Evaluation of Software Protection

Paolo Falcarin

Ca' Foscari University of Venice

Department of Enviromental Sciences, Informatics and Statistics

paolo.falcarin@unive.it

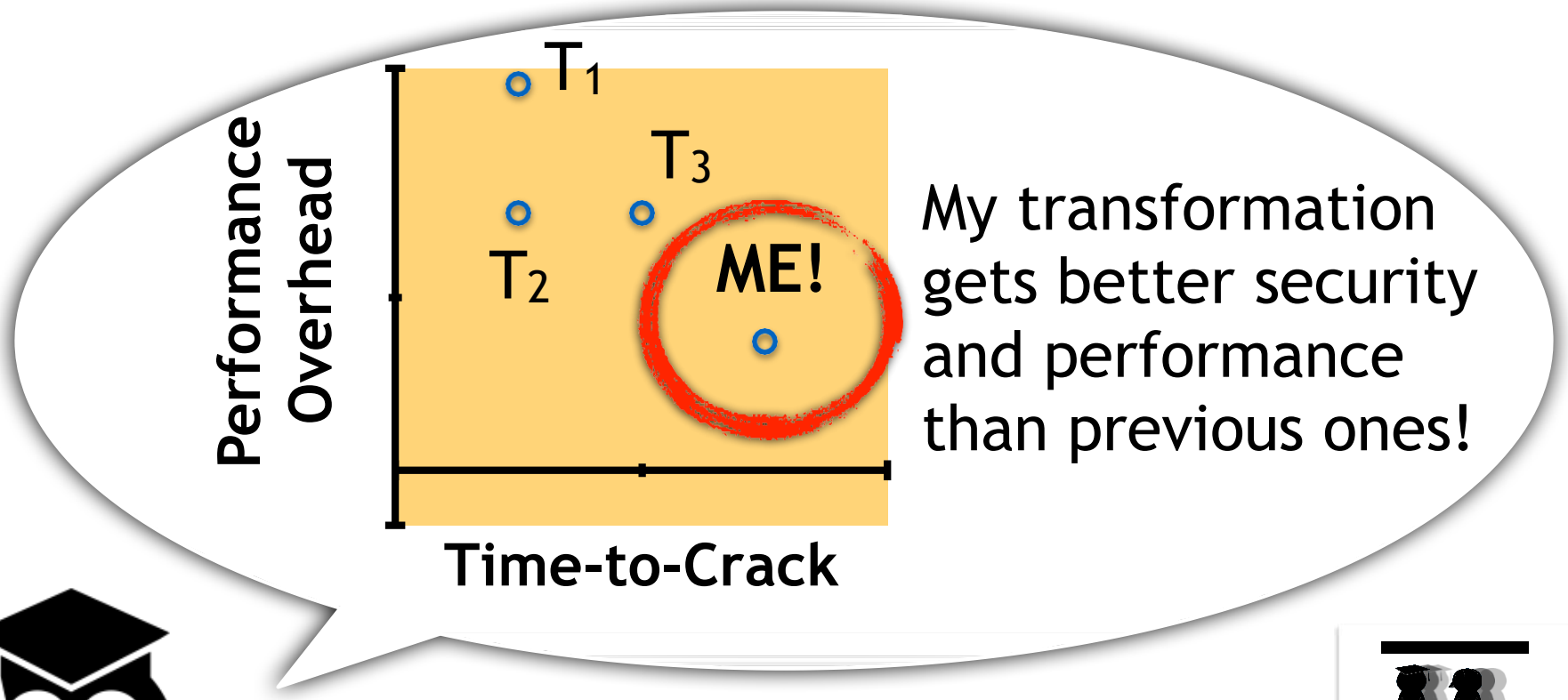
CM0626 – Software Security

CM0631-2 – Software Security

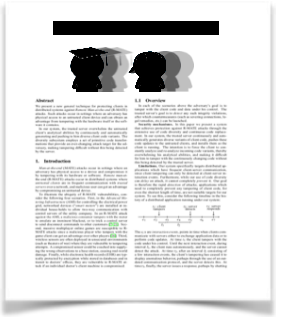
1 April 2025

Evaluation in Academia

Ca' Foscari University of Venice



The basics of research progress



Evaluation in Industry

Ca' Foscari University of Venice

- Professional **Tiger teams** evaluate new systems and transformations
  - *Tiger Team's purpose is to penetrate security, and thus test security measures.*

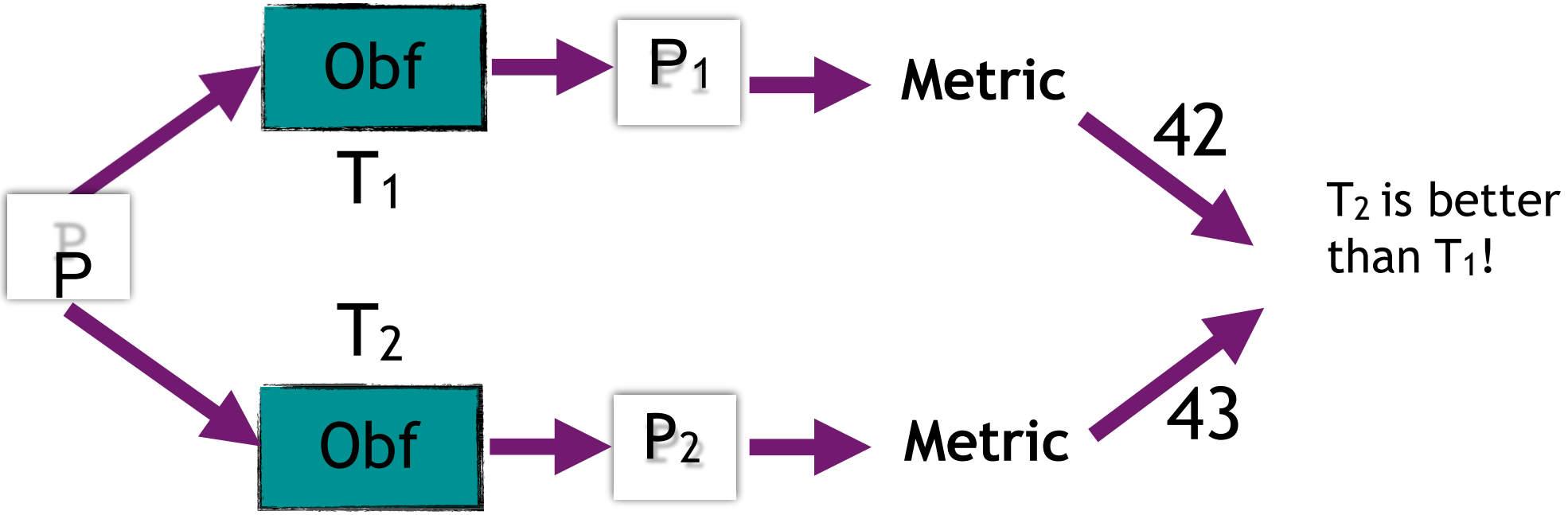
- Experience from monitoring real world adversaries

Transformation	Status
T_1	Broken in 2009
T_2	Soon to be broken
T_3	Works for now

Programmatic Evaluation

Ca' Foscari University of Venice

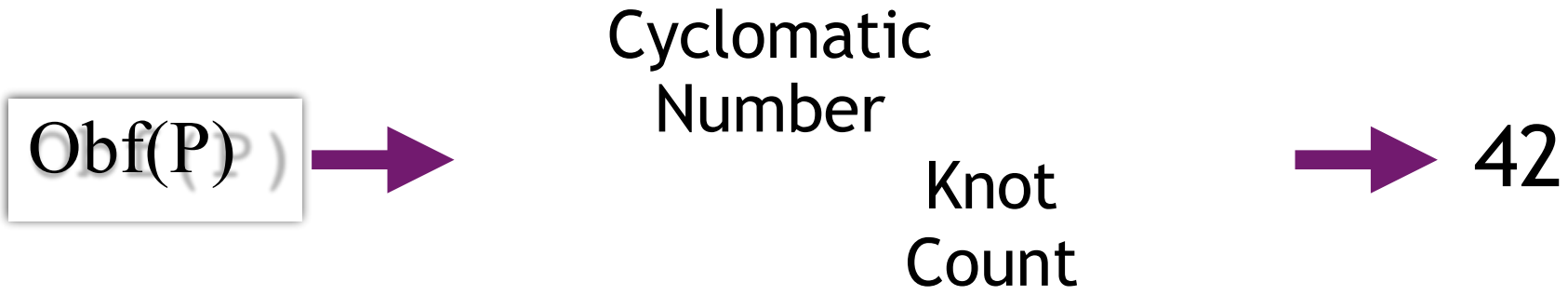
- Invent substitutes for red team evaluation
- Which metrics should we use?



Metric 1: SW Complexity Metrics

Ca' Foscari University of Venice

- Combine a few Software Complexity Metrics
- Issues:
  - SCMs were not designed to measure code badness;
  - Hundreds of SCMs - which ones should we use?



Metric 1: SW Complexity Metrics

Ca' Foscari University of Venice

- Knot Count:
  - Number of crossings of control flow arrows in a graph
- Cyclomatic number:
- Number of decision points:

  #edges - #nodes+2*(#connected components)

1. Anckaert, et al., Program Obfuscation: A Quantitative Approach
2. Ceccato, M., Capiluppi, A., Falcarin, P., & Boldyreff, C. (2015). A large study on the effect of code obfuscation on the quality of java code. *Empirical Software Engineering*, *20*(6), 1486-1524.

Metric 2: Resilience to Analysis Tools

Ca' Foscari University of Venice

- Measure the runtime and precision of code analysis tool

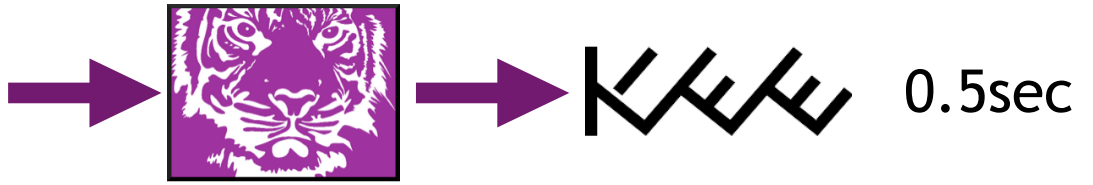
Banescu, et al., A Framework for Measuring Soft. Obf. Res. Against Automated Attacks, SPRO 2015

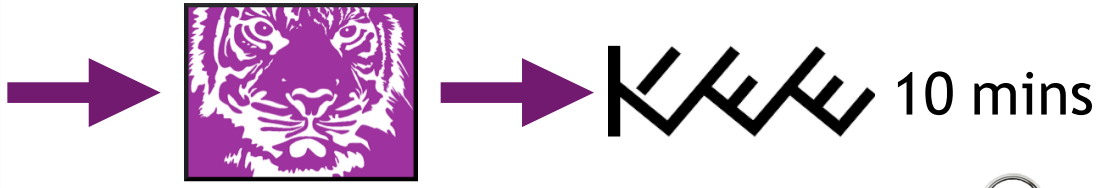
Metric 2: Resilience to Analysis Tools

Ca' Foscari University of Venice

- Obfuscation Resilience can be measured as the percentage of the program that has not been removed by an automated de-obfuscation treatment (using static or dynamic analysis tools)
- Failure due to bugs, lack of performance tuning, or code transformation is good

```
int main(int argc,
        char* argv[]) {
  if (argv[1][0] == 97 &&
      argv[1][1] == 98 &&
      argv[1][2] == 99 &&
      argv[1][3] == 100 &&
      argv[1][4] == 101) {
    printf("win\n");
  } else {
    printf("lose\n");
  }
}
```



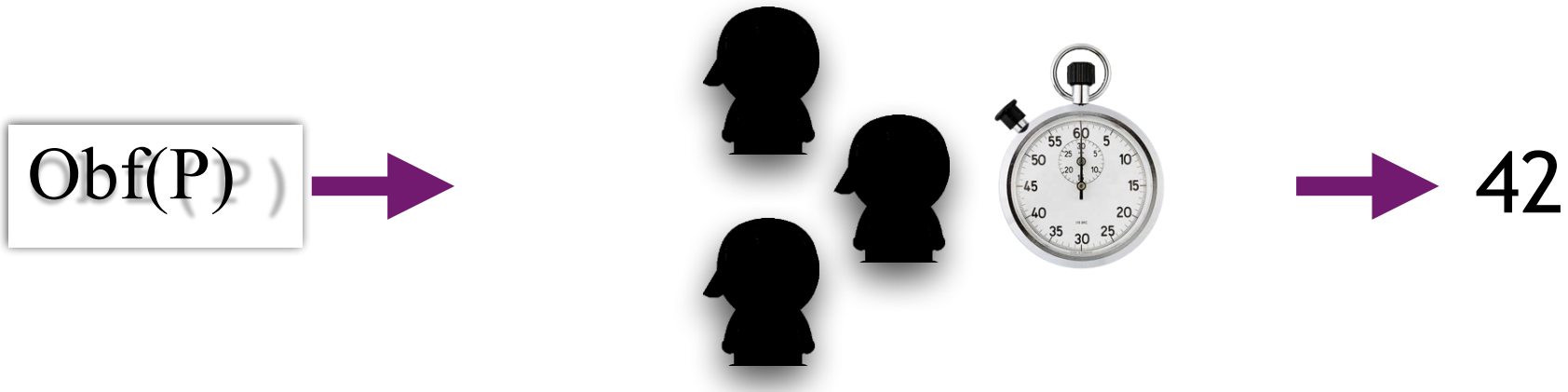


Virtualize +
Encode Program Array +
Make Input Dependent

Metric 3: Empirical Experiments

Ca' Foscari University of Venice

- Measure the time it takes for students to solve a task on the obfuscated code
- **Issues**: Inexperience, doesn't scale, students get better over time

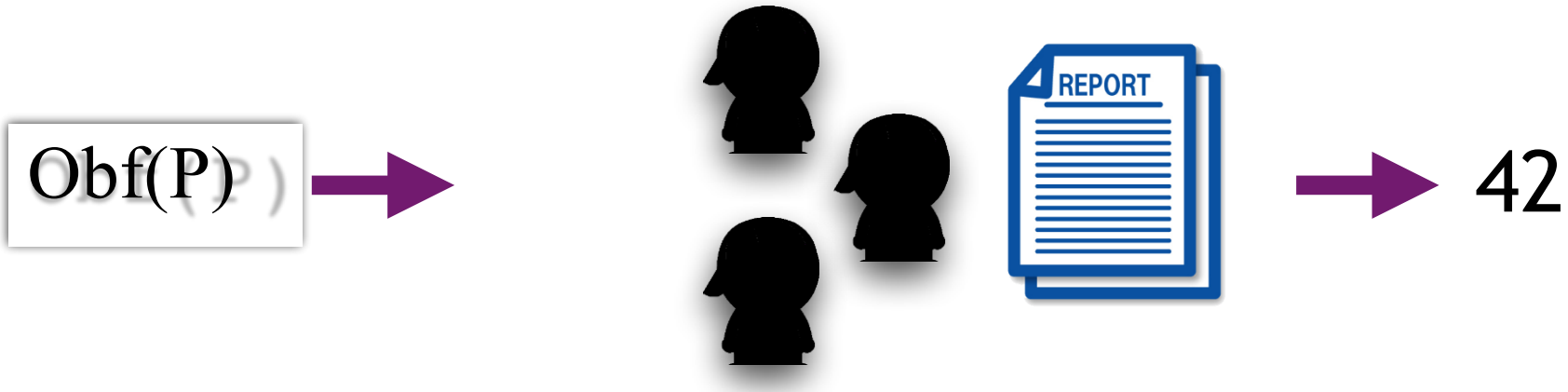


Ceccato et al., The effectiveness of source code obfuscation: …, ICPC'09

Metric 3: Empirical Experiments

Ca' Foscari University of Venice

- Analyse the reports from Tiger Teams try to break obfuscated code
- **Issues**: Hard to get collaboration, no control on time, only qualitative analysis possible



25 Years of Software Obfuscation – Can It Keep Pace with Progress in Code Analysis?

Ca' Foscari University of Venice

- (Schrittwieser et al, 2013)

Code analysis categories	Example
Pattern matching	Malware signatures
Automated static analysis	Heuristic malware detection
Automated dynamic analysis	Malware analysis in the labs of anti-virus vendors
Human-assisted analysis	Reverse engineering

Attacker's aims	Example
Finding the location of data (LD)	Extraction of licensing keys from binary
Finding the location of program functionality in the code (LC)	Finding the location of a copy protection mechanism
Extraction of code fragments (EC)	Extraction of code fragments for rebuilding verification routines for licensing keys
Understanding the program (UC)	Understand a proprietary cipher in order to start cryptanalysis attempts

Analysis of the Strength of Code Obfuscation

Ca' Foscari University of Venice

	PM		Autom. Static				Autom. Dynamic				Human Assisted			
Name	**LD**	**LC**	**LD**	**LC**	**EC**	**UC**	**LD**	**LC**	**EC**	**UC**	**LD**	**LC**	**EC**	**UC**
Data obfuscation														
Reordering data			✓				✓							
Changing encodings	✓						✓							
Converting static data to procedures	✓						✓				✓			
Static code rewriting														
Replacing instructions		✓		✓										
Opaque predicates				✓										
Inserting dead code						✓			✓	✓				
Inserting irrelevant code		✓												
Reordering														
Loop transformations														
Function splitting/recombination						✓								
Aliasing						✓			✓					
Control flow obfuscation	✓								✓	✓		✓		✓
Parallelized code														
Name scrambling						✓								
Removing standard library calls														✓
Breaking relations														
Dynamic code rewriting														
Packing/Encryption		✓		✓				✓	✓			✓		✓
Dynamic code modifications														
Environmental requirements														
Hardware-assisted code obfuscation											✓			
Virtualization			✓	✓				✓				✓		✓
Anti-debugging techniques						✓		✓	✓					✓

Analysis of the Strength of Code Obfuscation

Ca' Foscari University of Venice

Dynamic code rewriting													
Packing/Encryption		✓		✓			✓	✓			✓		✓
Dynamic code modifications													
Environmental requirements													
Hardware-assisted code obfuscation										✓			
Virtualization			✓	✓			✓				✓		✓
Anti-debugging techniques					✓		✓	✓					✓

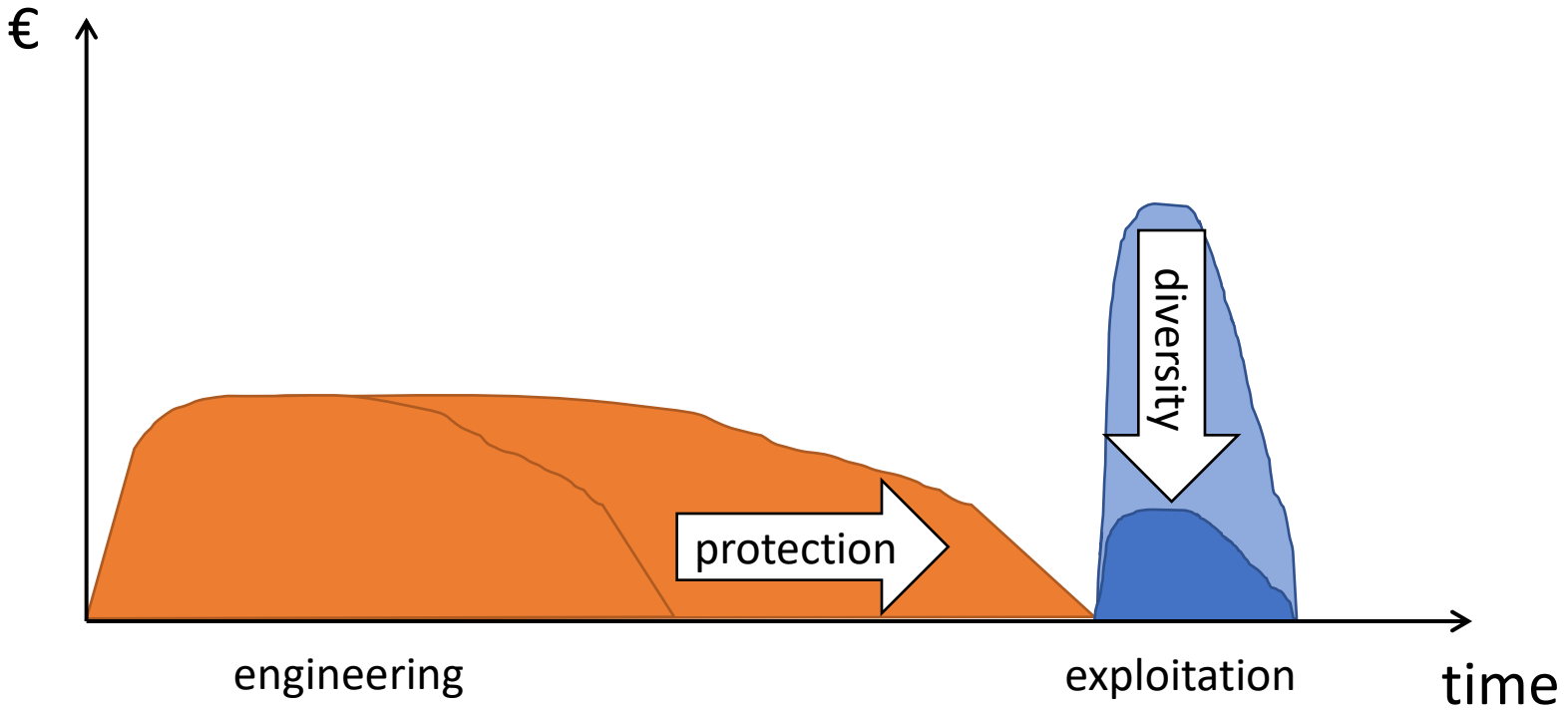
Legend		
		obfuscation breaks analysis fundamentally
		obfuscation is not unbreakable, but makes analysis more expensive
		obfuscation only results in minor increases of costs for analysis
	✓	A checkmark indicates that the rating is supported by results in the literature
		Scenarios without a checkmark were classified based on theoretical evaluation

Complexity Metrics

Economics of MATE attacks

Ca' Foscari University of Venice

1. What is the most appropriate protection?
2. What does a protection really buy us?

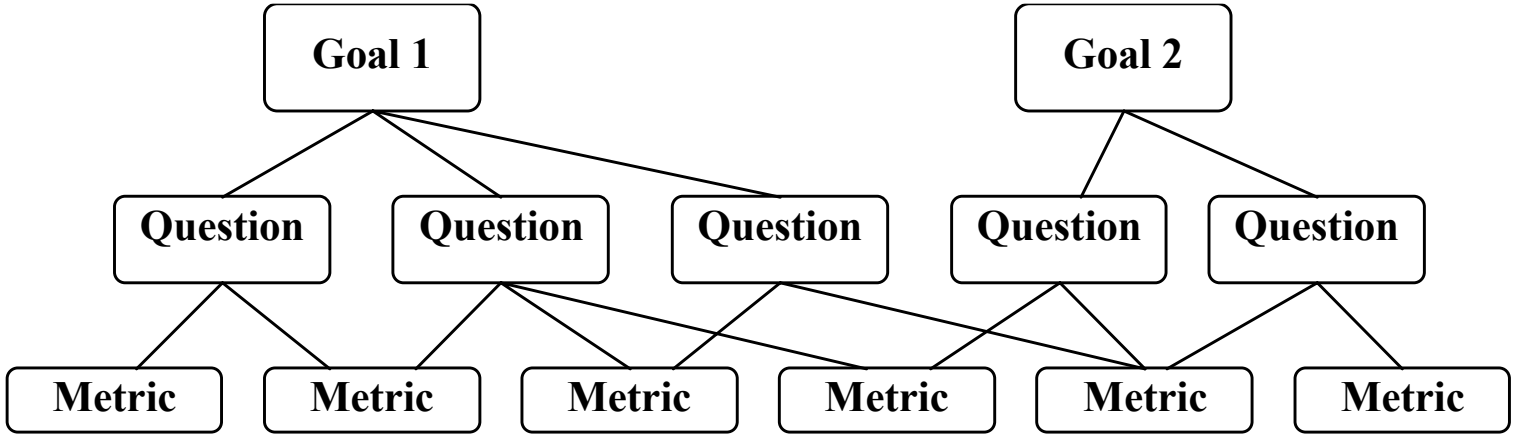


15

The Goal – Questions – Metrics Approach (Basili et al)

Ca' Foscari University of Venice

- Goal: What am I trying to achieve?
- Questions: What matters for achieving that?
- Metrics: How do we evaluate that?



The Goal – Questions – Metrics Approach

Ca' Foscari University of Venice

- ASPIRE project level

- Goal:
  - Optimize protection process
- Questions:
  - Which assets, attack steps, tools, protections, ...
  - What is their potency, resilience, cost, value, ...
- Metrics:
  - Measurable features of attacks, of protections and of (un)protected software

The Goal – Questions – Metrics Approach

Ca' Foscari University of Venice

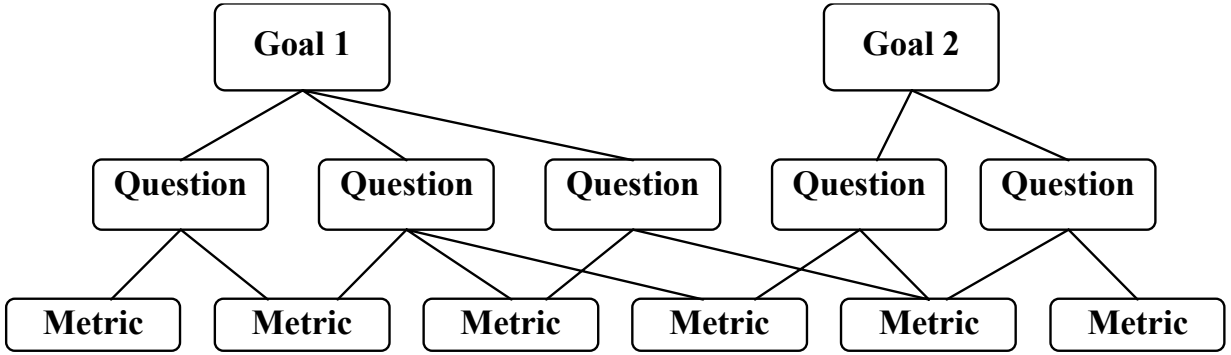
- Your Individual Protection

- Goal:
  - Protect specific software assets against specific attack(s)
- Questions:
  - What determines effort, what is delta in effort?
- Metrics:
  - Measurable features of attack steps and of (un)protected software

The Goal – Questions – Metrics Approach

Ca' Foscari University of Venice

- Advance warning



- established metrics were designed for other goals!
  - maintainability, testability, reliability, ...
- custom metrics are very specific
- specific vs generic goals?

Software Protection Strength?

Ca' Foscari University of Venice

Four criteria (Collberg et al)

- Potency: confusion, complexity, manual effort
- Resilience: resistance against (automated) tools
- Cost: performance, code size
- Stealth: identification of (components of) protections

Metrics on Obfuscation

Ca' Foscari University of Venice

- 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.

Cyclomatic number (McCabe, 1976)

- control flow complexity

  V(cfg) = #edges − #nodes + 2 * P

- P is the number of connected components in the graph
- single component:

  V(cfg) = #edges − #nodes + 2

- related to the number of linearly independent paths
- related to number of tests needed to invoke all paths

Cyclomatic number (McCabe, 1976)

- Quite some problems:
  - no recognition of familiar structures
  - what about obfuscated unstructured CFGs?
  - what to do when functions are not identified well?
  - no recognition of data dependencies
  - what about object-oriented code?
  - what about conditional statements?
  - combinatoric issues

Program size & derivatives (Halstead, 1977)

- Lines of code
- Derivatives

Measure	Symbol	Formula
Program length	N	$N = N_1 + N_2$
Program vocabulary	n	$n = n_1 + n_2$
Volume	V	$V = N*(\log_2 n)$
Estimated abstraction level	L	$L= (2\ n_2) / (n_1*N_2)$
Difficulty	D	$D= 1 / L$
Effort	E	$E = V * D$
Time	T	$T = E / 18$
Remaining bugs	B	$B = E^{2/3} / 3000$

n_1 = number of distinct operators,
n_2 = number of distinct operands,
N_1 = total number of operator occurrences, and
N_2 = total number of operand occurrences.

Combine all of them (Anckaert et al, 2007)

Ca' Foscari University of Venice

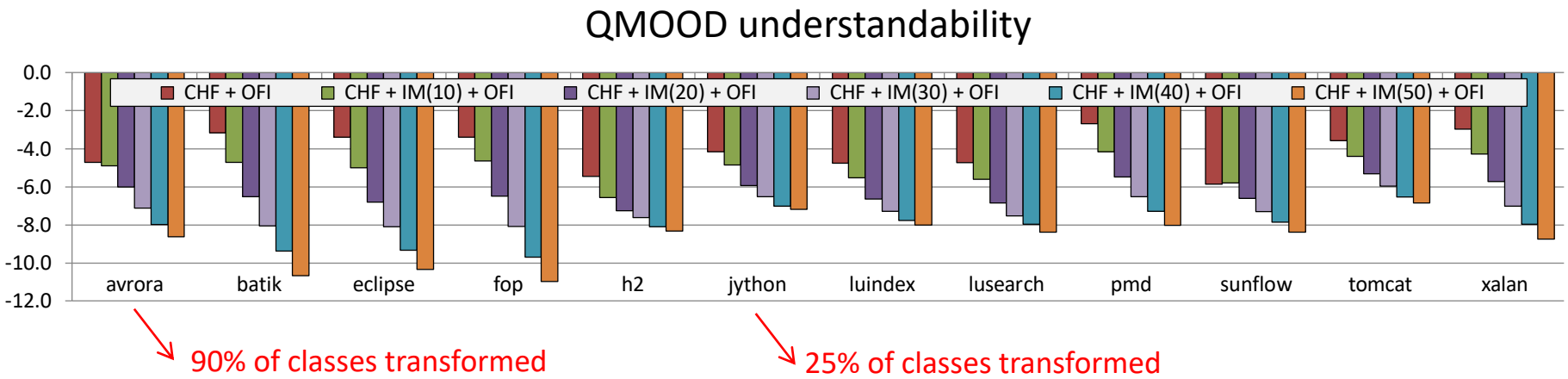
- code & code size
  - e.g., #instructions, weighted by "complexity"
- control flow complexity
- data flow complexity
  - sizes slices
  - sizes live sets, working sets
  - sizes points-to sets
  - fan-in, fan-out
  - data structure complexities (Munson and Khoshgoftaar, 1993)
- data
  - application-specific

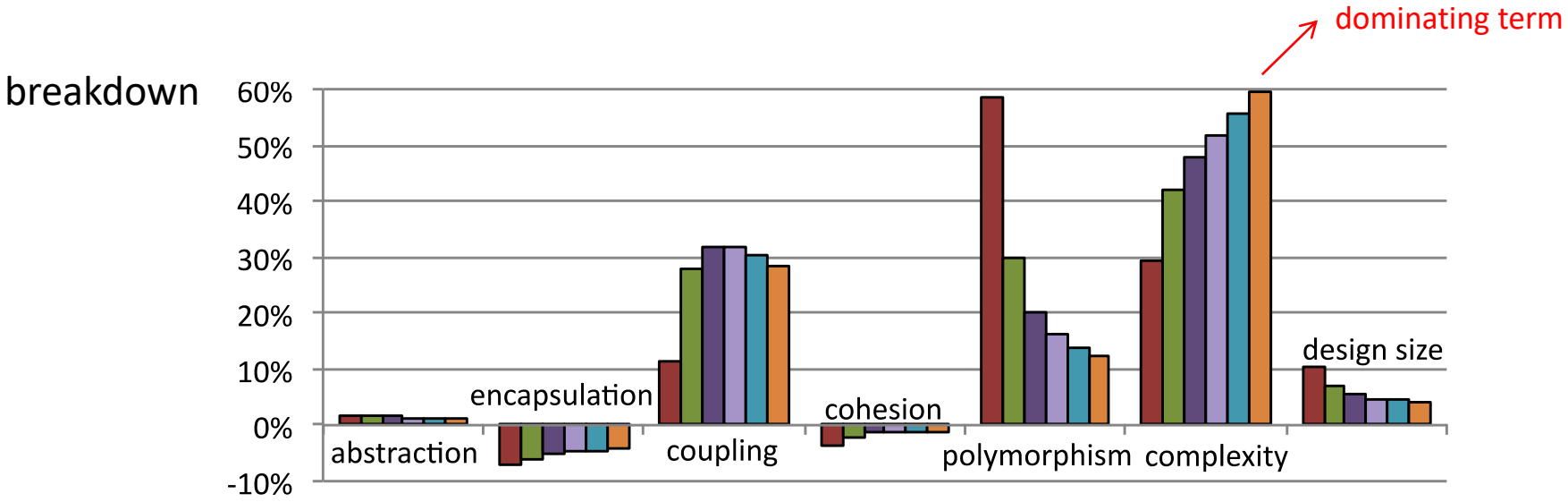
static -> graphs

dynamic -> traces

Object-Oriented Quality Metrics (Bansiya & Davis, 2002)

Ca' Foscari University of Venice





OO Quality Metrics for Java obfuscation

Ca' Foscari University of Venice

- Weighted Methods Per Class (WMC)
- Depth of Inheritance Tree (DIT)
- Number of Children (NOC)
- Coupling Between Object Classes (CBO)
- Response For a Class (RFC)
- Lack of Cohesion in Methods (LCOM)

- M Ceccato, A. Capiluppi, P Falcarin, C Boldyreff: A Large Study on the Effect of Code Obfuscation on the Quality of Java Code. In Empirical Software Engineering, Springer, 2015.

Exercise

Ca' Foscari University of Venice

- Try the Software Metrics in Tigress
- https://tigress.wtf/softwareMetrics.html
- Search for one or more C programs with a variable that can be considered an asset to protect
- Apply one transformation and measure the metrics before and after
- What is the potency (After/Before) of the change ?

Abstract Interpretation (Dalla Preda, Giacobazzi et al)

Ca' Foscari University of Venice

- Abstract domains model program properties
  Abstract interpretation computes properties
- Domains are partially ordered in terms of concreteness
- Obfuscating transformation is less potent if it preserves more concrete properties
- Automatic deobfuscation of opaque predicates
- Not clear how this scales ...

Tool-based metrics

Ca' Foscari University of Venice

- Use attacker's tools and heuristics
  - Model effort/time in terms of input size
  - Compute output size
  - Compute relevance of output
    - false positives/negatives
    - receiver operator curves (ROC)
    - recall and precision
    - pruning factors
- Major problems:
  - predicting tool output
  - generallity of the results

Disassembly Thwarting (Linn & Debray, 2003)

Ca' Foscari University of Venice

- Confusion factor

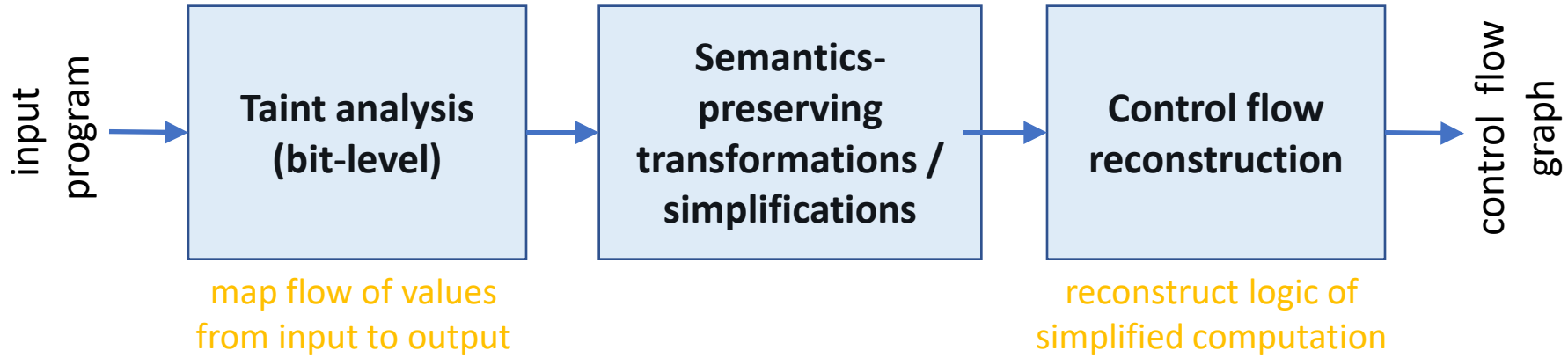
$$CF = |A - P| / |A|.$$

with A = ground truth set of instruction addresses
and P = set determined by static disassembly

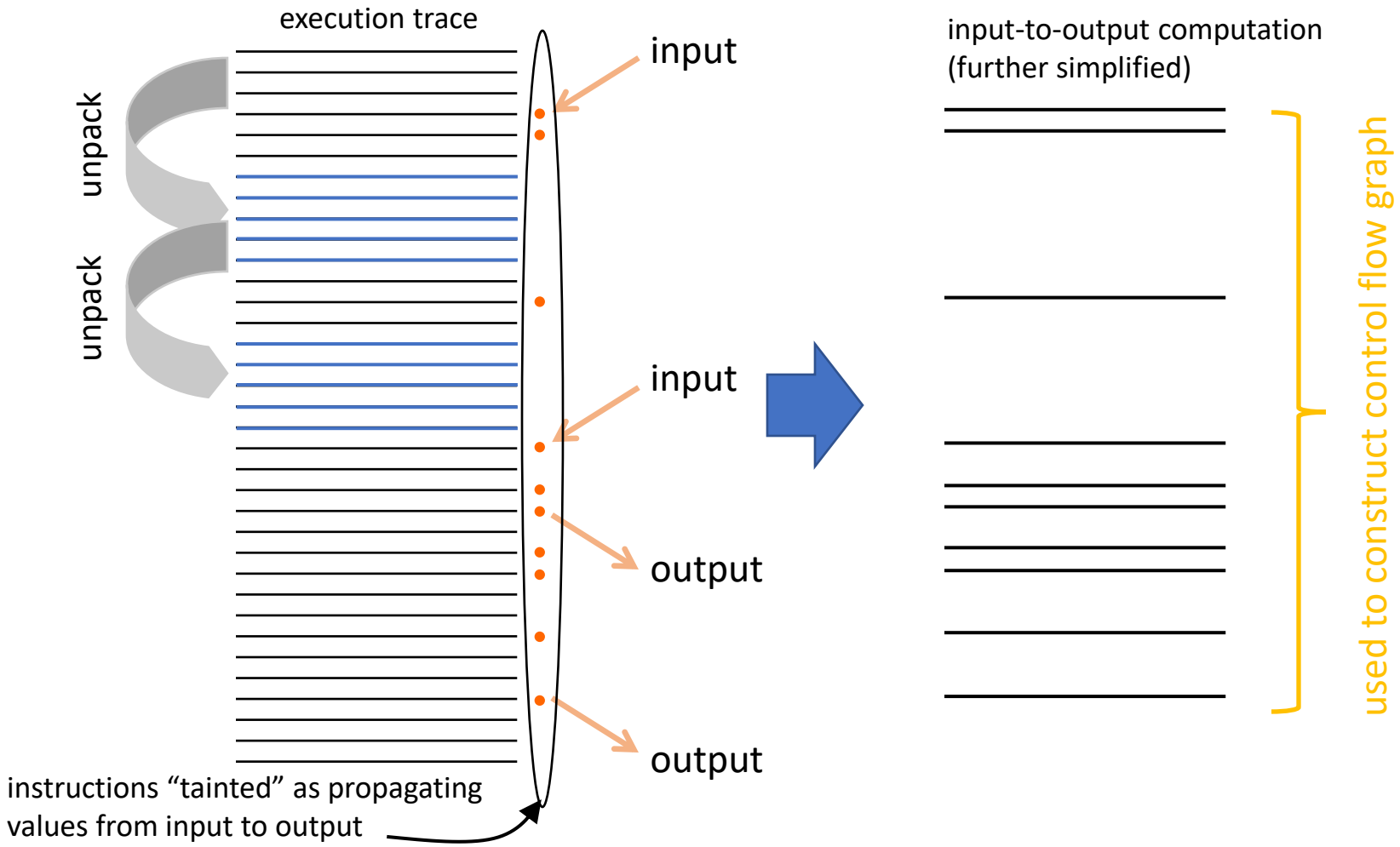
Program	Confusion factor (%)								
	Linear sweep (Objdump)			Recursive traversal			Commercial (IDA Pro)		
	Instructions	*Basic blocks*	*Functions*	*Instructions*	*Basic blocks*	*Functions*	*Instructions*	*Basic blocks*	*Functions*
compress95	43.93	63.68	100.00	30.04	40.42	75.98	75.81	91.53	87.37
gcc	34.46	53.34	99.53	17.82	26.73	72.80	54.91	68.78	82.87
go	33.92	51.73	99.76	21.88	30.98	60.56	56.99	70.94	75.12
ijpeg	39.18	60.83	99.75	25.77	38.04	69.99	68.54	85.77	83.94
li	43.35	63.69	99.88	27.22	38.23	76.77	70.93	87.88	84.91
m88ksim	41.58	62.87	99.73	24.34	35.72	77.16	70.44	87.16	87.16
perl	42.34	63.43	99.75	27.99	39.82	76.18	68.64	84.62	87.13
vortex	33.98	55.16	99.65	23.03	35.61	86.00	57.35	74.55	91.29
Geo. mean	39.09	59.34	99.75	24.76	35.69	74.43	65.45	81.40	84.97

Reverse-engineering obfuscated programs (Debray et al, 2014)

- no obfuscation-specific assumptions
  - treat programs as input-to-output transformations
  - use semantics-preserving transformations to simplify execution traces
- dynamic analysis to handle runtime unpacking



Trace simplification



"Semantic-preserving" simplification

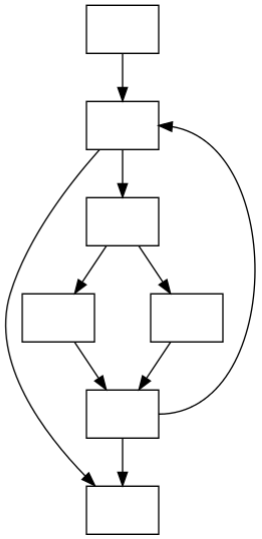
Ca' Foscari University of Venice

- Quasi-invariant locations: locations that have the same value at each use.
- Their transformations (currently):
  - Arithmetic simplification
    - adaptation of constant folding to execution traces
    - consider quasi-invariant locations as constants
    - controlled to avoid over-simplification
  - Control simplification
    - E.g., convert indirect jump through a quasi-invariant location into a direct jump
  - Data movement simplification
    - use pattern-driven rules to identify and simplify data movement.
  - Dead code elimination
    - need to consider implicit destinations, e.g., condition code flags.

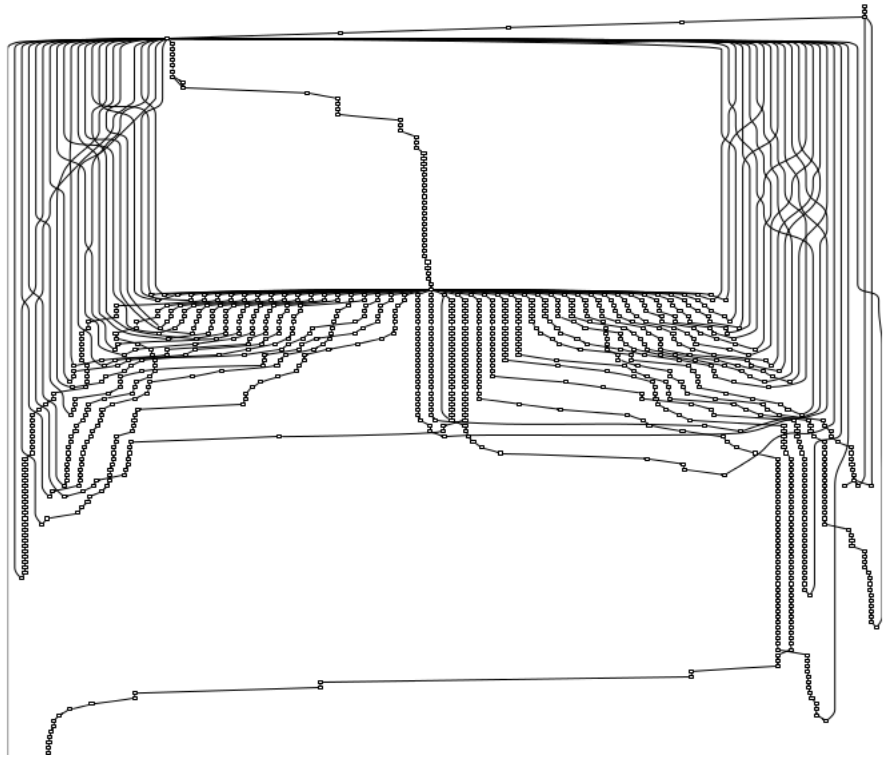
Example: Themida Emulation Obfuscation

Ca' Foscari University of Venice

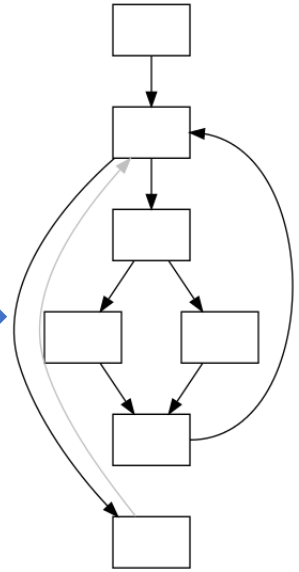
original



obfuscated (cropped)



deobfuscated



Empirical Experiments

Quantitative Analysis

Experiments with Human Subjects

- What is the real protection provided?
  - For identification/engineering
  - For exploitation
- Which protection is better?
- Against which type of attacker?
- How fast do subjects learn to attack protections?
- Which attack methods are more likely to be used?
- Which attack methods are more likely to succeed?

Experiments with Human Subjects

Ca' Foscari University of Venice

- Very hard to set up and get right
  - with students: cheap but representative?
  - with experts: expensive, but controlled?
  - what to test? (Dunsmore & Roper, 2000)
    - maintenance
    - recall
    - subjective rating
    - fill in the blank
    - mental simulation
  - How to extrapolate

Obfuscation

Ca' Foscari University of Venice

- 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");
```

Experiments with Human Subjects

- What is the real protection provided?
  - For identification/engineering
  - For exploitation
- Which protection is better?
- Against which type of attacker?
- How fast do subjects learn to attack protections?
- Which attack methods are more likely to be used?
- Which attack methods are more likely to succeed?

Research Questions

Ca' Foscari University of Venice

- 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

Ca' Foscari University of Venice

- Goal
  - To analyse 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

Ca' Foscari University of Venice

- Treatments
  - Decompiled obfuscated code
  - Decompiled clear code

- Dependent variables
  - Ability to perform comprehension tasks
  - Time required for comprehension
  - Ability to correctly perform an attack
  - Time required to perform an attack

Null hypotheses

Ca' Foscari University of Venice

- 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

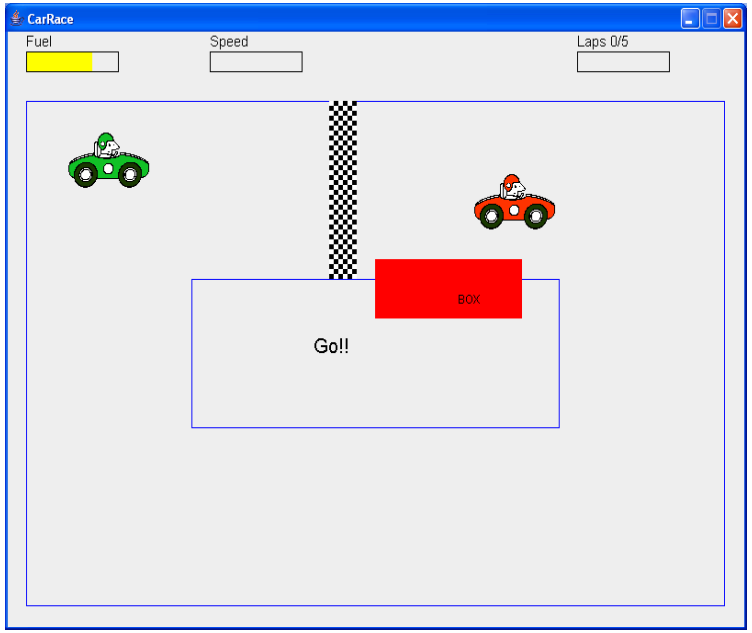
Ca' Foscari University of Venice

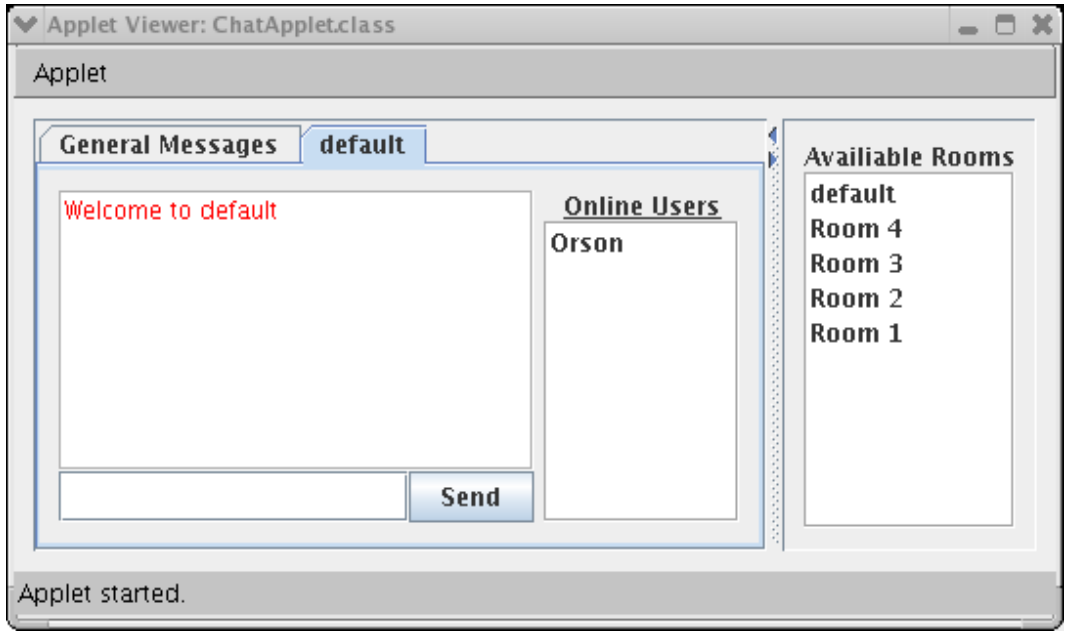
- More than a hundred of students from five universities
- Good knowledge of Java programming
- Knowledge of software engineering topics
  - Design
  - Testing
  - Code analysis

Objects of the Experiment

Ca' Foscari University of Venice

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





Experiment Balanced design

Ca' Foscari University of Venice

- 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**

Treatments

- Identifier Renaming obfuscation vs Opaque Predicates
- 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");
```

Clear code fragment chat program

Ca' Foscari University of Venice

```
public void addUserToList(String strRoomName, String strUser)
{
    RoomTabItem tab = getRoom(strRoomName);
    if(tab != null)
        tab.addUserToList(strUser);
}

public void removeUserFromList(String strRoomName, String strUser)
{
    RoomTabItem tab = getRoom(strRoomName);
    if(tab != null)
        tab.removeUserFromList(strUser);
}
```

Fragment with renamed identifiers

Ca' Foscari University of Venice

```
public void k(String s, String s1)
{
    h h1 = h(s);
    if(h1 != null)
        h1.k(s1);
}

public void l(String s, String s1)
{
    h h1 = h(s);
    if(h1 != null)
        h1.l(s1);
}
```

Fragment with opaque predicates

Ca' Foscari University of Venice

```
public void removeUserFromList(String strRoomName, String strUser) {
   RoomTabItem tab = null;
   if (Node.getI() != Node.getH()) {
      Node.getI().getLeft().swap(Node.getI().getRight());
      tab.transferFocusUpCycle();
   } else {
      Node.getF().swap(Node.getI());
      tab = getRoom(strRoomName);
   }
   if (Node.getI() != Node.getH()) {
      receiver.getClass().getAnnotations();
      Node.getH().getRight().swap(Node.getG().getLeft());
   } else {
      if (tab != null)
         if (Node.getI() != Node.getH()) {
            Node.getF().setLeft(Node.getG().getRight());
            roomList.clearSelection();
         } else {
            Node.getI().swap(Node.getH());
            tab.removeUserFromList(strUser);
         }
      Node.getI().getLeft().swap(Node.getF().getRight());
   }
}
```

Preliminary lecture

Ca' Foscari University of Venice

- 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.

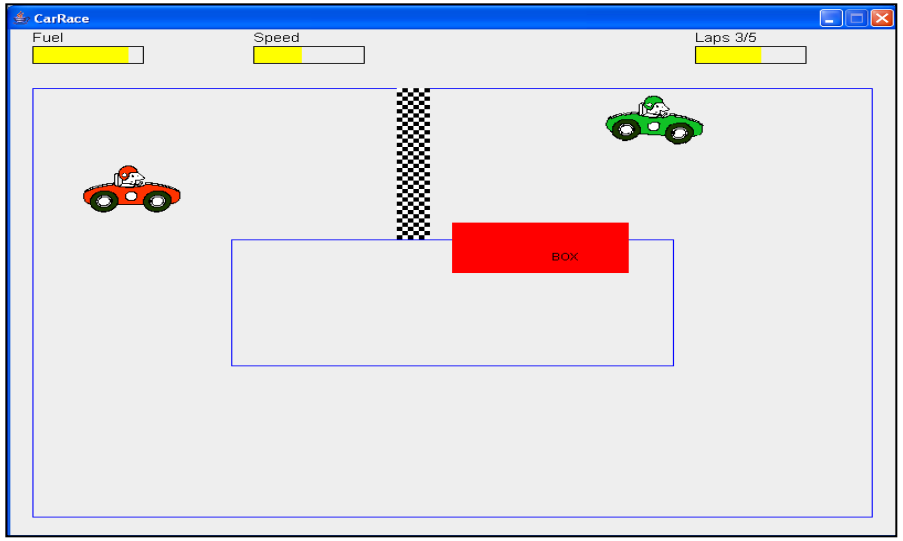
Experimental sessions

Ca' Foscari University of Venice

- 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

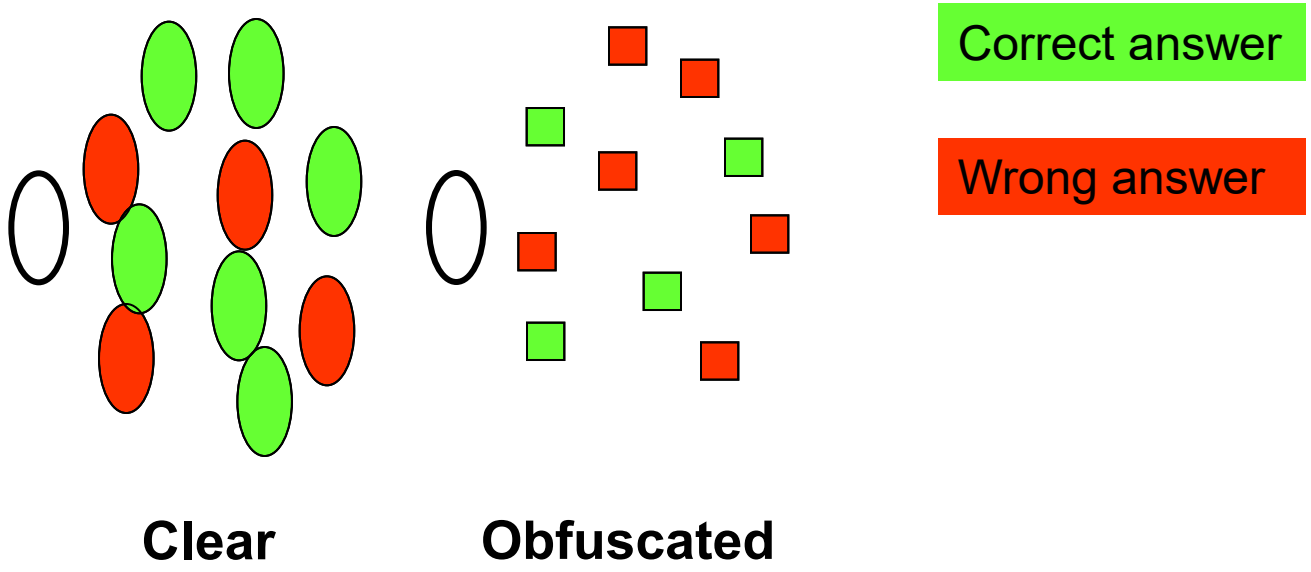


Survey questionnaire

Ca' Foscari University of Venice

- 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

Ca' Foscari University of Venice

	Comprehension		Attack		Overall	
Treatment	**Wrong**	**Correct**	**Wrong**	**Correct**	**Wrong**	**Correct**
Clear	7					
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

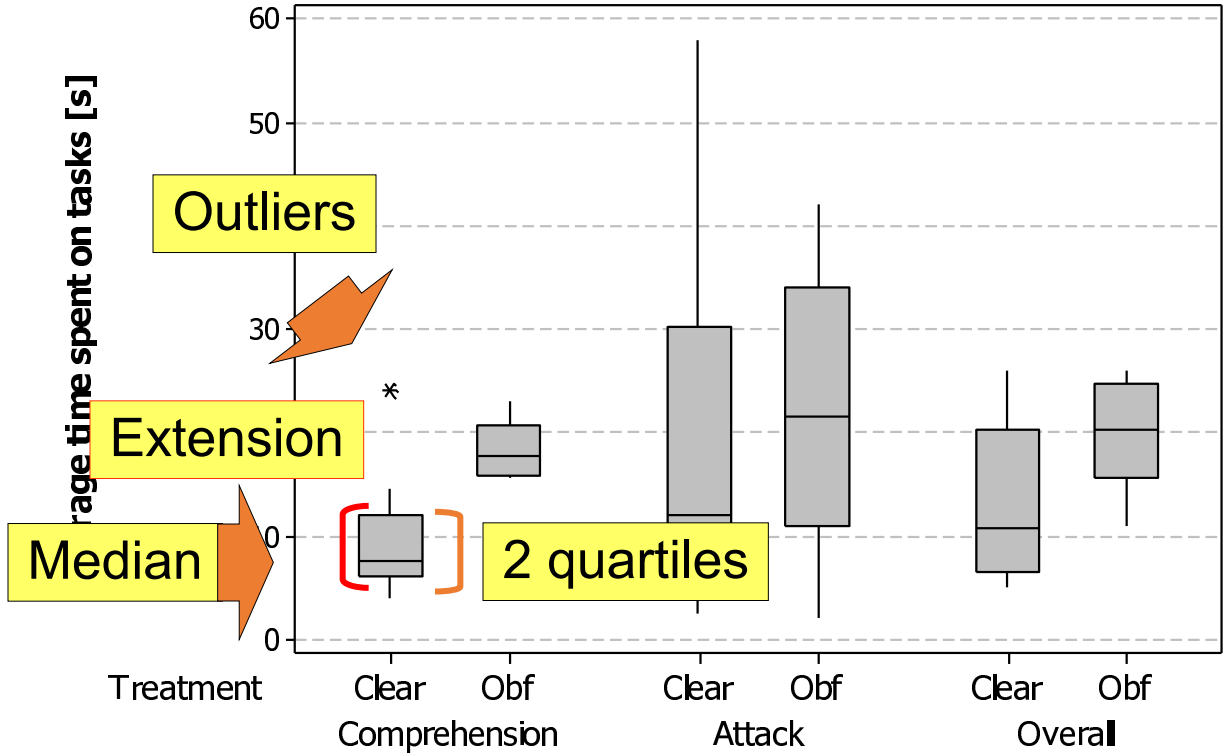
$$OR = \frac{\frac{p}{1-p}}{\frac{q}{1-q}}$$

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



	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

Ca' Foscari University of Venice

- **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.

Null hypotheses

Ca' Foscari University of Venice

- **H01** The obfuscation does not significantly reduce source code comprehensibility.
- ~~**H02** The~~ ded to perform
- ~~**H03** The~~ of subjects
- **H04** The ded to perform an attack.

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

Threat to validity

Ca' Foscari University of Venice

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

Ca' Foscari University of Venice

- Obfuscation (Id-Renaming) thwarts reverse engineering by reducing success factor of attacks
- Obfuscation slightly delays Comprehension
  - Delay can be used to estimate the renewability time in dynamic security

- Publications
  - M Ceccato, M Di Penta, P Falcarin, F Ricca, M Torchiano, P Tonella: A Family of Experiments to Assess the Effectiveness and Efficiency of Source Code Obfuscation Techniques. In Empirical Software Engineering, Springer, 19 (4), 2014.
  - M. Ceccato, M. DiPenta, J. Nagra, P. Falcarin, F. Ricca, M. Torchiano, and P. Tonella, "The effectiveness of source code obfuscation: an experimental assessment," in *IEEE International Conference on Program Comprehension (ICPC 2009),*
  - 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, 2008*

Empirical Experiments

Qualitative Analysis

Participants

Ca' Foscari University of Venice

- Professional penetration testers working for security companies
- Routinely involved in security assessment of company's products
- Profiles:
  - Hackers with substantial experience in the field
  - Fluent with state of the art tools (reverse engineering, static analysis, debugging, profiling, tracing, …)
  - Able to customize existing tools, to develop plug-ins for them, and to develop their own custom tools
- Minimal intrusion (hacker activities cannot be traced)

Experimental procedure

Ca' Foscari University of Venice

- Attack task definition
  - Description of the program to attack, attack scope, attack goal(s) and report structure
- Monitoring (long running experiment: 30 days)
  - Minimal intrusion into the daily activities
    - Could not be traced automatically or through questionnaires
  - Weekly conf-call to monitor the progress and provide support for clarifying goals and tasks
- Attack reports
  - Final (narrative) report of the attack activities and results
  - Qualitative analysis

Objects	C	H	Java	C++	Total
DemoPlayer	2,595	644	1,859	1,389	6,487
LicenseManager	53,065	6,748	819	-	58,283
OTP	284,319	44,152	7,892	2,694	338,103

Experiment in Company A

Ca' Foscari University of Venice

- 2 phases for 2 protection configurations:
  - All protections except Anti-debugging & Remote Attestation
    - Allows to collect execution traces and perform dynamic analysis
  - All protections
    - To face the final protection configuration
- Task: extract the crypto key from binary
- Hackers in the forensic lab
  - 10 years of experience in forensics such as analysis of pirate set-top boxes.
  - Security evaluation of their products before release
  - Security evaluation for third parties (e.g. police forces)

Experiment in Company B

Ca' Foscari University of Venice

- Protection configuration:
  - Remote attestation
  - White box cryptography
  - Data obfuscation
  - Binary code obfuscation
  - Call back checks
  - SoftVm (client side splitting)
  - Code guards
  - Client/server Splitting

- 2 phases for 2 hacker groups with complementary background:

  Expert reverse engineer from industry
  - 10 year of experience in obfuscation, de-obfuscation, reversing and all the relevant tools

  External hacker team
  - Experts in reverse engineering (Windows and Linux) in kernel and user mode systems, code obfuscation.

- Task: violate the license

Experiment in Company C

Ca' Foscari University of Venice

- Phase 1:
  - External independent testing team (needed for commercial exploitation)
    - Security analysts with 15 years of experience in penetration testing, code review and reverse engineering
    - Very qualified in vulnerability research, exploitation techniques and attacking crypto systems
  - Protection: only Diversified Crypto Library
- Phase 2:
  - Internal testing team
    - 10+ years in security (mobile security in particular, e.g. malware detection)
    - Comfortable with binary code and tools to used by hackers
  - Protections: phase 1 (DCL) +
    - Binary code obfuscation
    - Code guards
    - Call stack check
- Phase 3:
  - Internal testing team
  - Same configuration of phase 2 + Anti-debugging

Data collection

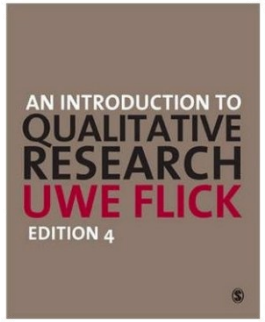
Ca' Foscari University of Venice

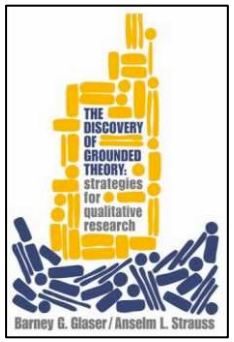
- Report in free format
- Professional hackers were asked to cover these topics:
  - type of activities carried out during the attack;
  - level of expertise required for each activity;
  - encountered obstacles;
  - decision made, assumptions, and attack strategies;
  - exploitation on a large scale in the real world
  - return / remuneration of the attack effort

Data analysis

Ca' Foscari University of Venice

- Qualitative data analysis method from Grounded Theory
  - Data collection
  - Open coding
  - Conceptualization
  - Model analysis





Open coding method (Grounded Theory)

Ca' Foscari University of Venice

- Performed by 7 coders from 4 academic project partners
  - Autonomously & independently
  - High level instructions
    - Maximum freedom to coders, to minimize bias
- Annotated reports have been merged
- No unification of annotations, to preserve viewpoint diversity

	Annotator							
Case study	A	B	C	D	E	F	G	Total
VideoPlayer	52	34	48	53	43	49	-	279
License	20	10	6	12	7	18	9	82
OTP	12	22	-	29	24	11	-	98
Total	**84**	**66**	**54**	**94**	**74**	**78**	**9**	**459**

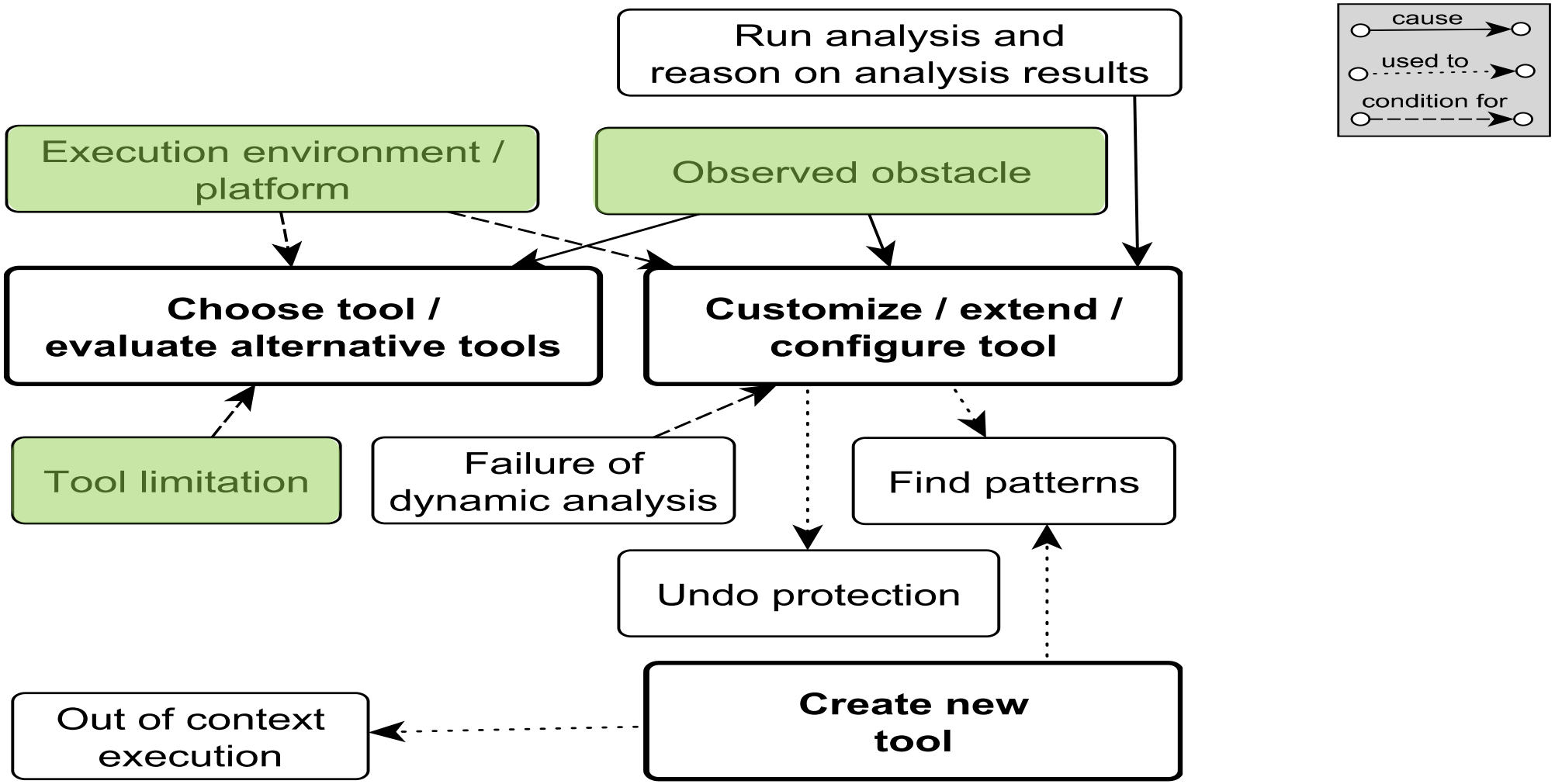
Conceptualization

Ca' Foscari University of Venice

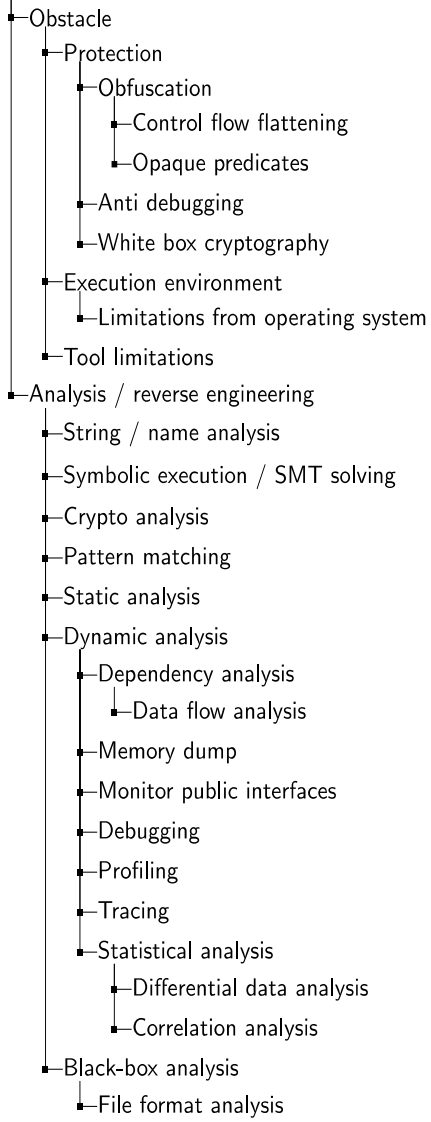
- Concept identification
  - Identify key concepts used by coders
  - Organize key concepts into a common hierarchy
- Model inference
  - Temporal relations (e.g., before)
  - Causal relations (e.g., cause)
  - Conditional relations (e.g., condition for)
  - Instrumental relation (e.g., used to)
- 2 joint meetings:
  - Merge codes (sentence by sentence, annotation by annotation)
  - Abstractions have been discussed, until consensus was reached
- Subjectivity reduction:
  - Consensus among multiple coders
  - Traceability links between abstractions and annotations to help decision revision

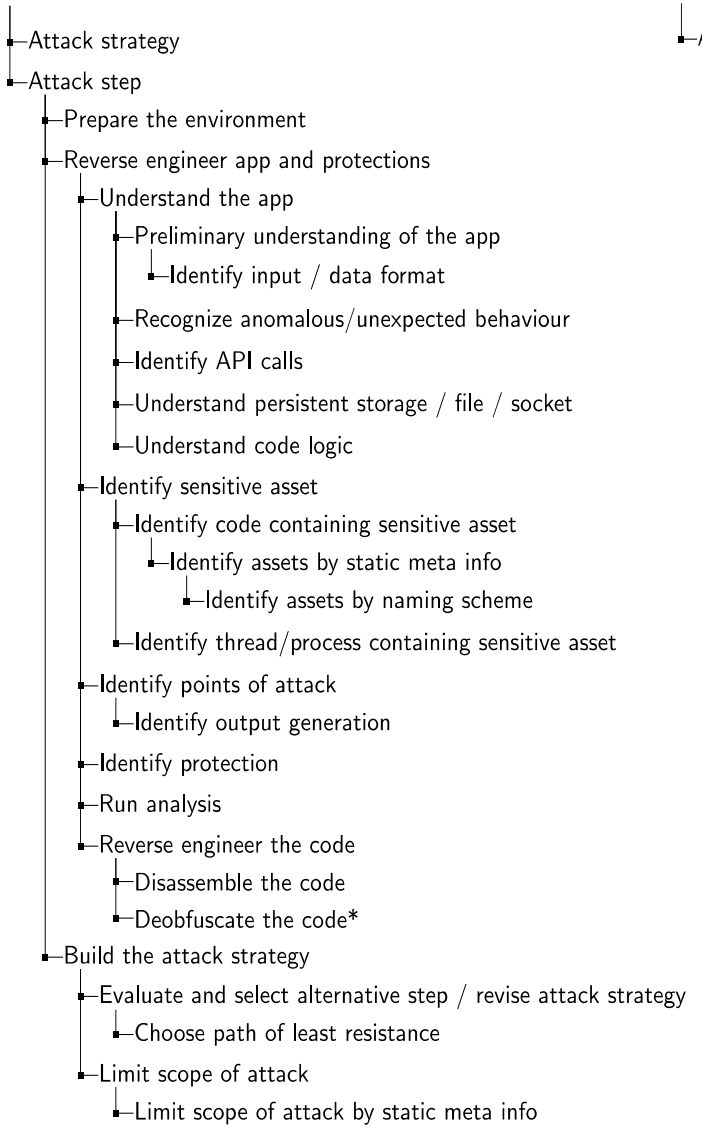
How attackers chose & customize tools

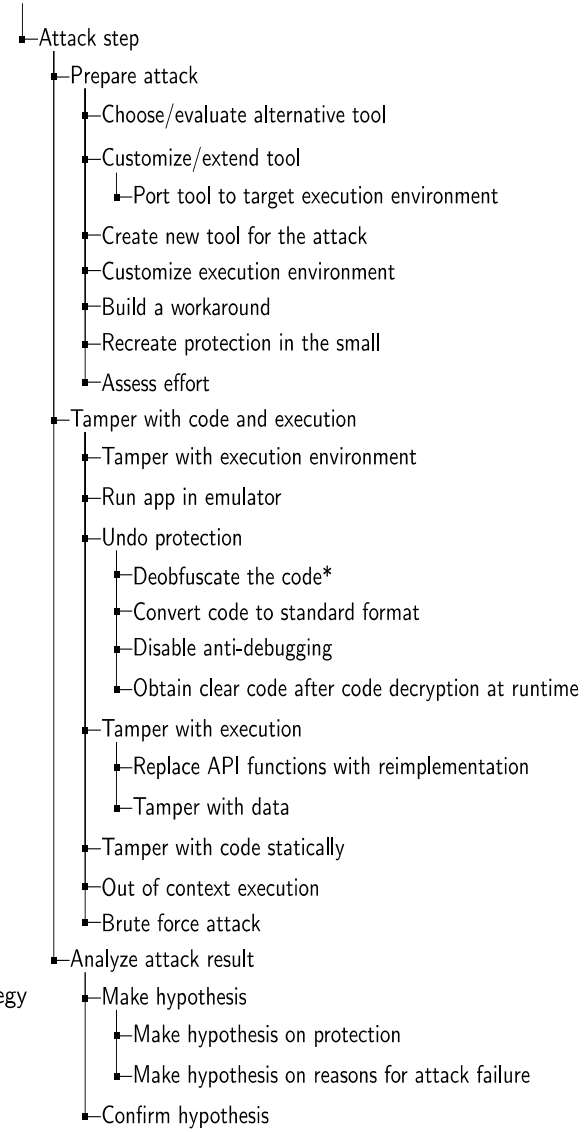
Ca' Foscari University of Venice

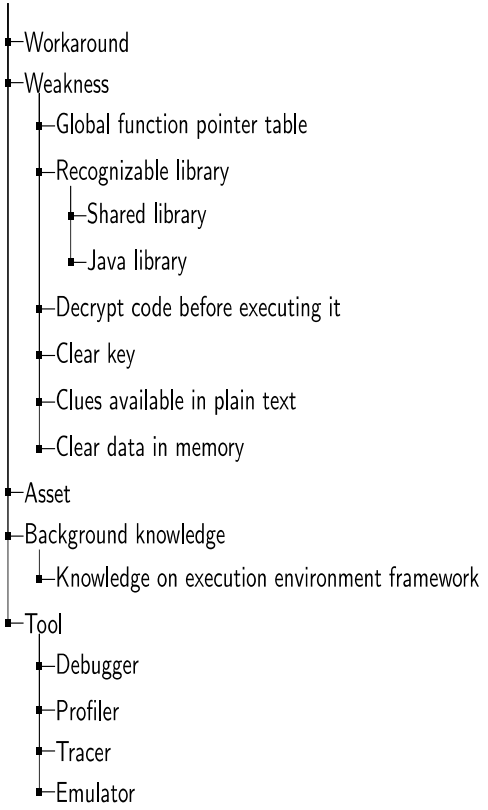


Results: taxonomy of concepts

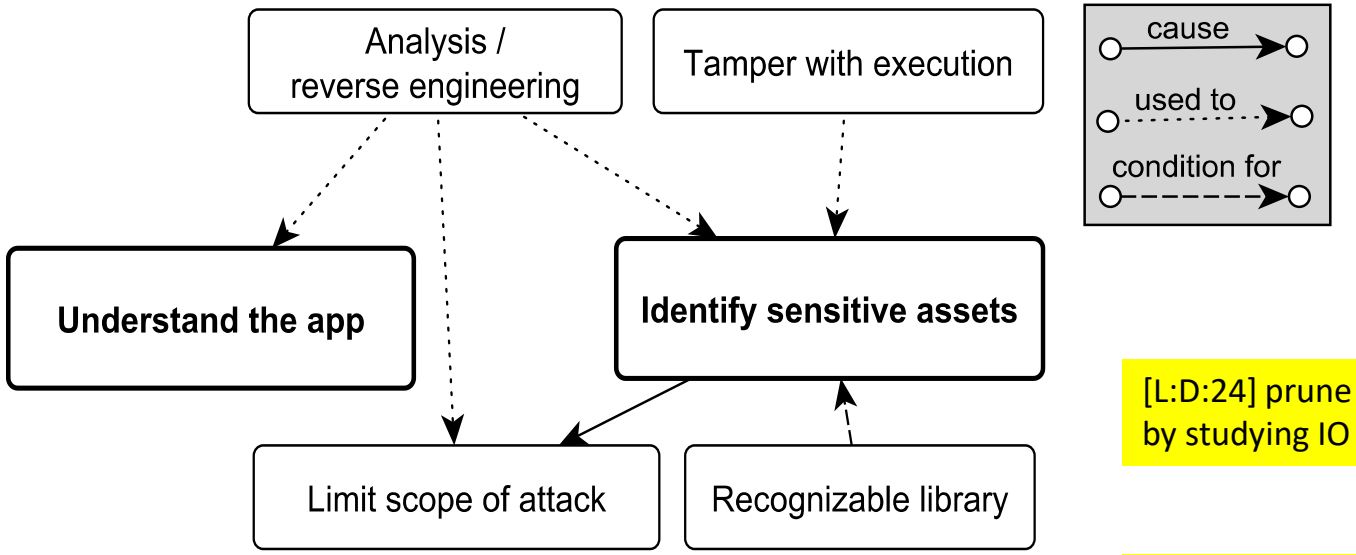








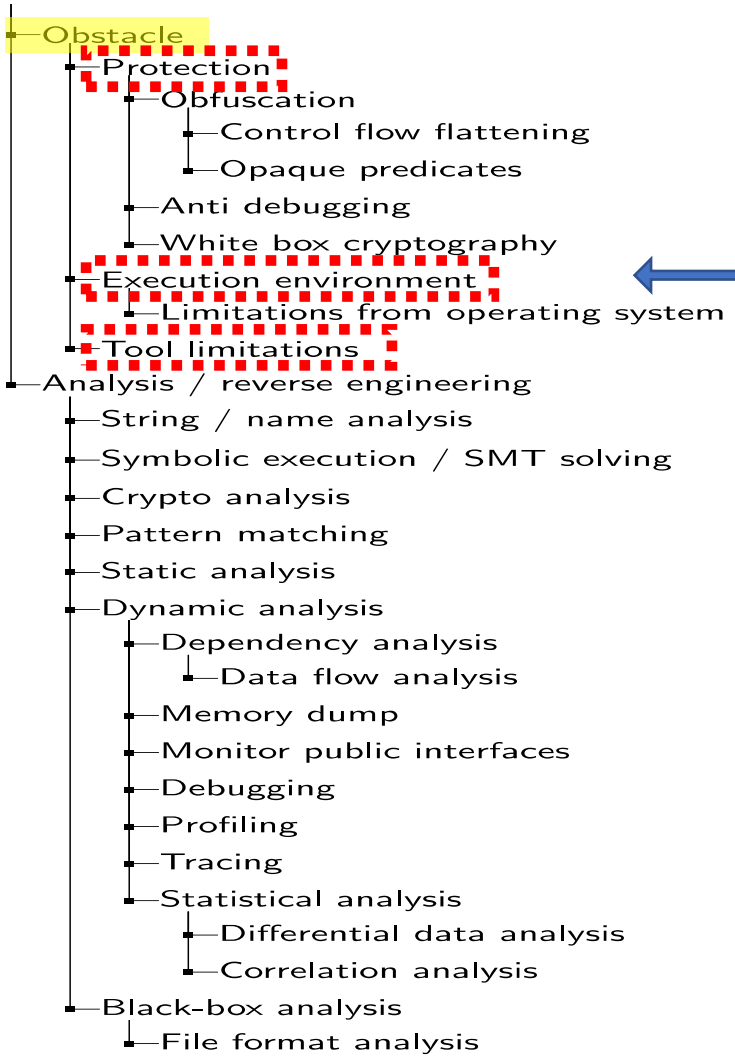
How hackers understand protected software



[L:D:24] prune search space for interesting code by studying IO behavior, in this case system calls

[L:D:26] prune search space for interesting code by studying static symbolic data, in this case string references in the code

Taxonomy

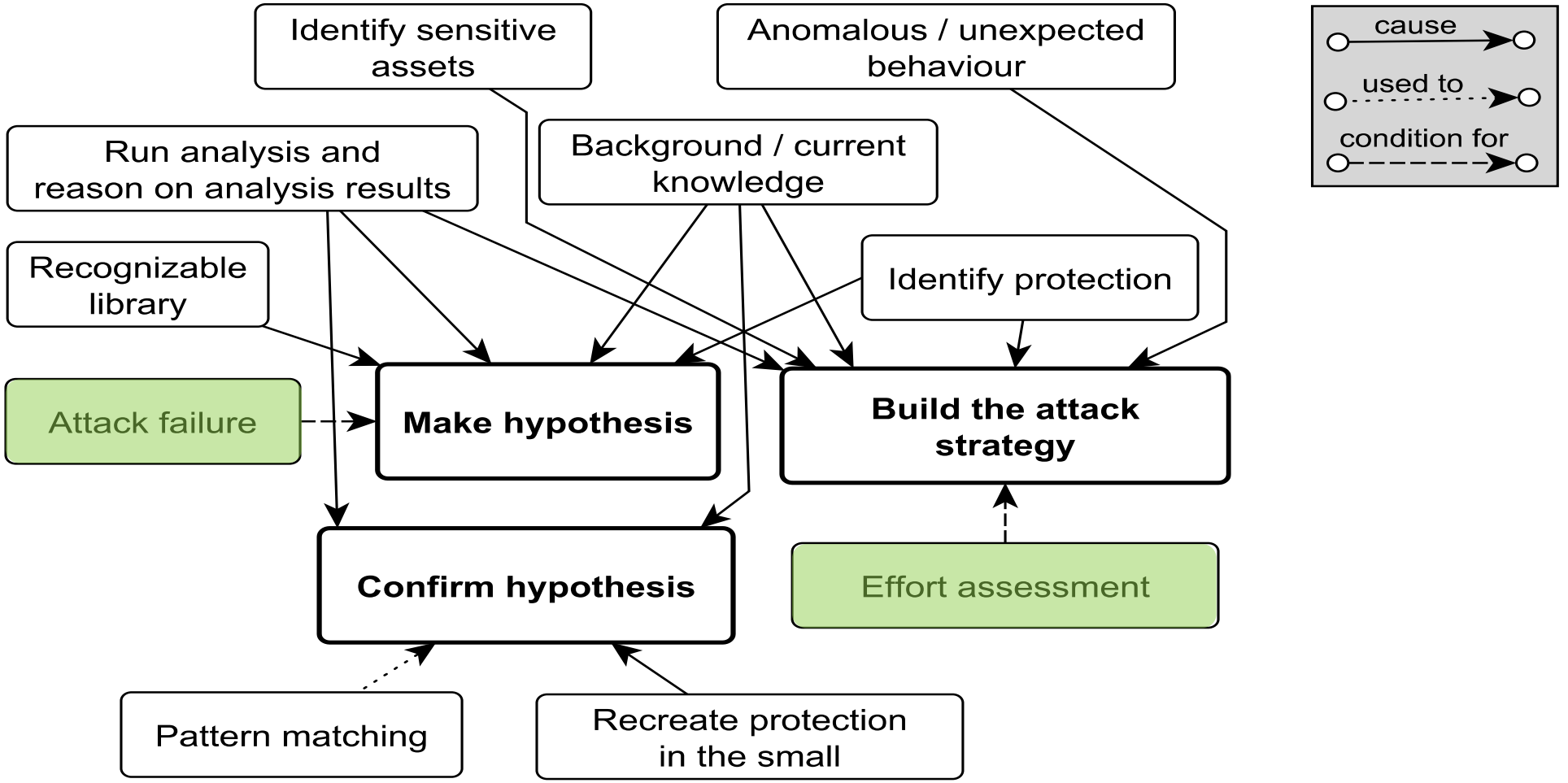


[P:F:7] General obstacle to understanding [by dynamic analysis]: execution environment (Android: limitations on network access and maximum file size)

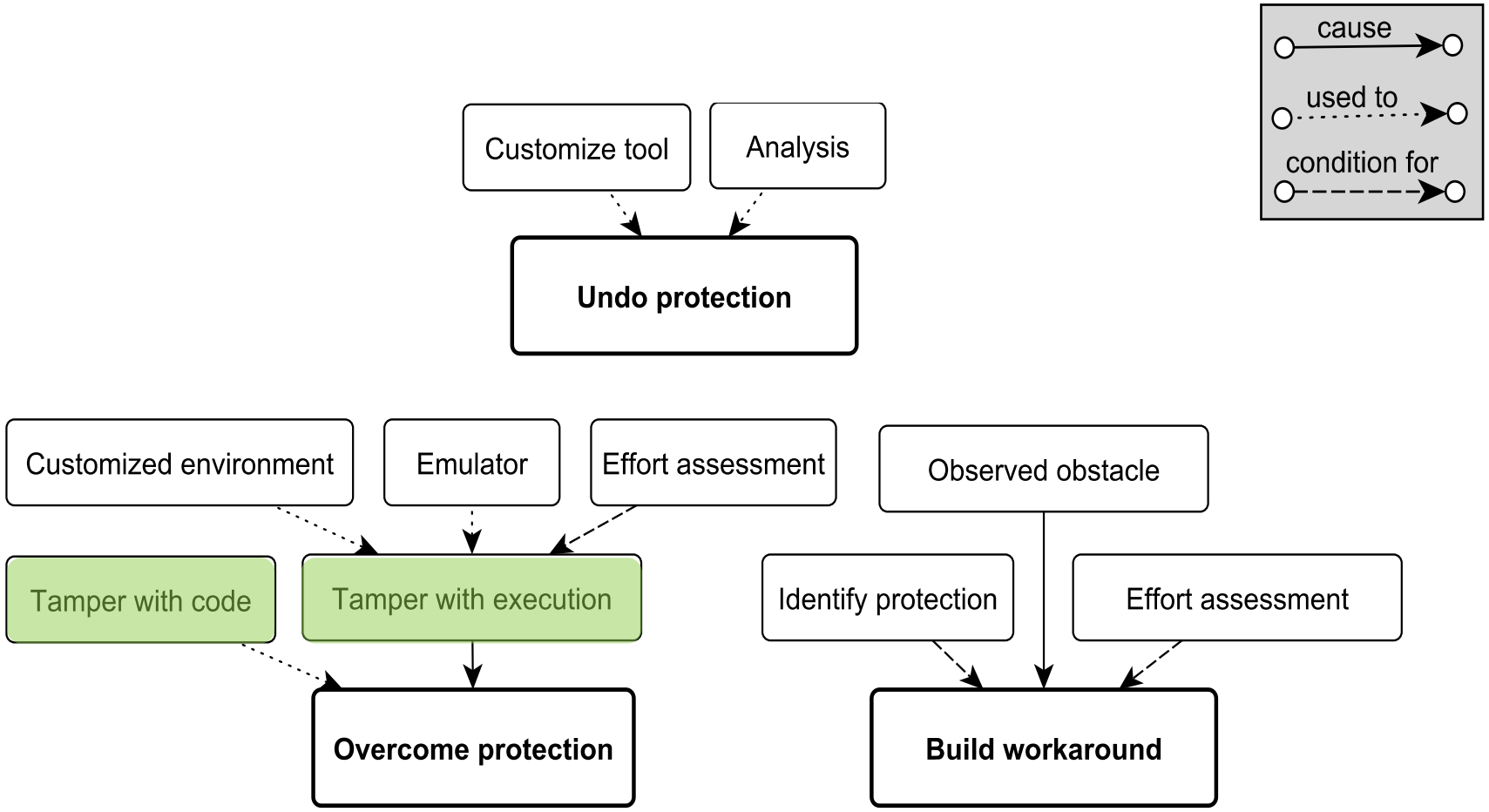
"Aside from the [omissis] added inconveniences [due to protections], execution environment requirements can also make an attacker's task much more difficult. [omissis] Things such as limitations on network access and maximum file size limitations caused problems during this exercise"

How hackers build attack strategies

Ca' Foscari University of Venice



How hackers work around & defeat protections



Summary: Experiments with Industry

Ca' Foscari University of Venice

- Validation of protections on real programs with best attackers
  - Effectiveness in delay attacks
- Validation of different combinations of protections
  - Protection mutually protect each other to offer no single starting point
- Real attack strategies with actual hacker tools
  - Continuously revise and change attack strategy
  - Prefer to circumvent than defeat protections
  - Adaptation based on found obstacles and conjectures about possible defenses
- Opportunities for improving protections
  - Extend boundaries of protections
  - Integration of off-line and on-line protections

Publications

Ca' Foscari University of Venice

The results have been published at

- IEEE International Conference on Program Comprehension (ICPC-2017), Buenos Aires
  - M Ceccato, P Tonella, C Basile, B Coppens, B De Sutter, P Falcarin, M Torchiano: How Professional Hackers Understand Protected Code while Performing Attack Tasks. In Proceedings of the 25th IEEE Intern. Conf. on Program Comprehension (ICPC-2017)
  - **The Best Paper Award at the conference and The ACM Distinguished paper Award**
- An extended journal version including analysis of reports of ASPIRE public challenge
  - Ceccato, M., Tonella, P., Basile, C., Falcarin, P., Torchiano, M., Coppens, B., & De Sutter, B. (2018). Understanding the behaviour of hackers while performing attack tasks in a professional setting and in a public challenge. Empirical Software Engineering, Springer, pp 1-47.
- Also presented at
  - UEL conference in London, June 2017
  - Dagsthul seminar, July 2017
  - OWASP invited talk, Cagliari, 2017

Conclusions

Open challenges

Ca' Foscari University of Venice

- Comprehensive attack model
- Automatic instantiation for application + assets
- Model the relation between attack steps and protections
- How to choose the best set of protections?
  - from possibly a very large set
  - applicable at many places in the code
  - tool-based metrics require running the tools
  - iterative methods or machine learning

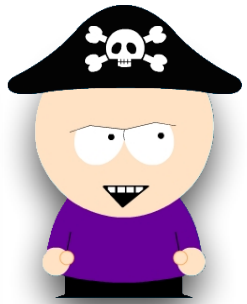
Missing: Validation (Collberg, DeSutter)

Ca' Foscari University of Venice

1. Build model from the behavior of real hackers:

Adversarial Model

- **X is hard**
- **Y is easy**

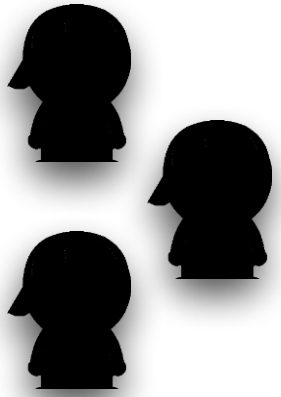


2. Correlate with potential metrics:

SCM_1

SCM_2

SCM_3



odd

even

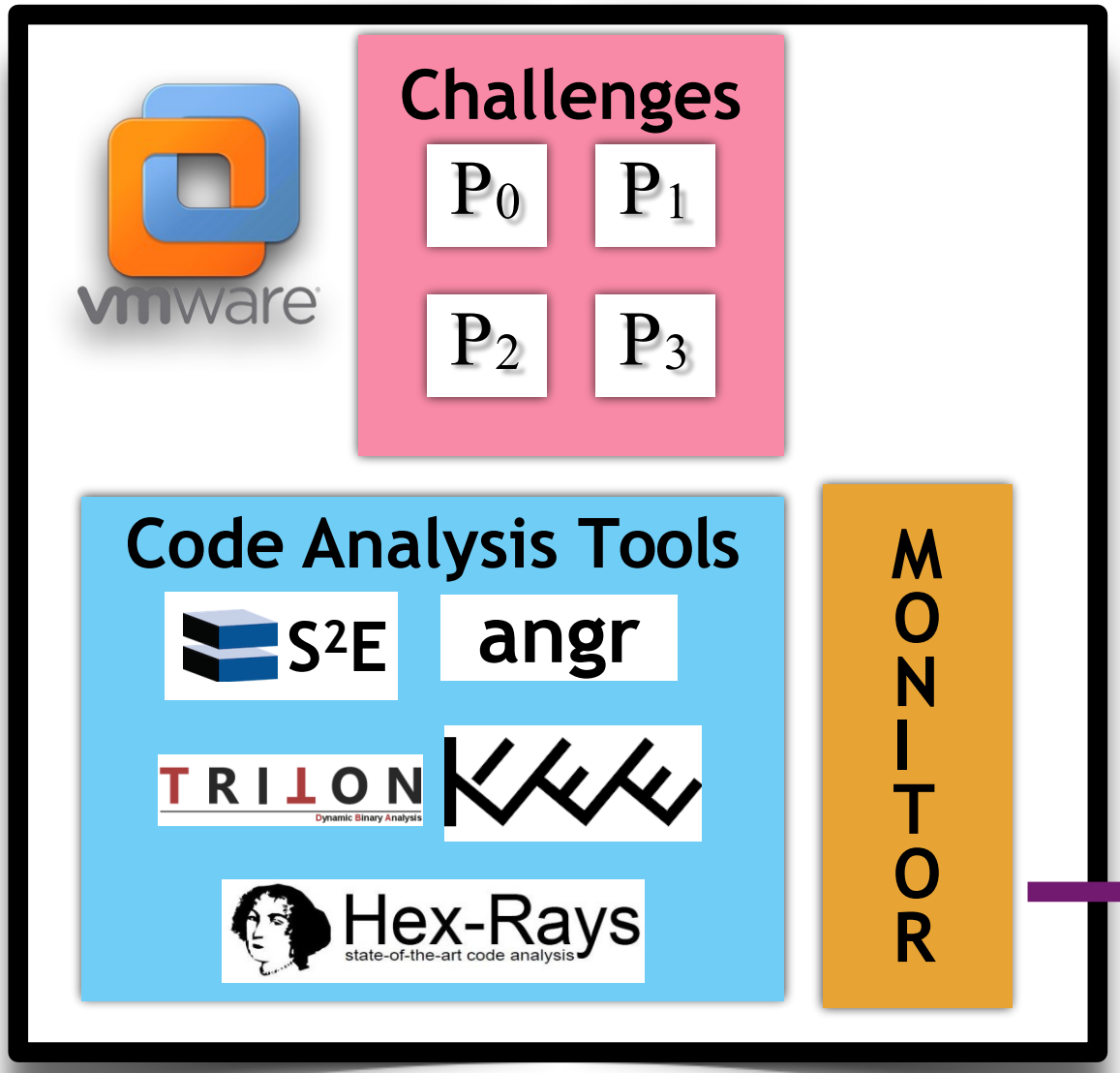
S²E angr

TRITON Dynamic Binary Analysis KLEE

Hex-Rays state-of-the-art code analysis

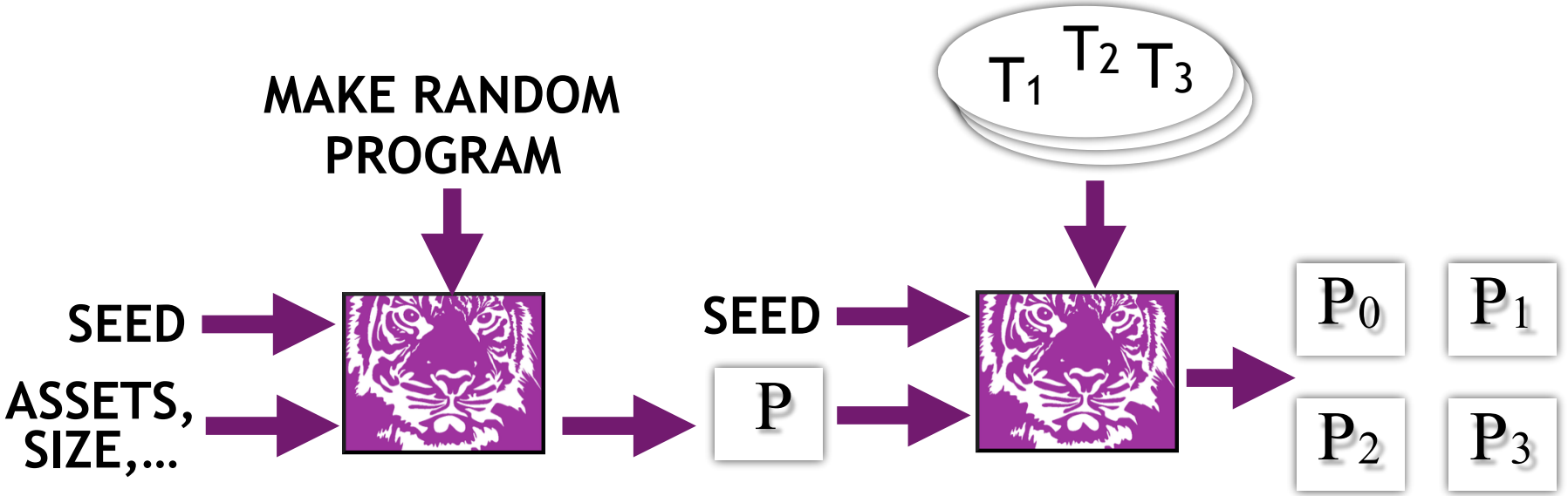
Adversarial Model Building

Ca' Foscari University of Venice



Model
- X is hard
- Y is easy

Generating Challenges



- Automatically generate many challenges
- Varying levels of complexity

Challenges So Far…

Ca' Foscari University of Venice

- Easiest challenge broken by Google engineer in 8 hours.

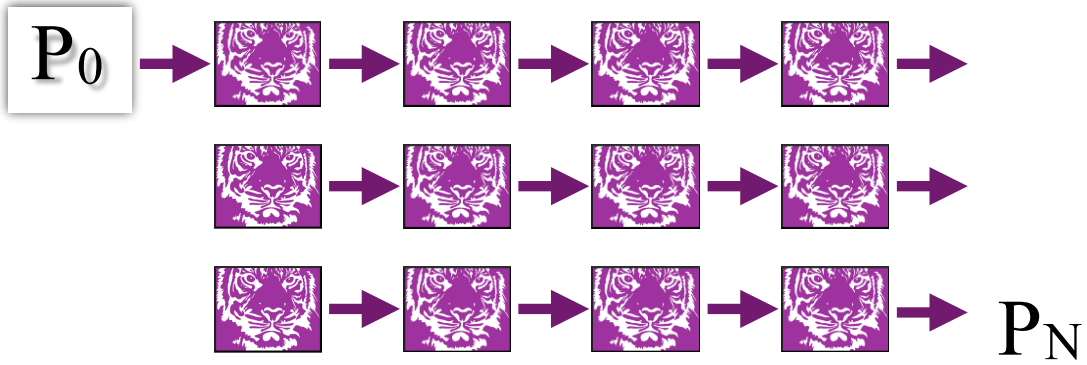
http://tigress.cs.arizona.edu/challenges.html

Cash and/or book prizes!

Security vs Performance

Ca' Foscari University of Venice

- Meeting security criteria without meeting performance criteria is not a solution in a MATE scenario.
- Arbitrary levels of protection, at arbitrary levels of slowdown, is easy:



Precision vs Performance

Ca' Foscari University of Venice

- Meeting precision criteria without meeting performance criteria is not a solution for anti-MATE analyses.
- Real programs are large, and analyses need to scale.
- Saying that an obfuscation falls against a particular analysis is meaningless without knowing the performance cost.

Dynamic Security

Ca' Foscari University of Venice

- Obfuscating transformations are primitives that provide time-limited protection.
- Updatable security can extend the protection they provide.
- All language-based obfuscations will break.
- Updatable security can increase the cost to the attacker.

Evaluation

Ca' Foscari University of Venice

- To make progress in this field, the community must settle on rigorous evaluation procedures.
- Evaluation is a mess — we need to fix this.
- Learn from public challenges.

MATE Predictions?

Ca' Foscari University of Venice

	Performance	Security	Scenarios
Hardware based			
Language based			
Crypto based			
Dynamic Renewable Security			

Which techniques will prevail?
Will they coexist, but in different scenarios?
Will we see combinations of techniques?