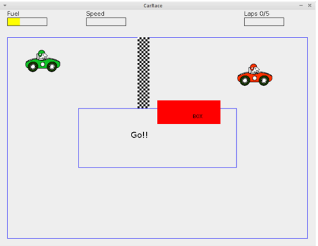

Figure: Race Program

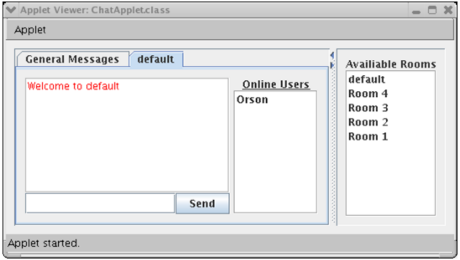

Figure: Chat Program

The Study - Code Examples

Listing 1: Code obfuscated with Name Overloading (NO)

```
public void __m1(int i)
{
    if(__f22)
        if(__f19 == 0)
        {
            __f5 += i;
            if(__f5 > __f6 / 10)
                __f5 = __f6 / 10;
            else
            if(__f5 < __f7 / 10)
                __f5 = __f7 / 10;
        } else [...]
```

The Study - Code Examples

Listing 2: Clear code from Race MovingCarModel.java

```
public void changeSpeed(int i)
{
    if(started)
        if(gas == 0)
        {
            speed += i;
            if(speed > maxSpeed / 10)
                speed = maxSpeed / 10;
            else
            if(speed < minSpeed / 10)
                speed = minSpeed / 10;
        } else [...]
```

The Study - Code Examples

Listing 3: Code obfuscated with Opaque Predicates (OP)

```
public void changeSpeed(int i) {
  if (Node.getI() != Node.getH()) {
    lastFuel = (0L + time2) - (long) lap;
    started = lastFuel == 0L;
    Node.getF().setLeft(Node.getH().getLeft());
  } else {
    Node.getG().getLeft().swap(
        Node.getG().getRight());
    if (started)
      if (Node.getI() == Node.getH()) {
        if (gas == 0) {
          if (Node.getF() == Node.getG()) {
            Node.getF().setLeft(
                Node.getI().getRight()); [...]
```

The Study - Study Design

Group	**1st Program (clear code)**	**2nd Program (obfuscated)**
1	Race: *Rnd*(Box,Laps)	NO(Chat): *Rnd*(Messages,Users)
2	Race: *Rnd*(Box,Laps)	OP(Chat): *Rnd*(Messages,Users)
3	Chat: *Rnd*(Messages,Users)	NO(Race): *Rnd*(Box,Laps)
4	Chat: *Rnd*(Messages,Users)	OP(Race): *Rnd*(Box,Laps)

The Study - Demographics of the Participants

- 66 participants
  - 44 bachelor students
  - 20 master students
  - 2 PhD students
- 24.2% already participated in a course related to software obfuscation

Results - Code Comprehension

	Clear vs NO
Correctness	-0.113
Efficiency	-0.312*
Total time	0.338**
Time correct	0.351*

Table: Wilcoxon & Mann-Whitney-U tests; $*p < .05$, $**p < .01$
Effect sizes r (no effect, small, medium)

Results - Code Comprehension

	Clear vs NO	*Clear vs OP*
Correctness	-0.113	-0.154
Efficiency	-0.312*	-0.332**
Total time	0.338**	0.276*
Time correct	0.351*	0.193

Table: Wilcoxon & Mann-Whitney-U tests; *$p < .05$, **$p < .01$
Effect sizes r (no effect, small, medium)

Results - Code Comprehension

	Clear vs NO	*Clear vs OP*	*NO vs OP*
Correctness	-0.113	-0.154	-0.079
Efficiency	-0.312*	-0.332**	0.045
Total time	0.338**	0.276*	-0.156
Time correct	0.351*	0.193	-0.260

Table: Wilcoxon & Mann-Whitney-U tests; *$p < .05$, **$p < .01$
Effect sizes r (no effect, small, medium)

Results - Behavior

	Clear vs NO
Number of:	
File open commands	0.451**
Advanced commands	0.352**
Program executions	0.243
Debugging mode	0.420**
Time spent on:	
Program executions	0.290*
Debugging mode	0.433**
Code reading	0.080

Table: Wilcoxon & Mann-Whitney-U tests; $*p < .05$, $**p < .01$
Effect sizes r (no effect, small, medium)

Results - Behavior

	Clear vs NO	*Clear vs OP*
Number of:		
File open commands	0.451**	0.058
Advanced commands	0.352**	0.282*
Program executions	0.243	0.356**
Debugging mode	0.420**	0.349**
Time spent on:		
Program executions	0.290*	0.278*
Debugging mode	0.433**	0.308*
Code reading	0.080	0.016

Table: Wilcoxon & Mann-Whitney-U tests; $*p < .05$, $**p < .01$
Effect sizes r (no effect, small, medium)

Results - Behavior

	Clear vs NO	*Clear vs OP*	*NO vs OP*
Number of:			
File open commands	0.451**	0.058	-0.373**
Advanced commands	0.352**	0.282*	-0.106
Program executions	0.243	0.356**	-0.104
Debugging mode	0.420**	0.349**	0.030
Time spent on:			
Program executions	0.290*	0.278*	-0.014
Debugging mode	0.433**	0.308*	-0.079
Code reading	0.080	0.016	-0.081

Table: Wilcoxon & Mann-Whitney-U tests; $*p < .05$, $**p < .01$
Effect sizes r (no effect, small, medium)

The Study - Experience

- Survey questions
  - *Programming Experience*: quality and type of code written so far
  - *Obfuscation Experience*: experience with obfuscation and debugging
  - *Java Experience*: experience with Java and using Eclipse
- Experiment
  - *Comprehension Skills*: efficiency in working on clear code

The Study - Experience

- Survey questions
  - *Programming Experience*: quality and type of code written so far
  - *Obfuscation Experience*: experience with obfuscation and debugging
  - *Java Experience*: experience with Java and using Eclipse
- Experiment
  - *Comprehension Skills*: efficiency in working on clear code
- k-means cluster analysis:
  - 21 beginners
  - 45 experienced participants

Results - Experience

- Experience leads to significant differences concerning:

ω^2	p	Measurement
0.16	**	Correctness
0.10	**	Efficiency
0.13	**	Advanced commands
0.13	**	Debugging mode
0.05	*	Time spent debugging

Table: ANOVA; $*p < .05$, $**p < .01$;
Effect size ω^2 (small, medium, large effect)

Results - Exp. x Obf.: Behavior

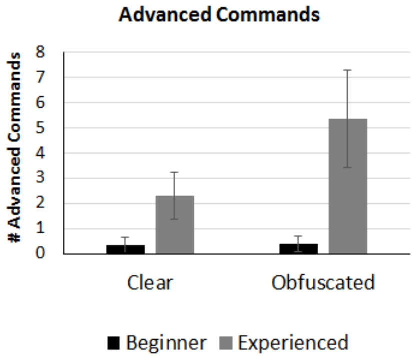

Figure: $\omega^2 = 0.06*$ significant difference (ANOVA; $*p < .05$, $**p < .01$.)

Results - Exp. x Obf.: Behavior

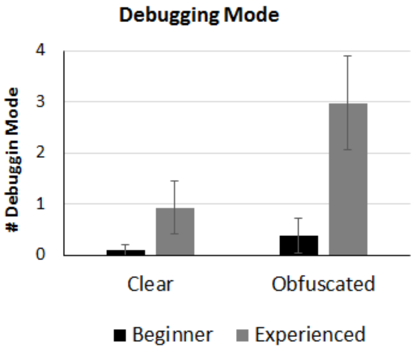

Figure: $\omega^2 = 0.09^{**}$ significant difference (ANOVA; $^*p < .05$, $^{**}p < .01$.)

Results - Exp. x Obf.: Code Comprehension

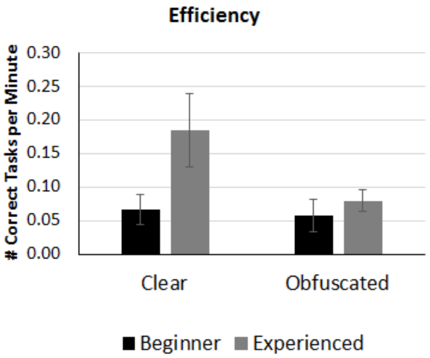

Figure: $\omega^2 = 0.01$* significant difference (ANOVA; *$p < .05$, **$p < .01$.)

Summary

1. Confirmation of all findings by Ceccato et al.
2. Empirical support of the taxonomy of Collberg et al.
3. Code comprehension behavior on obfuscated software may be different from comprehension on traditional programs.
4. Programming experience might not help in comprehending obfuscated source code efficiently.