securityandtrust.lu

Security Testing: a key challenge for software engineering

Yves Le Traon, yves.letraon@uni.lu

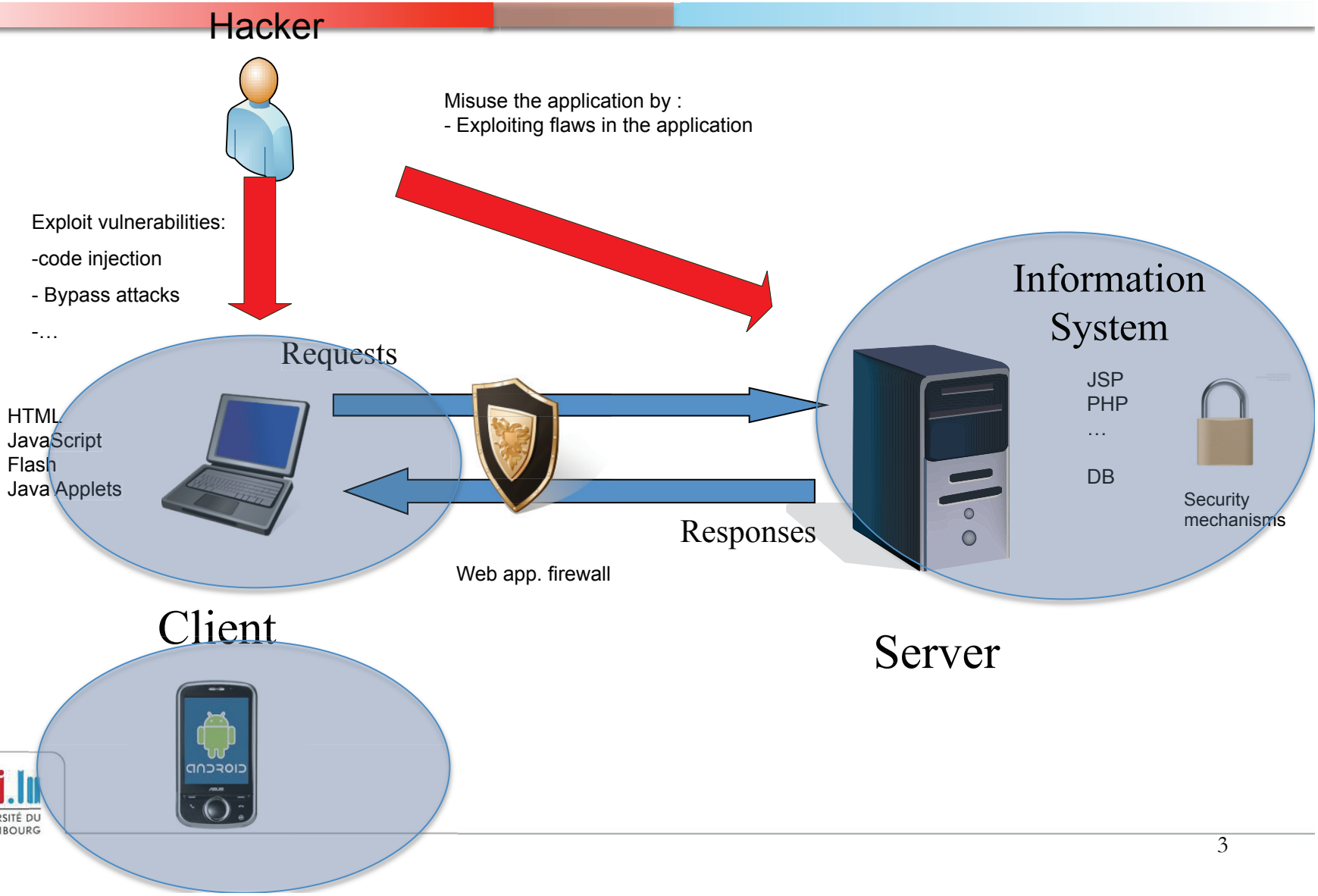
Professor, Univ of Luxembourg

SEcuRity and VALidation

Objectives of the presentation

- Promote applied research in systematic security testing
- How ?
  - Results of three research experiments
- Lessons learnt
- Open challenges

The perimeter of the talk



Overview

- Testing security?
- About XSS, web browsers and regression testing
- About mobile apps attack surface (Android)
- About internal information system security
- About emerging security testing challenges

About the team

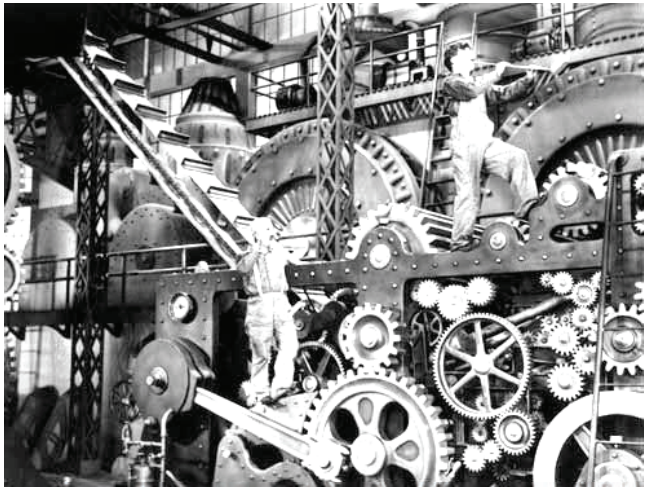
Applied research directions

- Modelling and Models@runtime
  - Keeping models and infrastructures synchronized
  - Model changes → deployment
  - Dynamic adaptation
- Security & Privacy
  - Access-control/permission-based
  - Usage control/obligations
  - Model-driven security
  - Automated implementation of security mechanisms
  - Socio-technical concerns/ conviviality
- Software Testing
  - Model-Based testing
  - Security testing
  - Cloud/large scale /DDoS

Activities of the team

Theory and formalization
Model-checking
Verification

Model-driven engineering
Model Composition



Design and code

Run
Experiments

Testing

About testing security

Looking for bugs and other errors

9/9

0800 Arctan started
1000 " stopped - arctan ✓ { 1.2700 9.037 847 025
9.037 846 795 correct
13" u/c (032) MP - MC ~~2.130476415 (033)~~ 4.615925059(-2)
(033) PRO 2 2.130476415
correct 2.130676415
Relays 6-2 in 033 failed special speed test
In relay " 10.00 test.
Relays changed
1100 Started Cosine Tape (Sine check)
1525 Started Multy Adder Test.

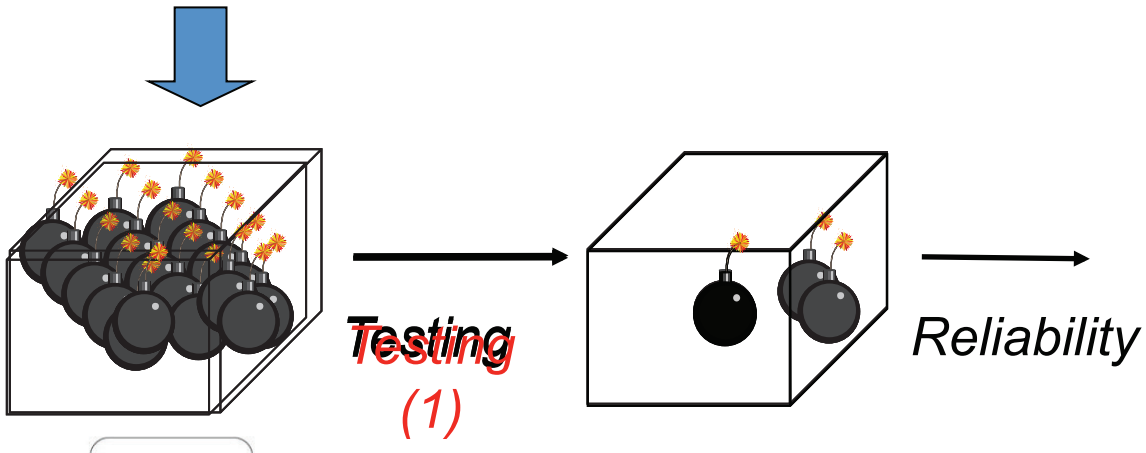
1545 Relay #70 Panel F
(moth) in relay.
First actual case of bug being found.
~~1700~~/1630 Arctangent started.
1700 closed down.

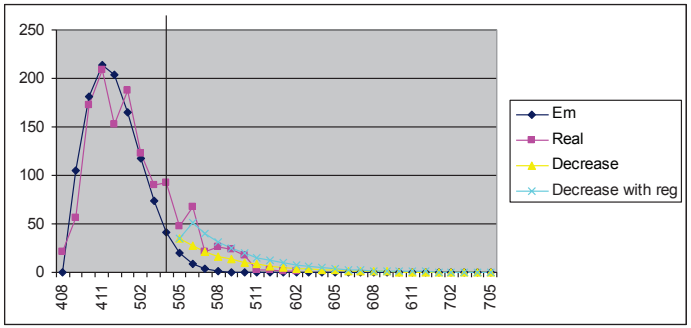
Software testing: cost and trust

Testing → *Detecting inconsistencies between implementation and specification*

Design for testability

(2) Design for trust





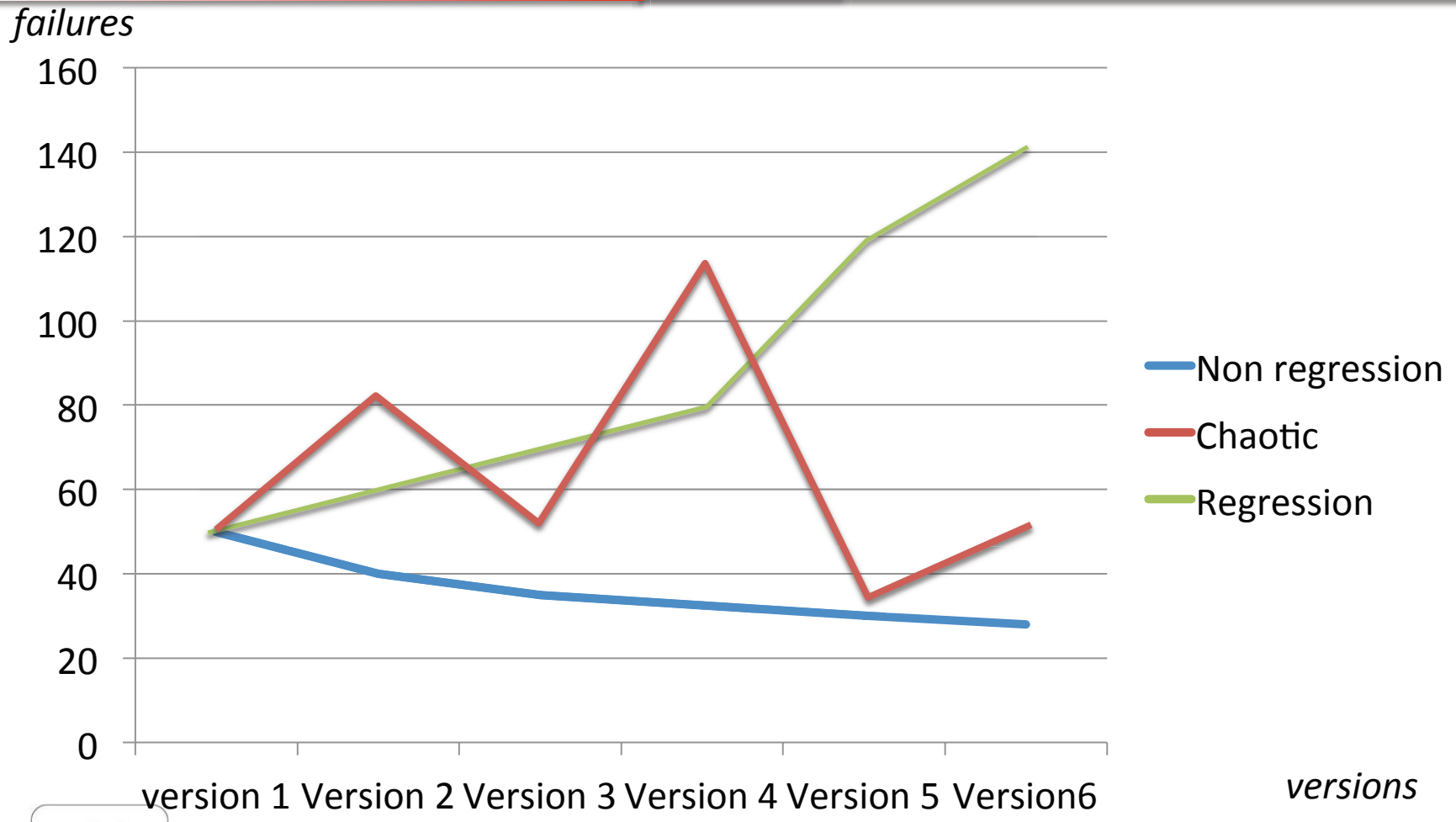
Classical testing issues

About XSS, web browsers and regression testing

Regression testing: From chaos to order

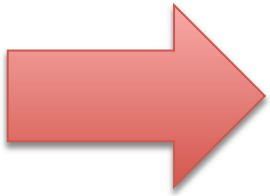
UNIVERSITÉ DU LUXEMBOURG

Software regression



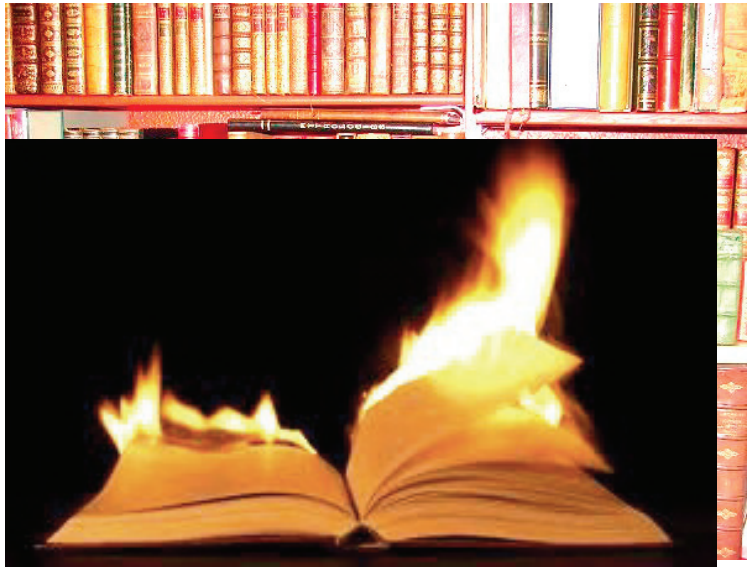
Security and software engineering

- Program understanding/Reverse engineering



About mobile apps attack surface (Android)

Software testing vs. Security Testing



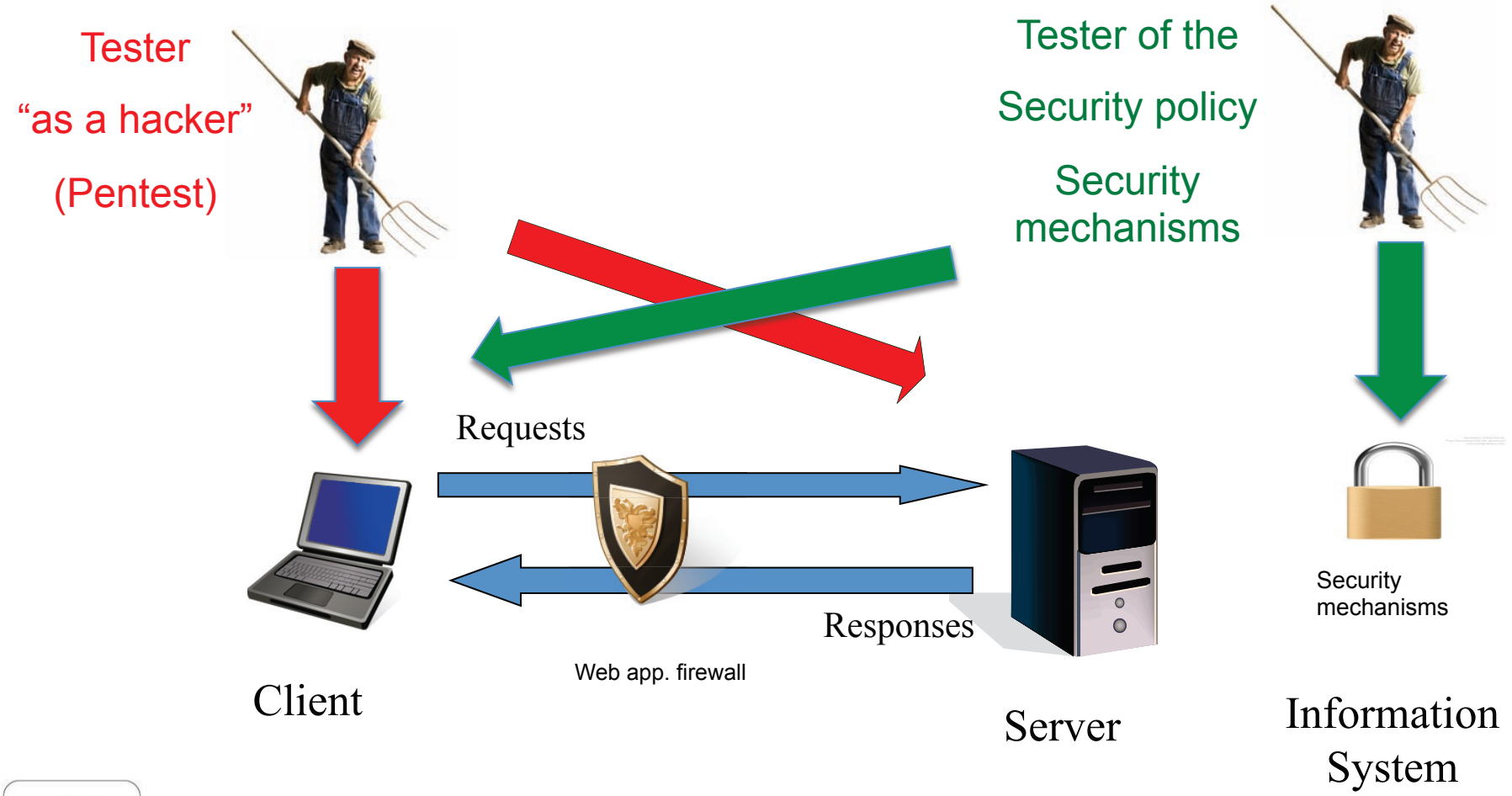
Book:
Open
Close
Read

Functional testing:
→ It works as expected

Robustness testing:
→ Then should still work

Security testing:
→ Very particular robustness
→ Then should still work

Security testing is two fold



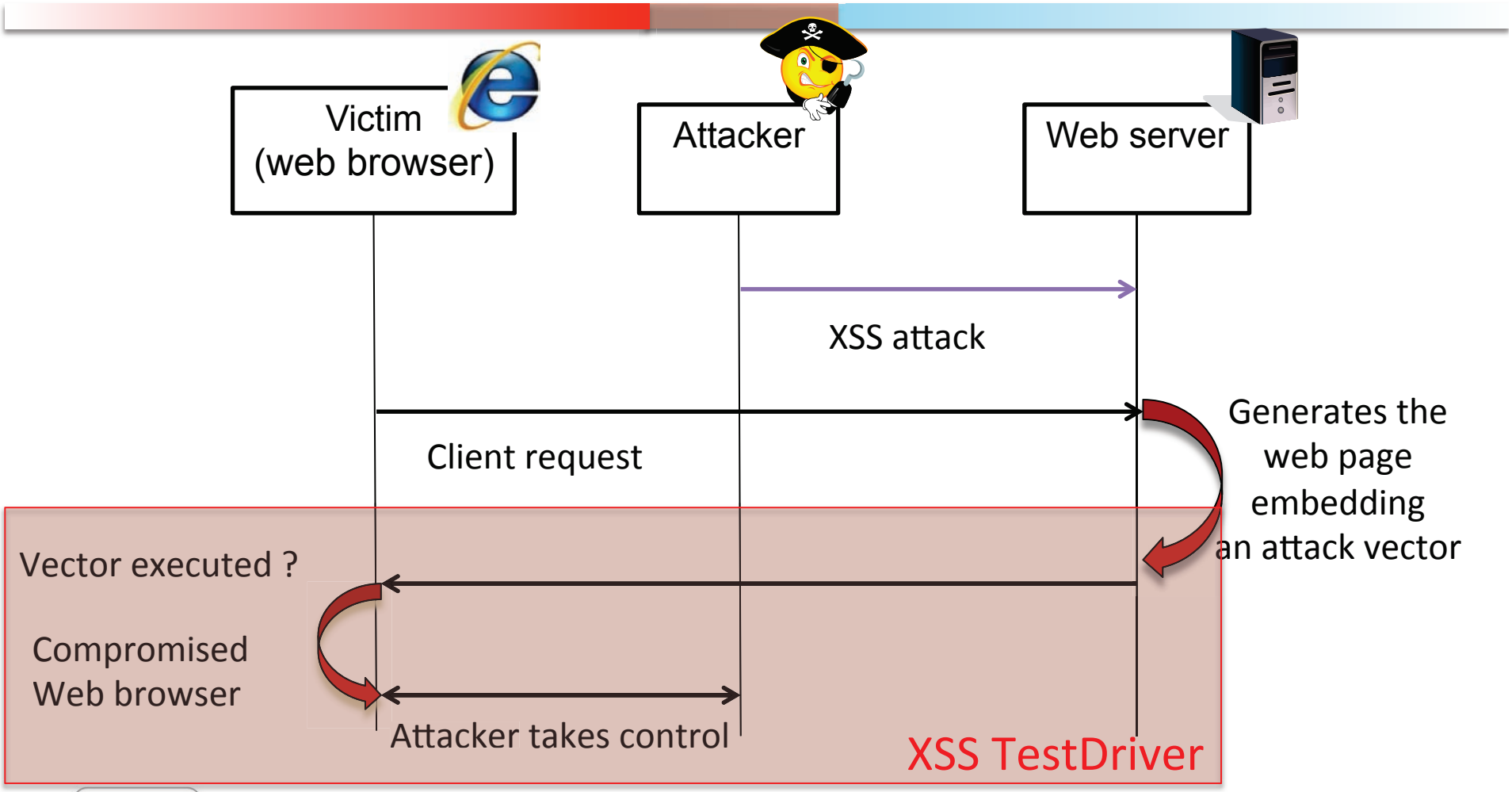
Security testing is two fold

- Testing "as a hacker"
  - Adapt known attacks
  - Generate new attack vectors and publish them
  - Fuzz testing
- Testing a security policy implementation
  - "normal test"
  - The specification = the security policy (access control/ usage control …)
  - Security mechanisms
    - Where are they?

About XSS, web browsers and regression testing

Erwan Abgrall – PhD – Kereval, France
Sylvain Gombault – researcher – telecom Bretagne

Attacking process



Impact of a XSS attack

- Attacker may execute a script
  - Redirecting the client,
  - Session theft
  - Program install
  - Key logging etc.
- Many worms propagated through social networks (twitter, facebook, myspace (Samy)) were due to a vulnerability exploited with XSS.

- SaaS Cloud apps based on Ajax / Web 2.0 technologies increase the number of potential targets

Vector / Payload / Attack

- Vector: piece of HTML code enabling JavaScript code execution
- Payload: The javascript code to be executed

```
<script>alert(1)</script>
```

- Attack: Injection that makes the server generate the vector

  http://victime.fr/search?id='></input><script>alert(1)</script>

Existing attack frameworks

- XSS attack frameworks are already available
  - BeEF
    - http://beefproject.com/
  - XSSF
    - http://blog.conixsecurity.fr/?p=436
  - XSSER
    - http://xsser.sourceforge.net/
- XSS test drivers focuses on test vectors not on payloads and vulnerability exploits

Attack surface of a web browser

- Code within a web browser that can be run by an attacker
  - The set of executable vectors in a given browser

- XSS Test Driver aims at exercising this code…

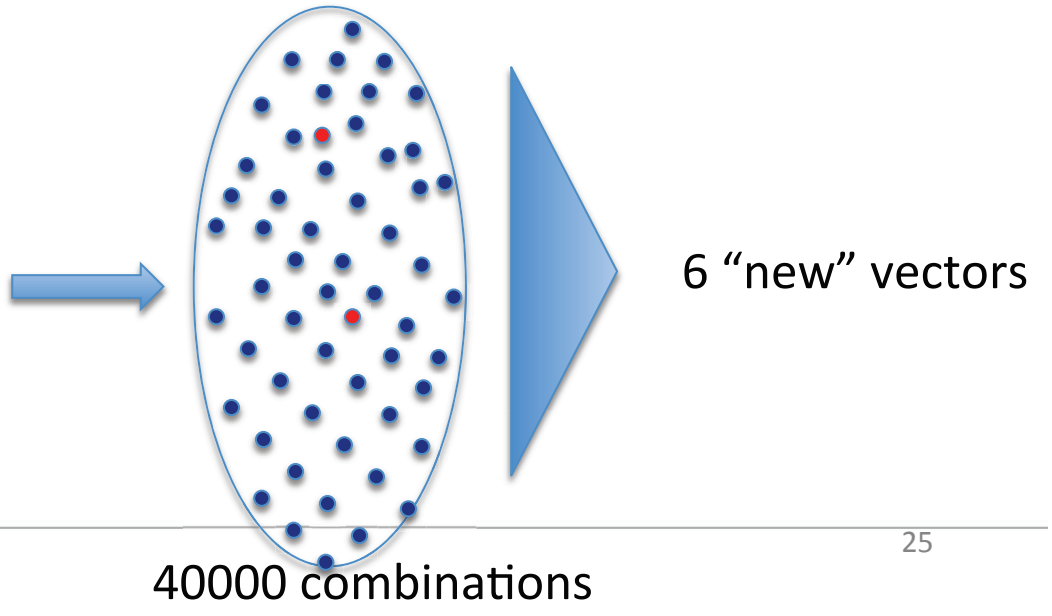
Selection of XSS test vectors

Referenced vectors

XSS Cheat Sheet : http://ha.ckers.org/xss.html
html5 xss cheat sheet: http://heideri.ch/jso/
UTF-7 XSS Cheat Sheet: http://openmya.hacker.jp/hasegawa/security/utf7cs.html
Final benchmark can be found
http://xss.labosecu.rennes.telecom-bretagne.eu/

New vectors generation

{html4tag} X {property} X {JScall}



Example 1

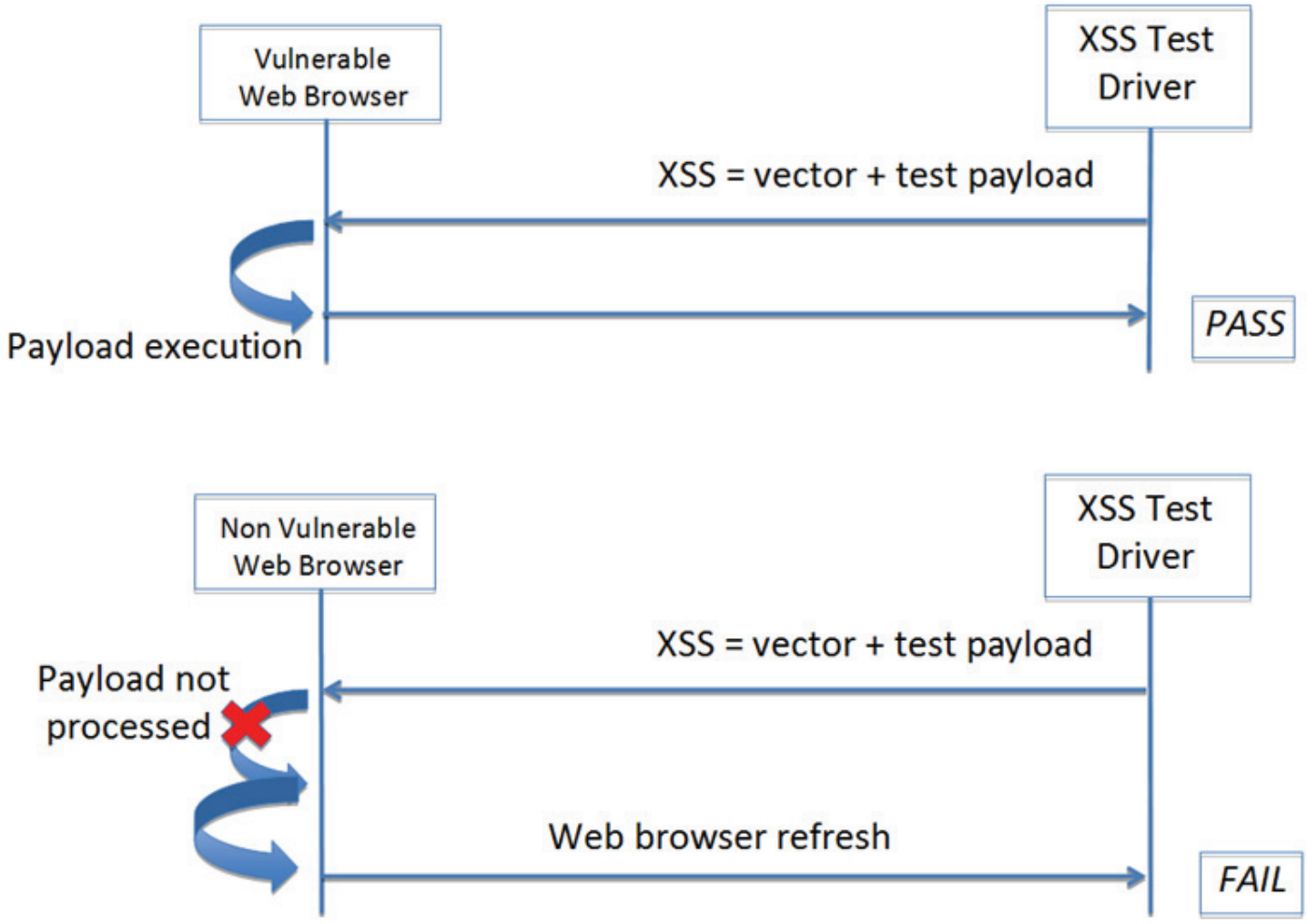
- svg based xss <g> onload
- http://html5sec.org/#11
- <svg xmlns="http://www.w3.org/2000/svg"><g onload="javascript:%(eval_payload)s"></g></svg>

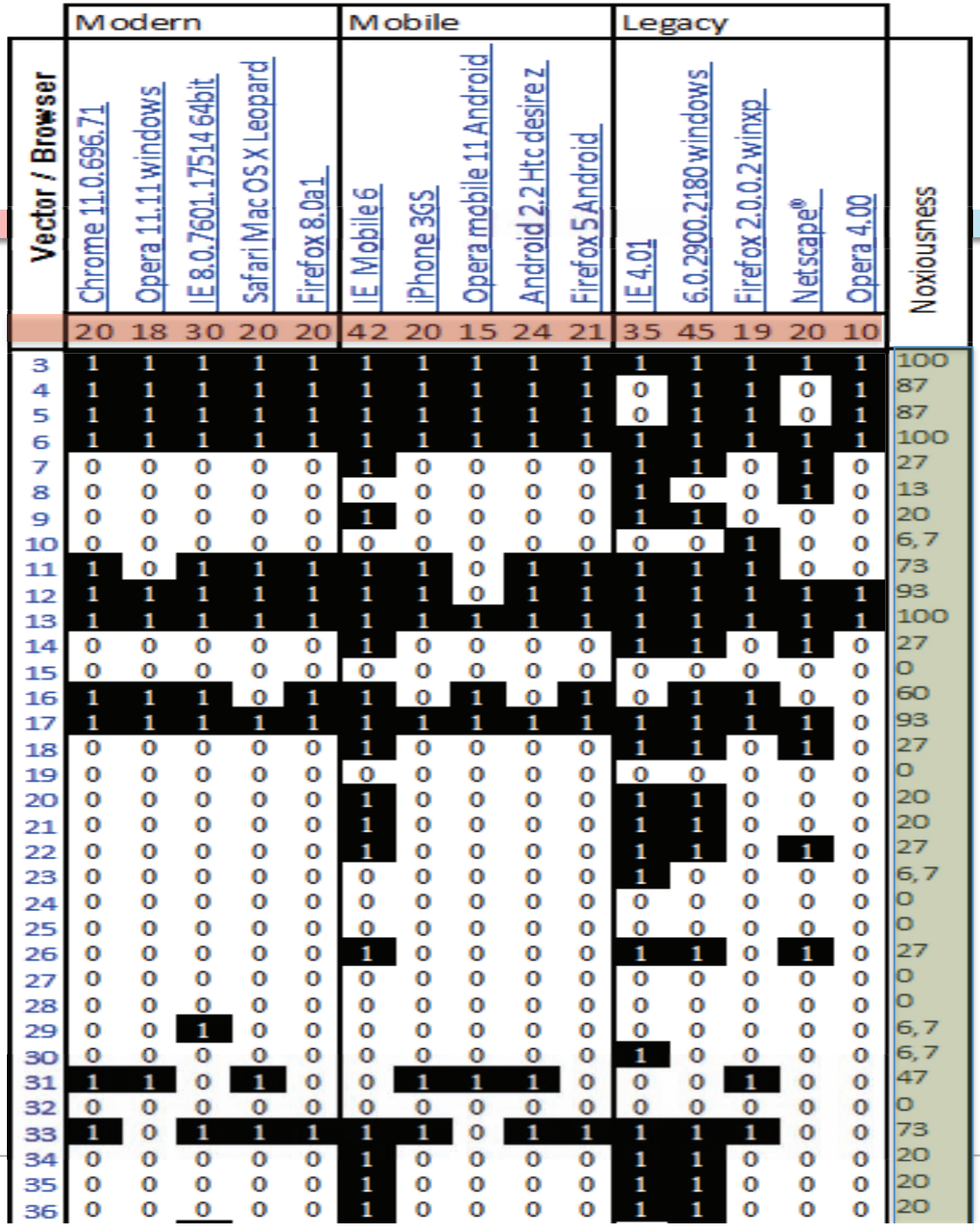
Example 2

- SVG chameleon behavior via embedded XSLT

- http://html5sec.org/#125<?xml version="1.0"?> <?xml-stylesheet type="text/xml" href="#stylesheet"?> <!DOCTYPE doc [<!ATTLIST xsl:stylesheet id ID #REQUIRED>]> <svg xmlns="http://www.w3.org/2000/svg"> <xsl:stylesheet id="stylesheet" version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"> <xsl:template match="/"> <iframe xmlns="http://www.w3.org/1999/xhtml" src="javascript:%(eval_payload)s"></iframe> </xsl:template> </xsl:stylesheet> <circle fill="red" r="40"></circle> </svg>

Web browser test mechanism





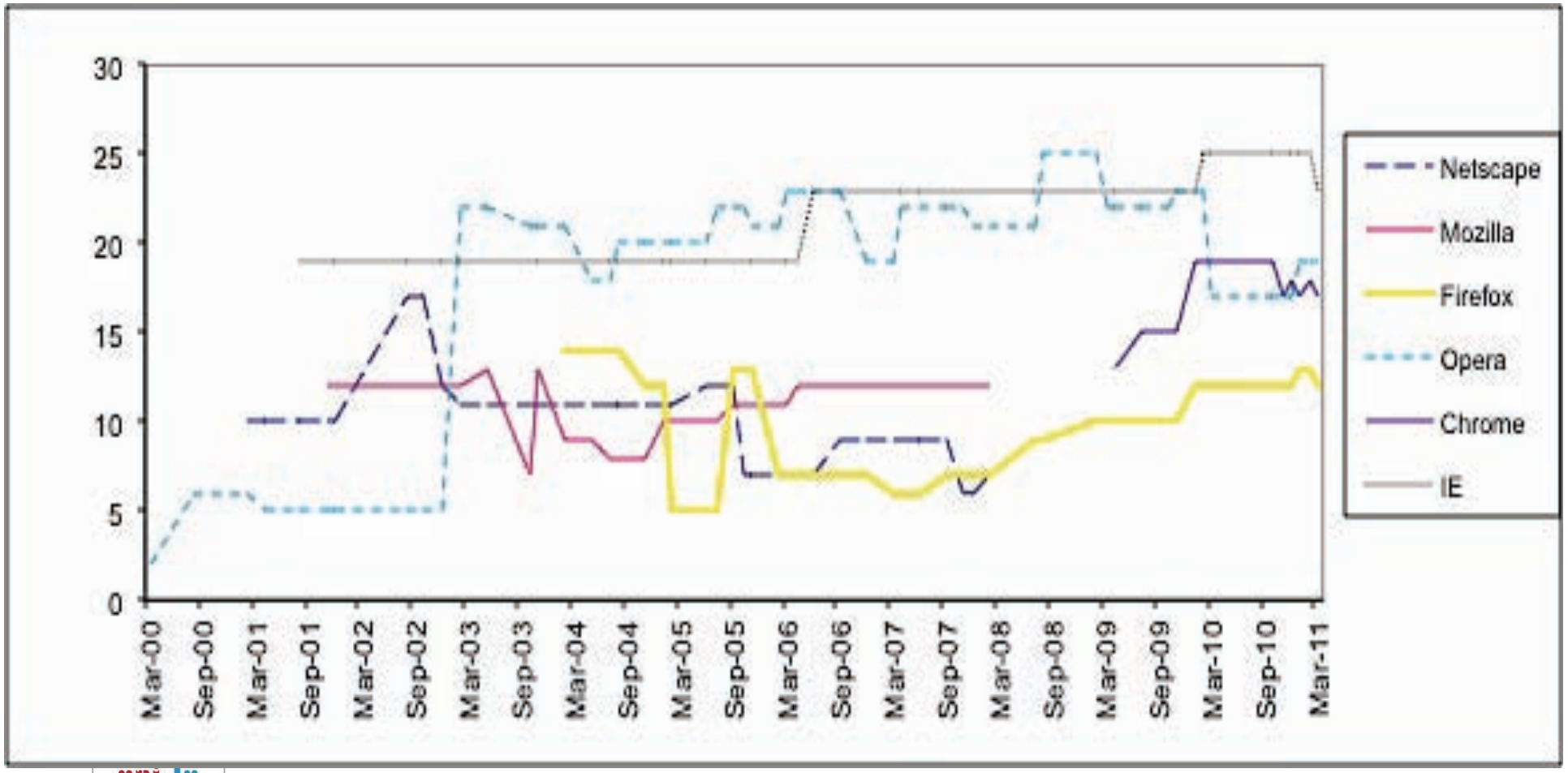
	Modern					Mobile					Legacy					
Vector / Browser	Chrome 11.0.696.71	Opera 11.11 windows	IE 8.0.7601.17514 64bit	Safari Mac OS X Leopard	Firefox 8.0a1	IE Mobile 6	iPhone 3GS	Opera mobile 11 Android	Android 2.2 Htc desire z	Firefox 5 Android	IE 4.01	6.0.2900.2180 windows	Firefox 2.0.0.2 winxp	Netscape®	Opera 4.00	Noxiousness
	20	18	30	20	20	42	20	15	24	21	35	45	19	20	10	
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	100
4	1	1	1	1	1	1	1	1	1	1	0	1	1	0	1	87
5	1	1	1	1	1	1	1	1	1	1	0	1	1	0	1	87
6	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	100
7	0	0	0	0	0	1	0	0	0	0	1	1	0	1	0	27
8	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	13
9	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	20
10	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	6,7
11	1	0	1	1	1	1	1	0	1	1	1	1	1	0	0	73
12	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	93
13	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	100
14	0	0	0	0	0	1	0	0	0	0	1	1	0	1	0	27
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	1	1	1	0	1	1	0	1	0	1	0	1	1	0	0	60
17	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	93
18	0	0	0	0	0	1	0	0	0	0	1	1	0	1	0	27
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	20
21	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	20
22	0	0	0	0	0	1	0	0	0	0	1	1	0	1	0	27
23	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	6,7
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	1	0	0	0	0	1	1	0	1	0	27
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	6,7
30	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	6,7
31	1	1	0	1	0	0	1	1	1	0	0	0	1	0	0	47
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	1	0	1	1	1	1	1	0	1	1	1	1	1	0	0	73
34	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	20
35	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	20
36	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	20

Are mobiles protected?

NO!

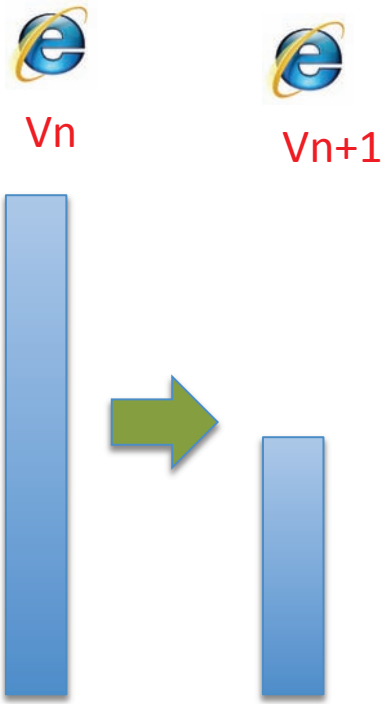
- Web browser is the n°1 application for smartphones
- These browsers inherit the defects from their parent browsers
  - Webkit / Gecko / Presto …
- They even access more information
    - Data URI & specific functionalities
    - HTML Storage
    - Saved Identifiers, since typing password is tedious on mobiles
    - …Camera API, Vibrator API, Contacts API – WebAPI…

Attack surface over time

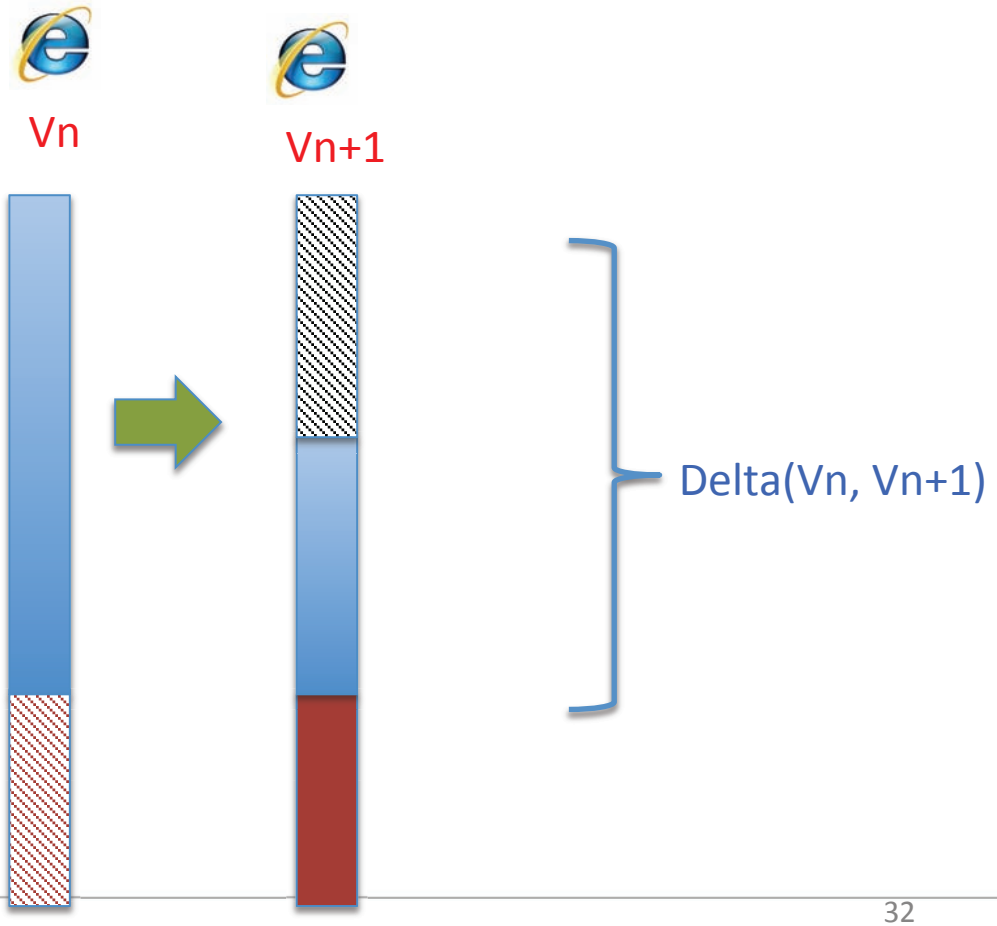


Regression is about deltas

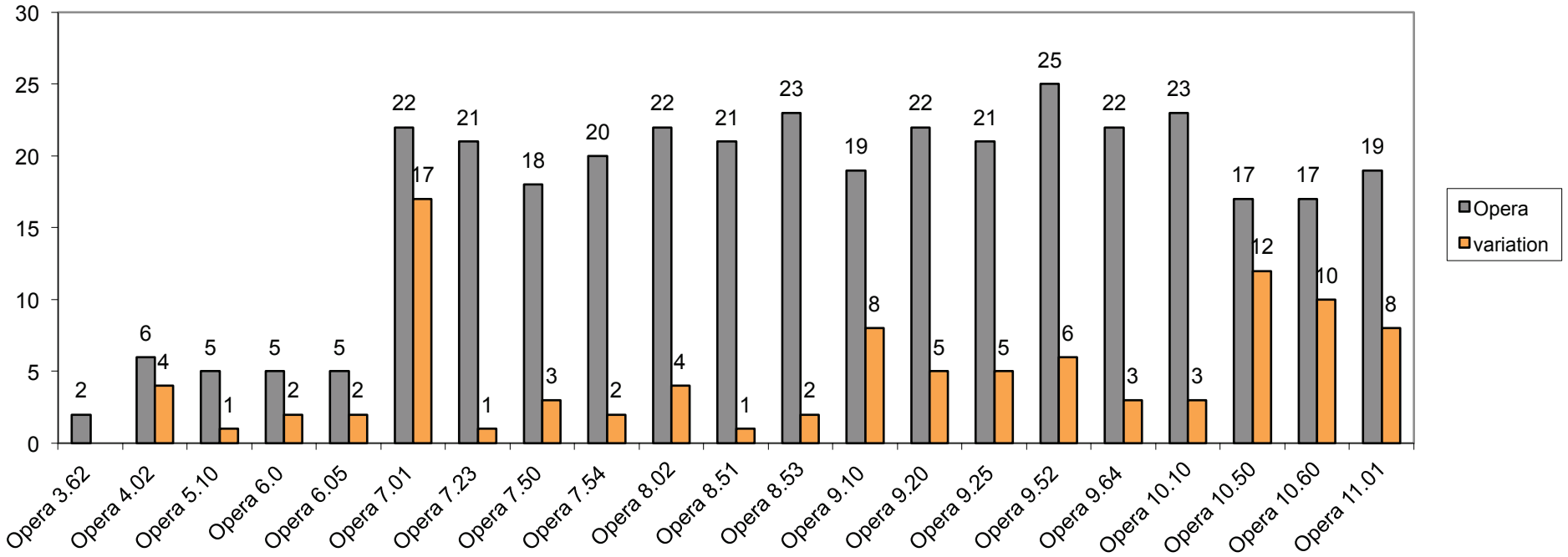
Ideally: convergence



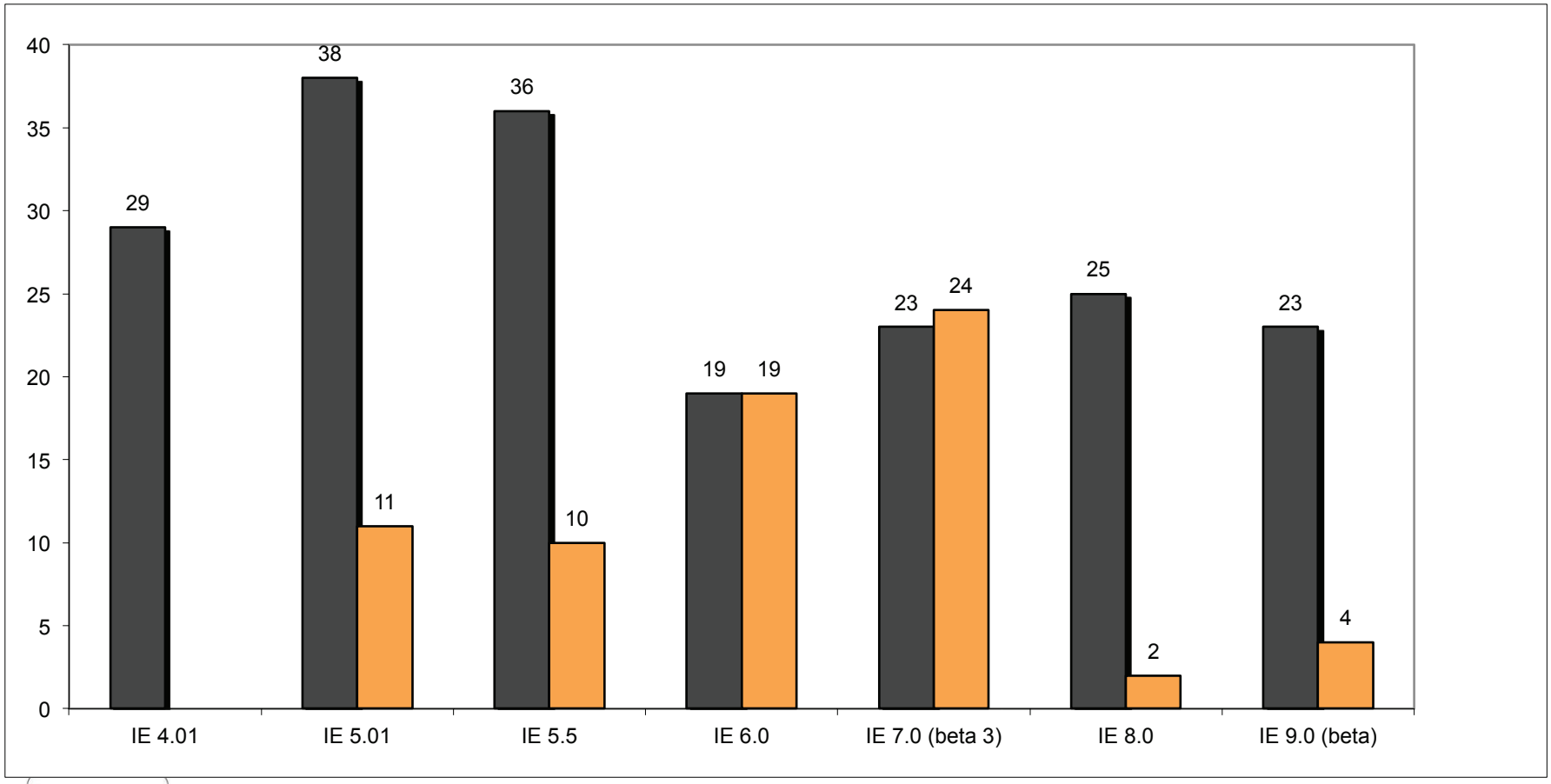
In practice



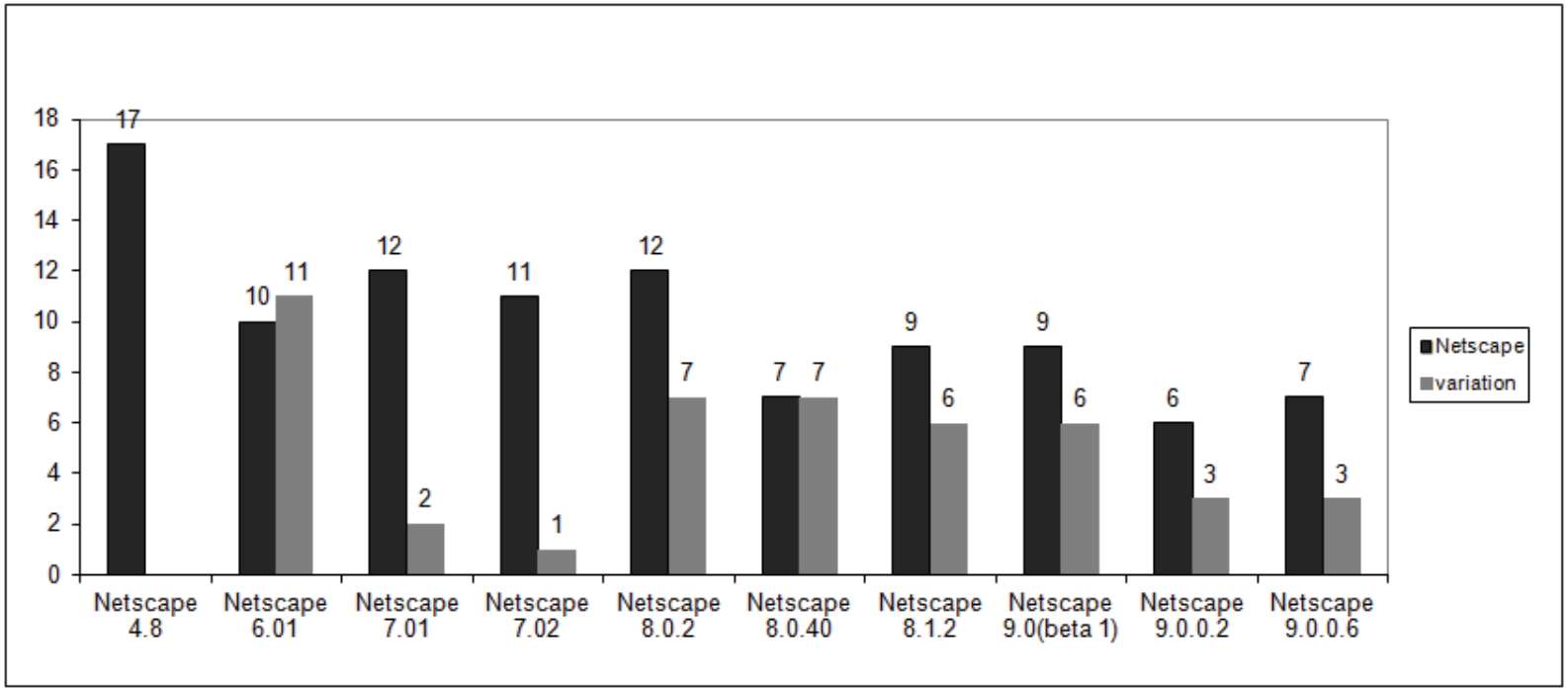
Opera


30
25
20
15
10
5
0
2
6
4
5
1
5
2
5
2
22
17
21
1
18
3
20
2
22
4
21
1
23
2
19
8
22
5
21
5
25
6
22
3
23
3
17
12
17
10
19
8
Opera 3.62
Opera 4.02
Opera 5.10
Opera 6.0
Opera 6.05
Opera 7.01
Opera 7.23
Opera 7.50
Opera 7.54
Opera 8.02
Opera 8.51
Opera 8.53
Opera 9.10
Opera 9.20
Opera 9.25
Opera 9.52
Opera 9.64
Opera 10.10
Opera 10.50
Opera 10.60
Opera 11.01
Opera
variation

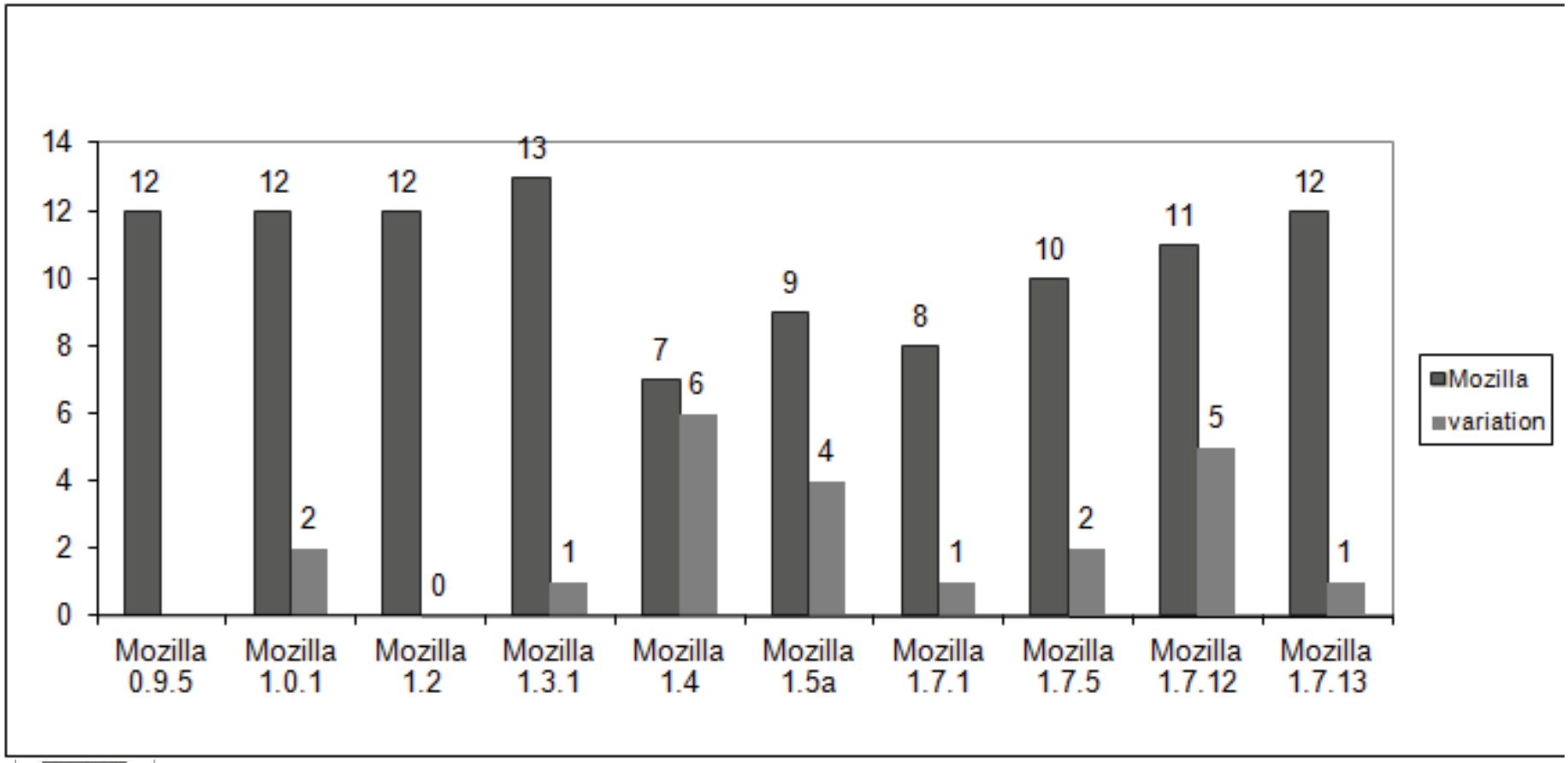
Internet Explorer



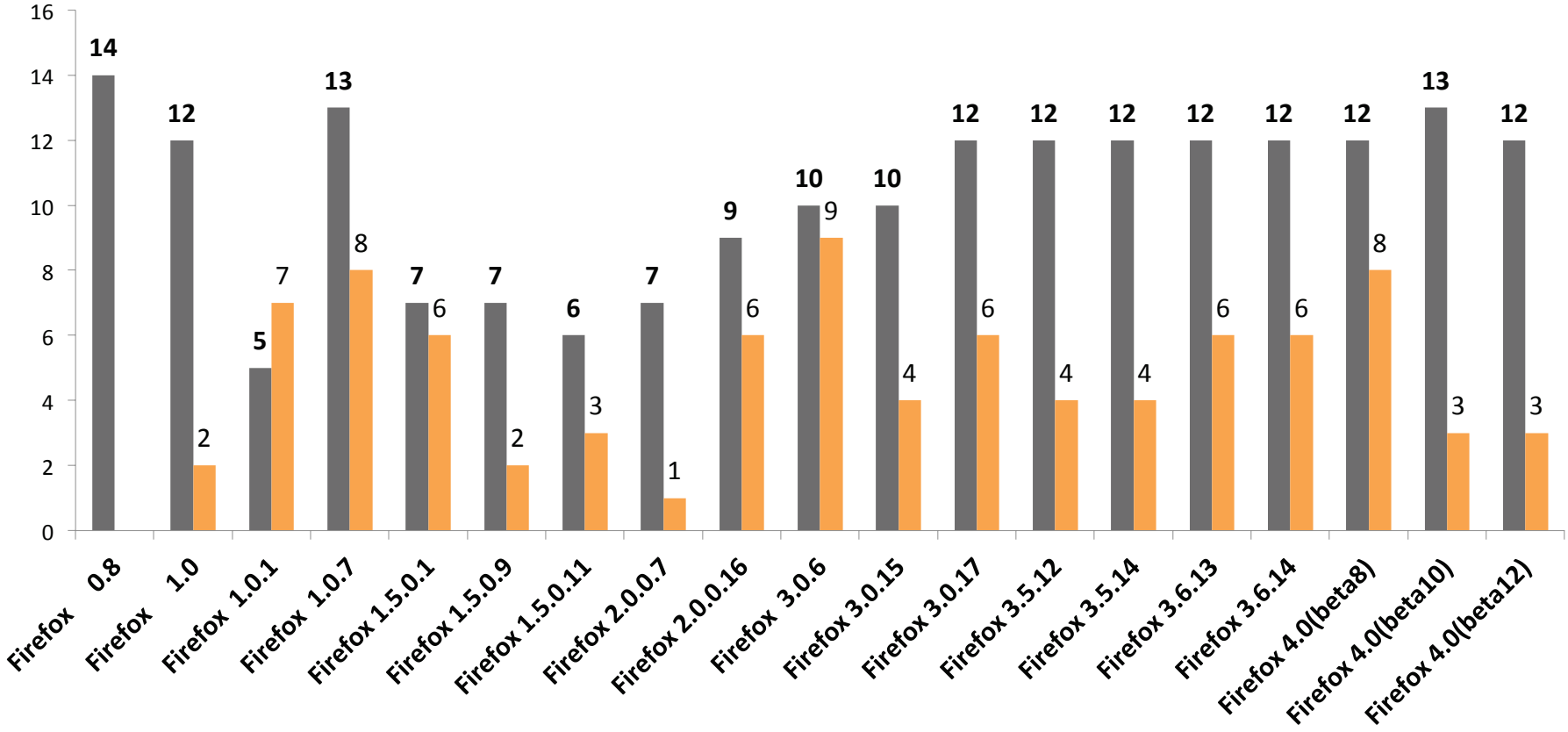
Netscape


18
16
14
12
10
8
6
4
2
0
17
10
11
12
2
11
1
12
7
7
7
9
6
9
6
6
3
7
3
Netscape 4.8
Netscape 6.01
Netscape 7.01
Netscape 7.02
Netscape 8.0.2
Netscape 8.0.40
Netscape 8.1.2
Netscape 9.0(beta 1)
Netscape 9.0.0.2
Netscape 9.0.0.6
Netscape
variation

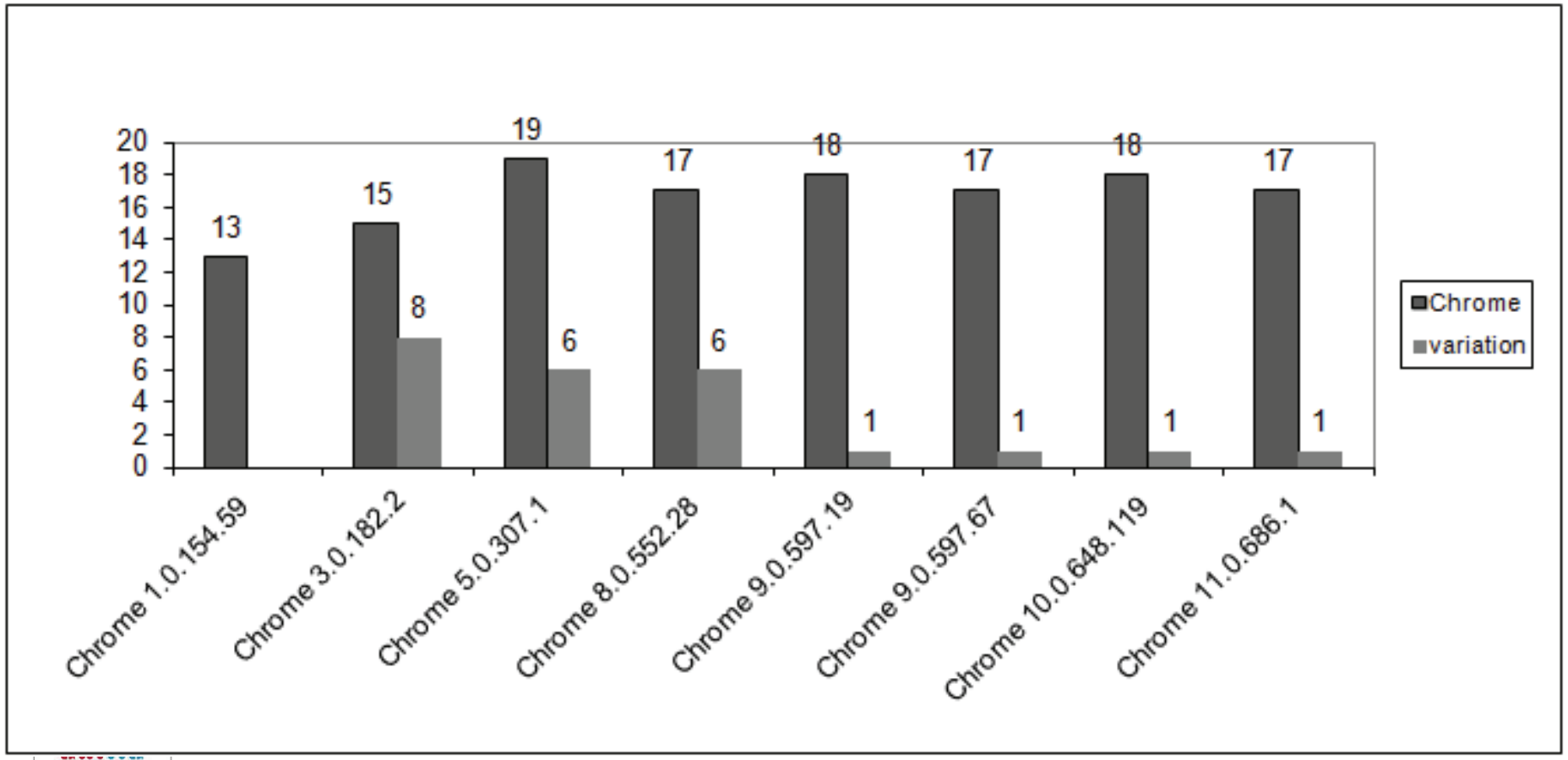
Mozilla


14
12
10
8
6
4
2
0
12
12
12
13
7
6
9
4
8
1
10
2
11
5
12
1
0
2
1
Mozilla 0.9.5
Mozilla 1.0.1
Mozilla 1.2
Mozilla 1.3.1
Mozilla 1.4
Mozilla 1.5a
Mozilla 1.7.1
Mozilla 1.7.5
Mozilla 1.7.12
Mozilla 1.7.13
Mozilla
variation

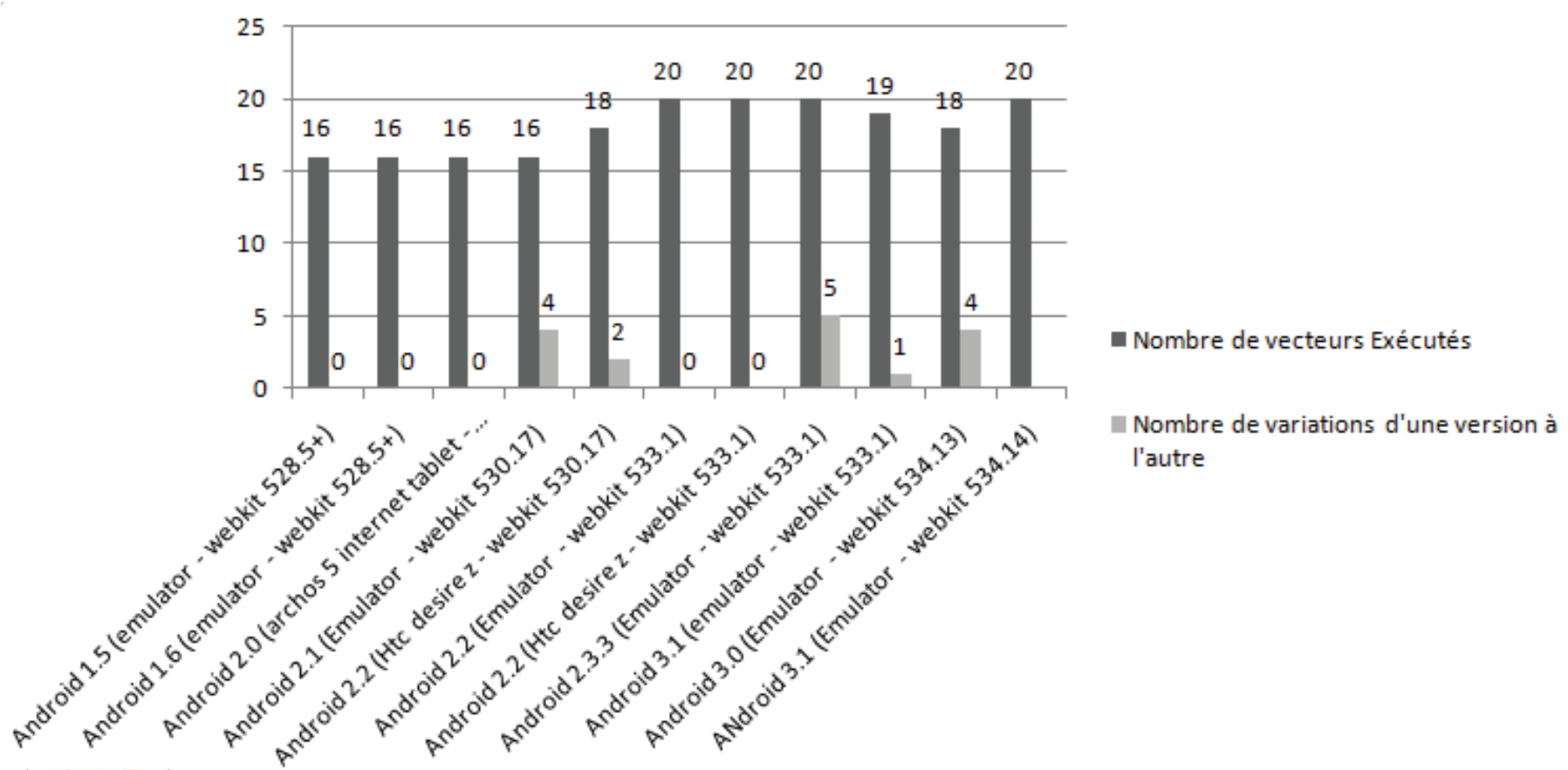
Firefox


16
14
12
10
8
6
4
2
0
14
12
2
5
7
13
8
7
6
7
2
6
3
7
1
9
6
10
9
10
4
12
6
12
4
12
4
12
6
12
6
12
8
13
3
12
3
Firefox 0.8
Firefox 1.0
Firefox 1.0.1
Firefox 1.0.7
Firefox 1.5.0.1
Firefox 1.5.0.9
Firefox 1.5.0.11
Firefox 2.0.0.7
Firefox 2.0.0.16
Firefox 3.0.6
Firefox 3.0.15
Firefox 3.0.17
Firefox 3.5.12
Firefox 3.5.14
Firefox 3.6.13
Firefox 3.6.14
Firefox 4.0(beta8)
Firefox 4.0(beta10)
Firefox 4.0(beta12)

Chrome


20
18
16
14
12
10
8
6
4
2
0
13
15
8
19
6
17
6
18
1
17
1
18
1
17
1
Chrome
variation
Chrome 1.0.154.59
Chrome 3.0.182.2
Chrome 5.0.307.1
Chrome 8.0.552.28
Chrome 9.0.597.19
Chrome 9.0.597.67
Chrome 10.0.648.119
Chrome 11.0.686.1

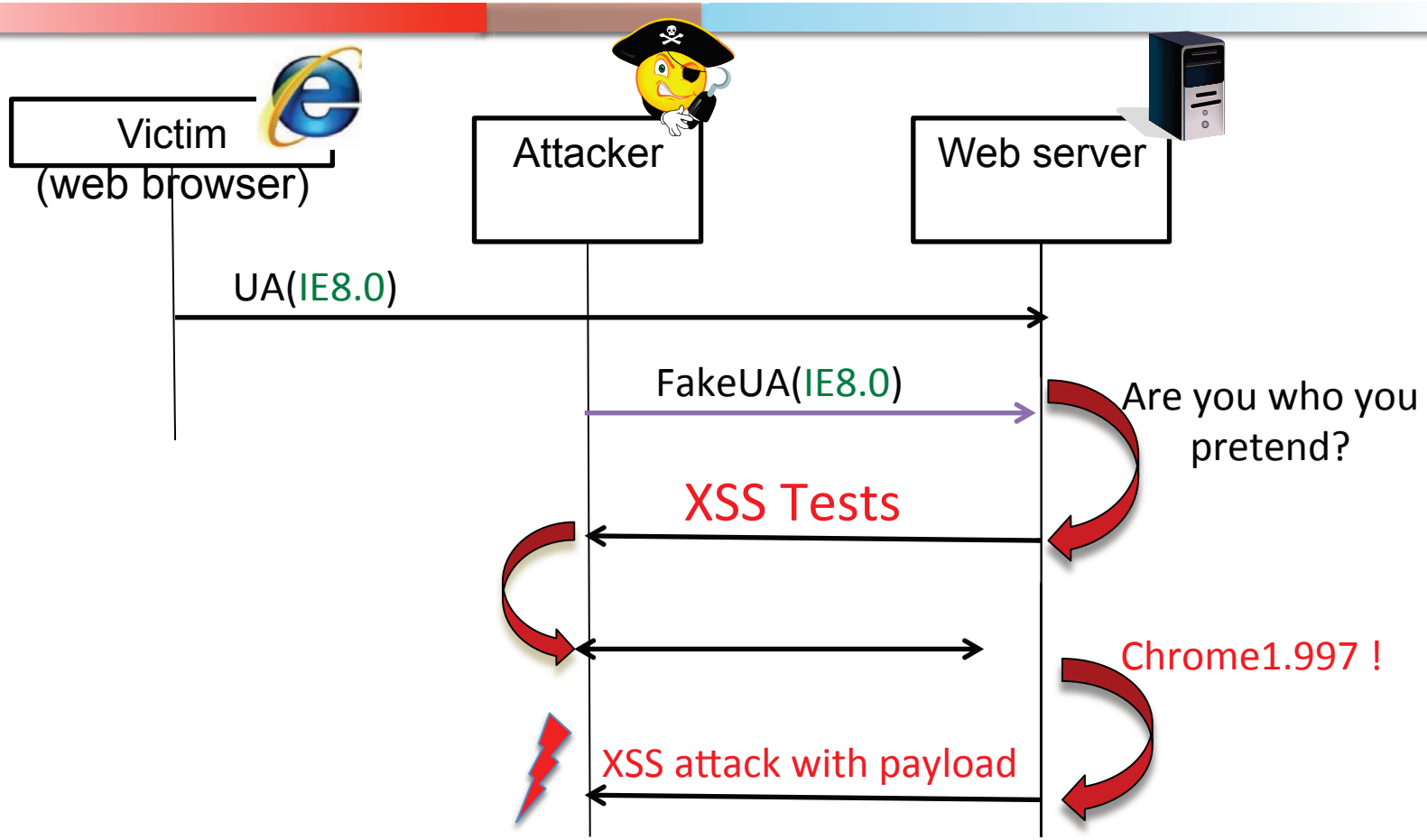
Android browser



One step further: Test for counter attacking

- A web browser differs from one another by its many features, one of them being its specific sensitivity to XSS attack vectors.
  - identifying a fake user-agent
  - determining the exact nature of an attacker's web browser
- for protecting and possibly counter-attacking.

Counter-attacking process

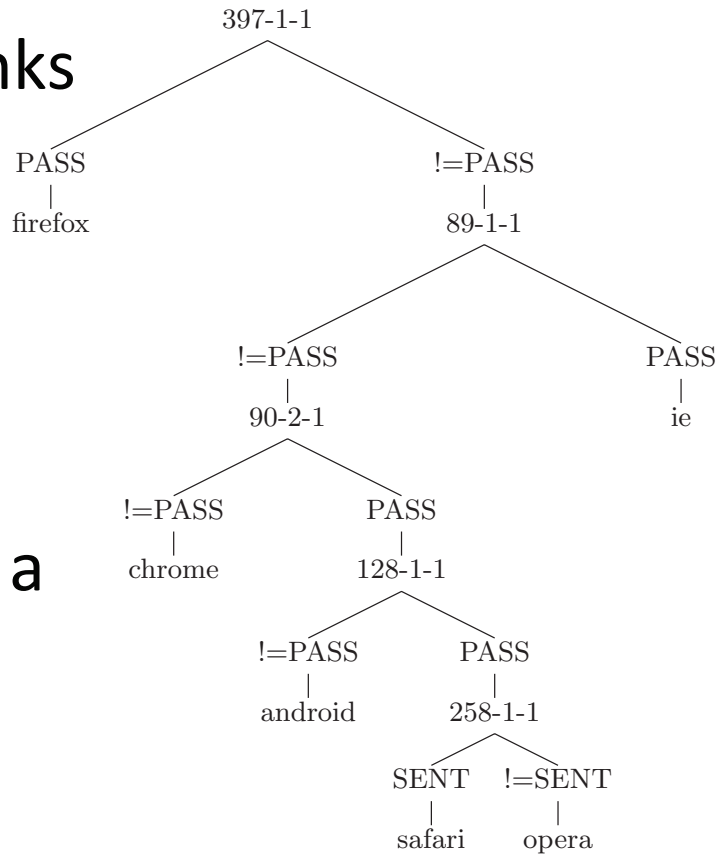


Test Method for attacker identification

- Use the reaction of a given web browser to such known XSS vectors as a signature
- identifies it precisely (family and version)
- Test driven web browser fingerprinting

Results

- the exact version of a web browser (out of 77) can be determined thanks to its signature (71% of accuracy).



- 6 XSS test vectors are sufficient to quickly determine the exact family a web browser belongs to, with an accuracy of 98.6 %

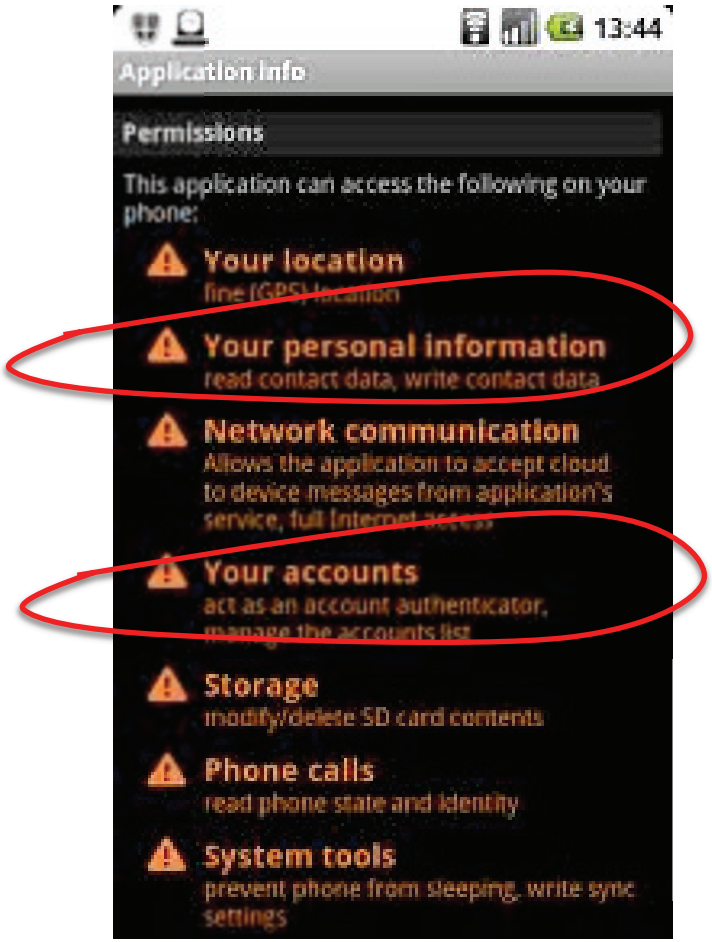
Lessons learnt and challenges

- No obvious systematic regression testing strategy for security
- Urgent need for
  - A tooled environment to systematically run regression tests
  - An updated benchmark of XSS vectors
- Research Challenges
  - Automate the generation of test vectors
    - Collaborative FuzzTesting: Shazzer
    - Possibly MBT?
  - Using dynamic tests to detect and identify an attack

About mobile apps attack surface (Android)

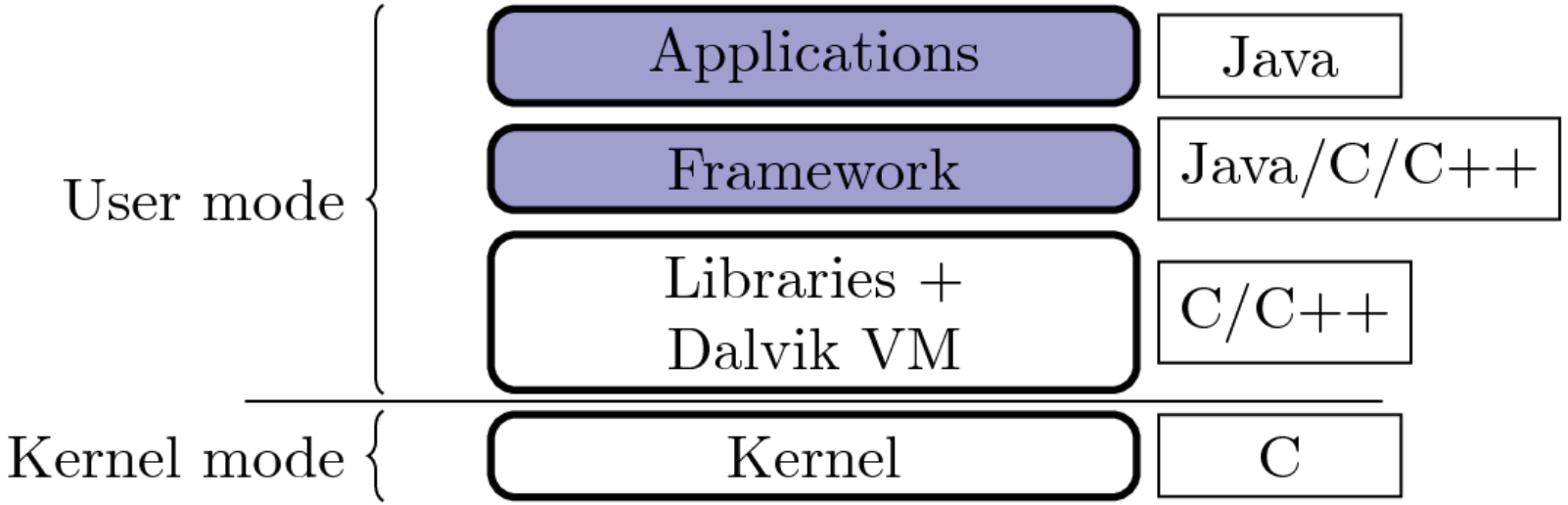
Alexandre Bartel – PhD - UL
Jacques Klein – Researcher - UL

Permission-based architectures



Android Overview (1/2)

- Android = Software Stack



Android – Overview (2/2)

- Android = permission based system
  - Every application has a list of permissions
  - Each permission controls access to a specific resource
  - Android 2.2 declares in total 142 high-level permissions.
  - Developers write the permission list

Application 1
INTERNET
CAMERA
READ_SMS

Example of Permission List

Permission Gap

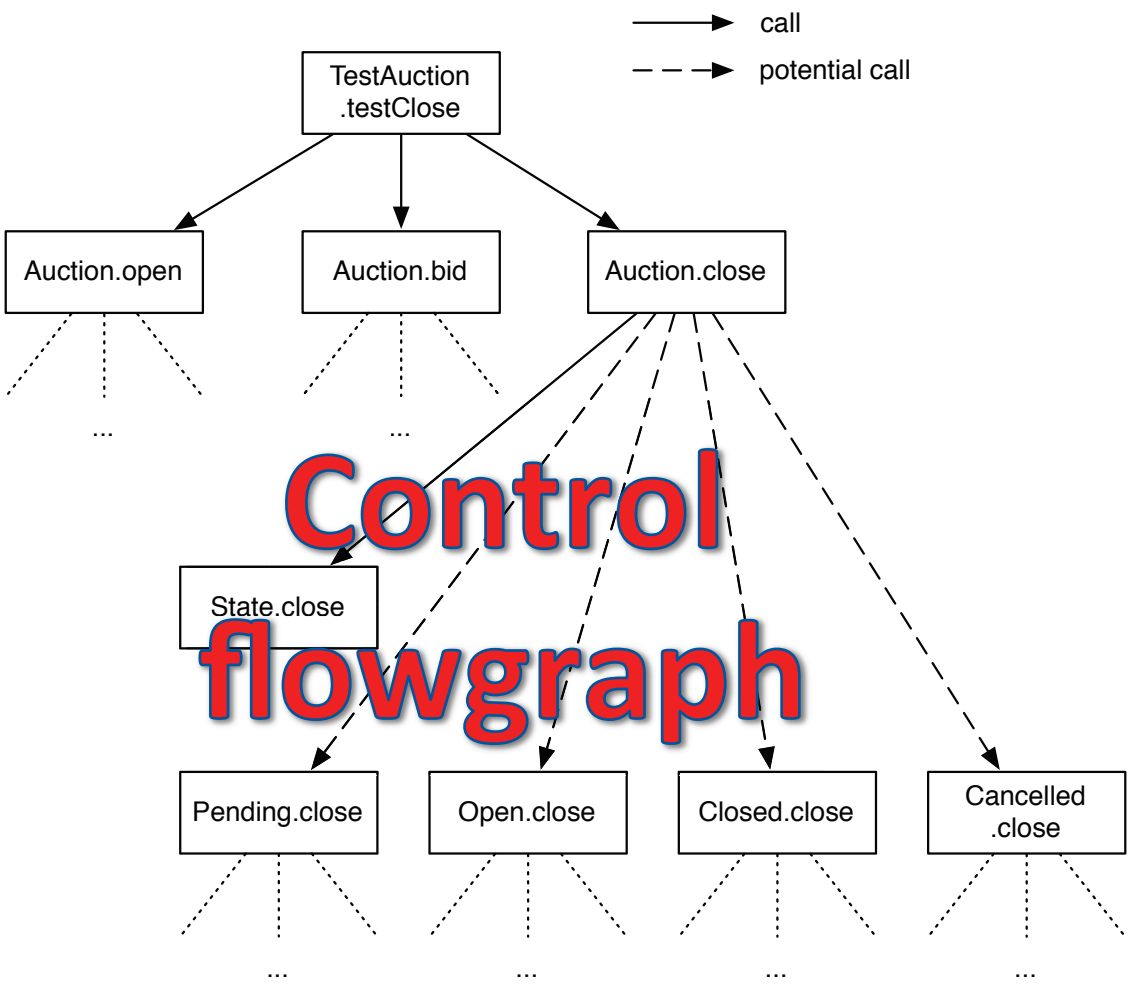
- Permission Gap =

{Declared Permission Set} - {Inferred Permission Set}

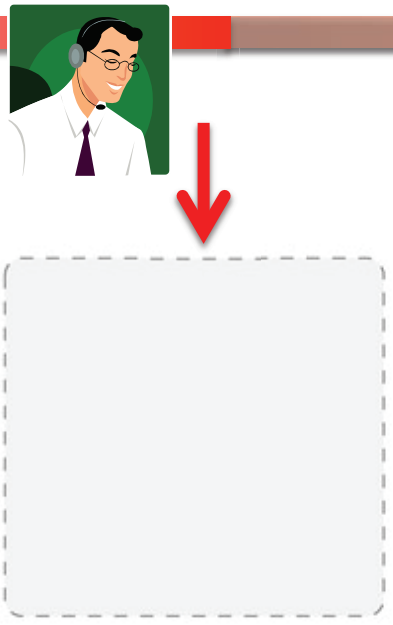
- Consequence: the attack surface is larger
- Example:
  - attacker exploits a buffer overflow in a C library -> he could take advantage of the permission which is declared but not used by the application.

-> How often do applications present a permission gap?

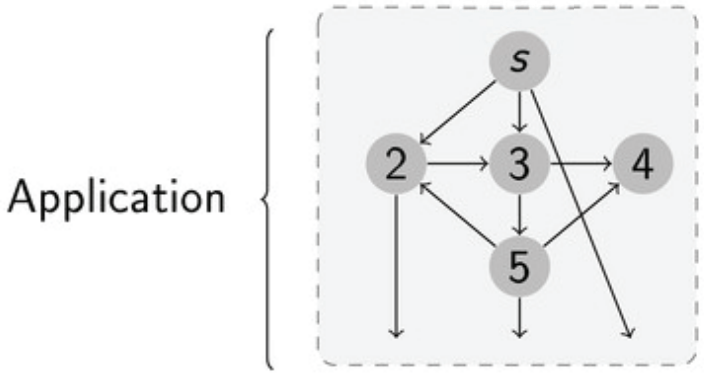
Code Static analysis: control flowgraph



Permission-Based Security Model

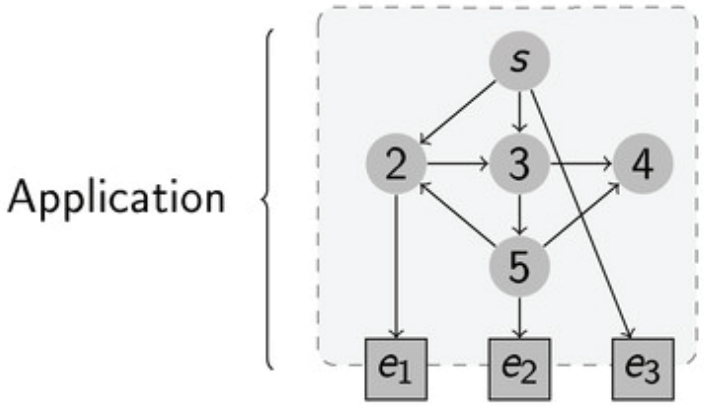


Permission-Based Security Model



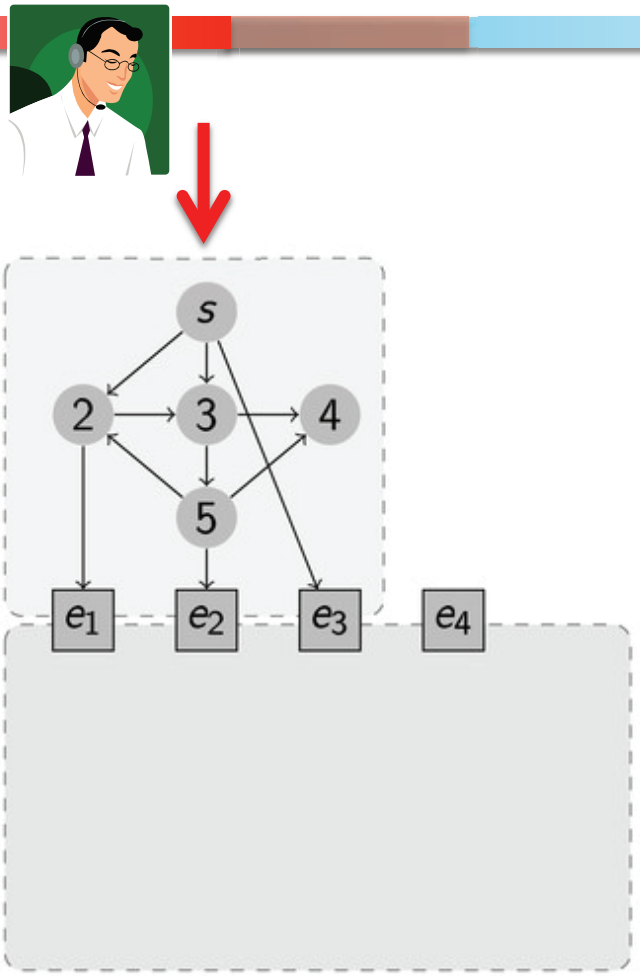
Application
Call graph

Permission-Based Security Model



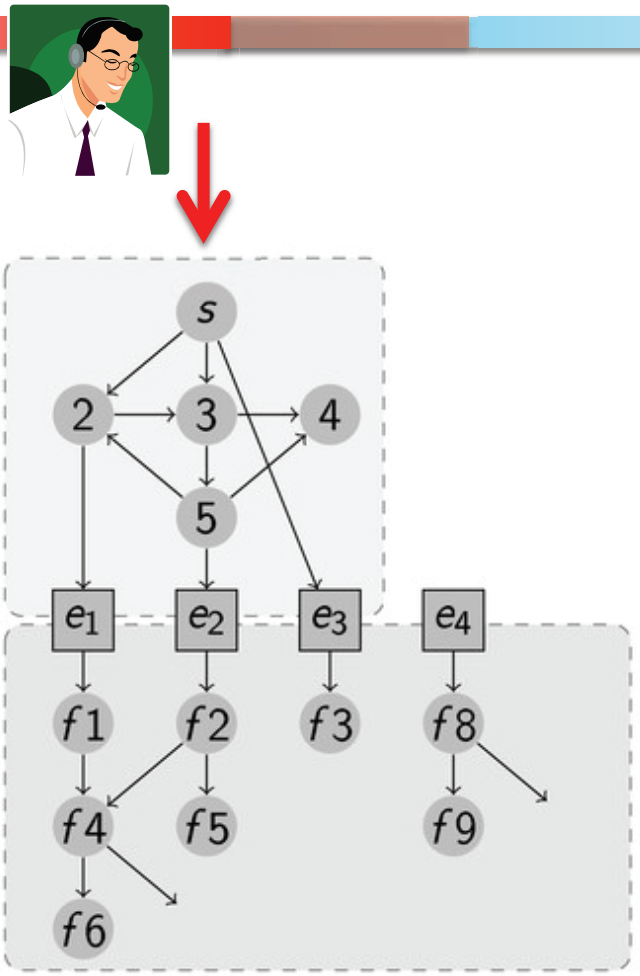
Entry points of the
Framework called
By the Application

Permission-Based Security Model



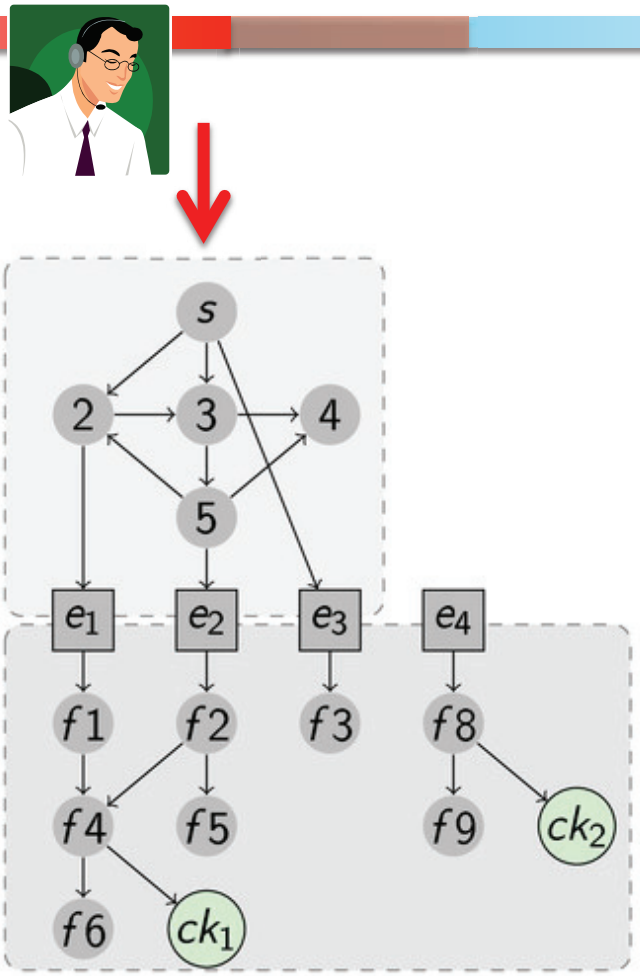
Framework with
All its entry points

Permission-Based Security Model



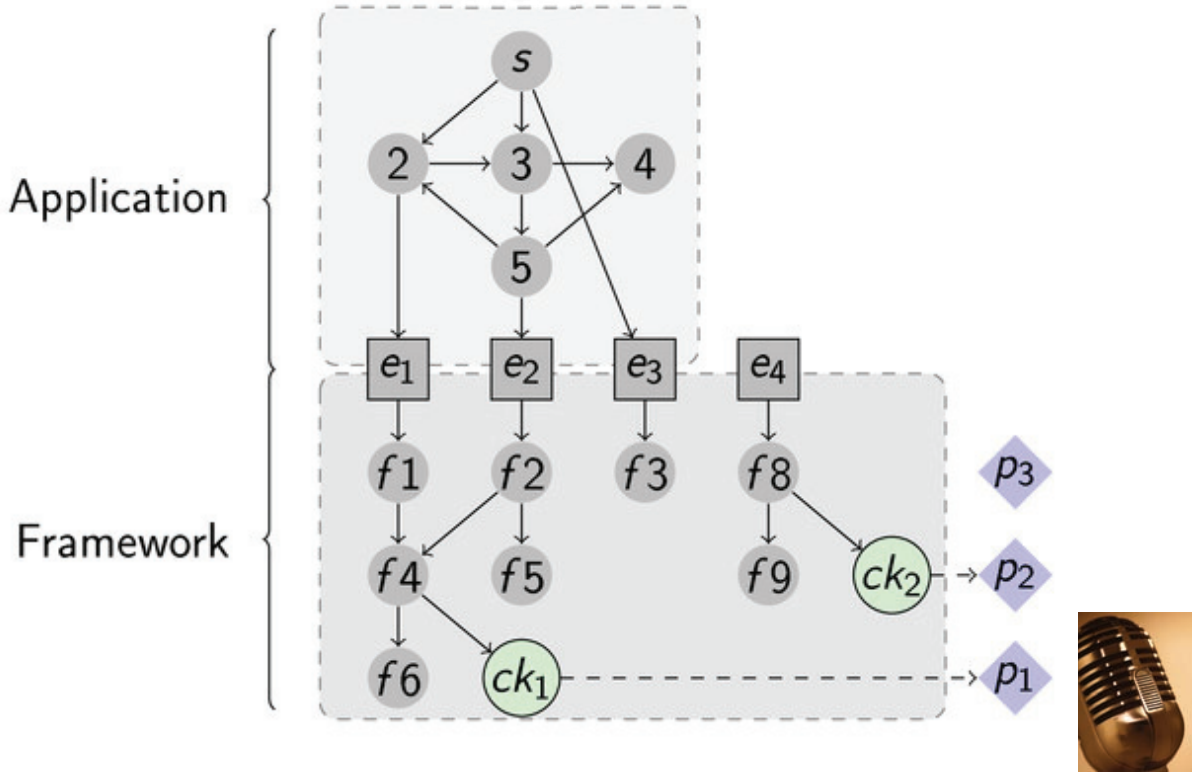
Framework
Entry points'
Call graphs

Permission-Based Security Model



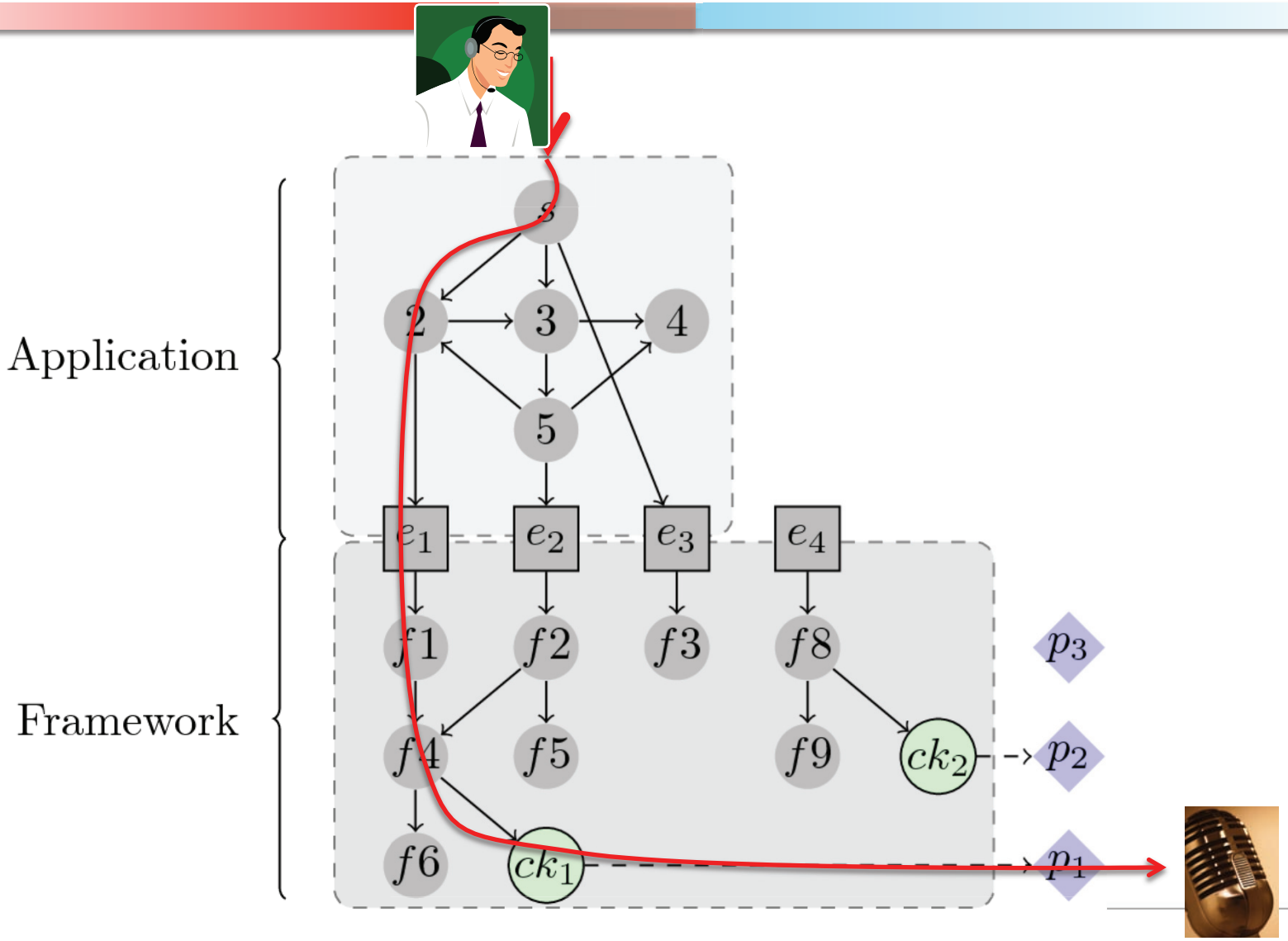
Permission Checks

Permission-Based Security Model



Permissions

Permission-Based Security Model

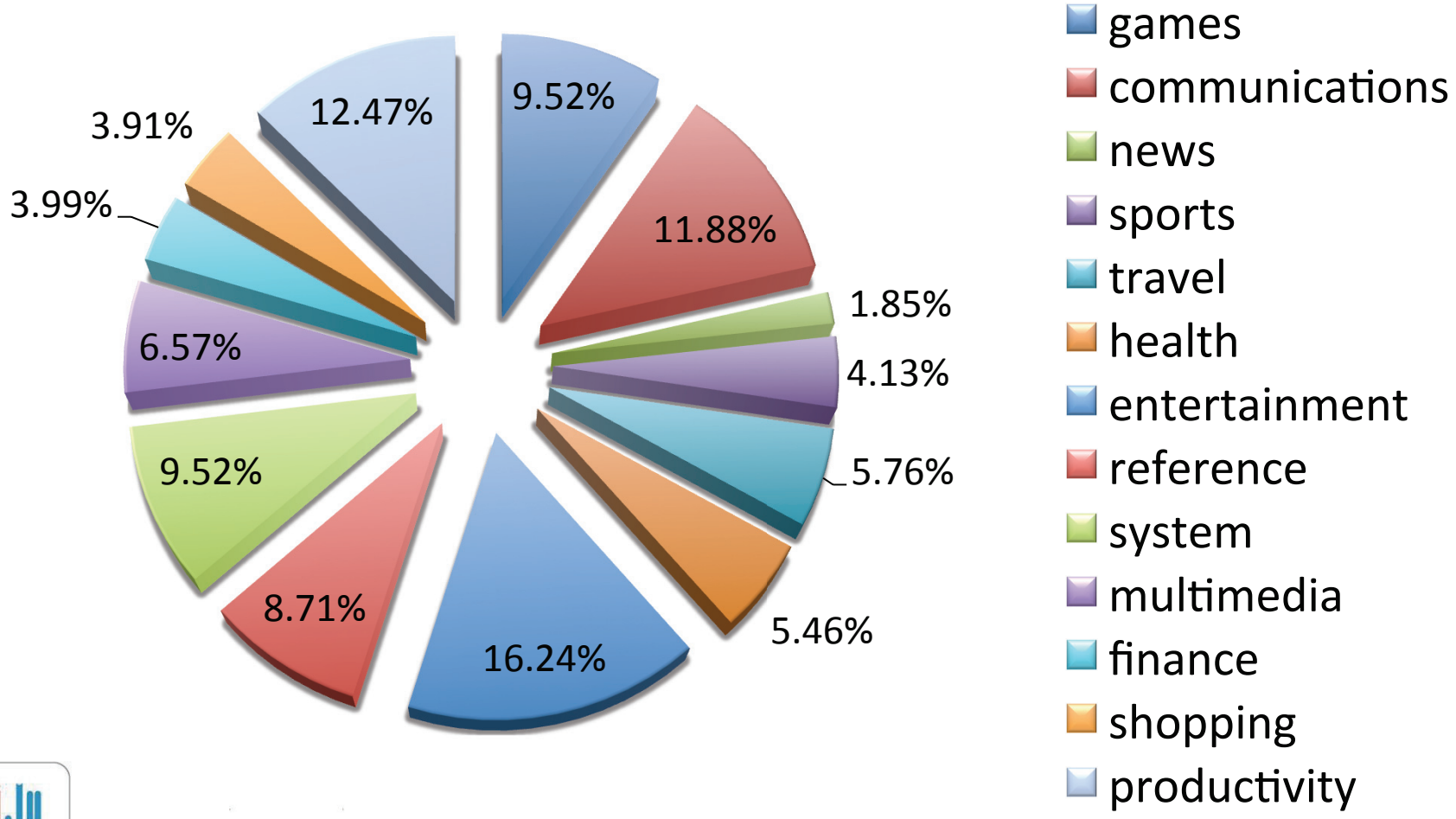


Android framework evaluation

- Android v2.2 bytecode and obtained a matrix M composed of 3957 methods

- We identified 4852 permissions checks in the framework
  - Google maintains more than 4000 enforcement points
  - Not necessarily programmed in a systematic way
  - Documentation and maintenance issues

Case study : 1355 Android applications (Alternative Markets, Nov. 2011)



358 / 1355 have a permission gap (26.4 %)

Permission gap size distribution

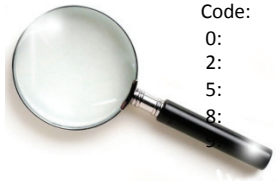
Perm. Gap	#Applications having this perm. gap	% of applications having this perm. gap
0	997	73,58%
1	209	15,42%
2	86	6,35%
3	34	2,51%
4	17	1,25%
5	4	0,30%
6	2	0,15%
7	1	0,07%
8	2	0,15%
9	2	0,15%
10	1	0,07%

Evaluation of the Android Framework Matrix

- Comparison with testing [Felt,2011]
  - At least 3 permissions were missing
  - Combining results from testing (under-approximation) and static analysis (over-approximation) may yield "correct" results
  - 552 methods with same permission list (83%)
  - 119 with one or more permissions

Testing and localizing suspicious byte code

SoundboardActivity



RingtoneManager.setActualDefaultRingtoneUri:
(Landroid/content/Context;ILandroid/net/Uri;)

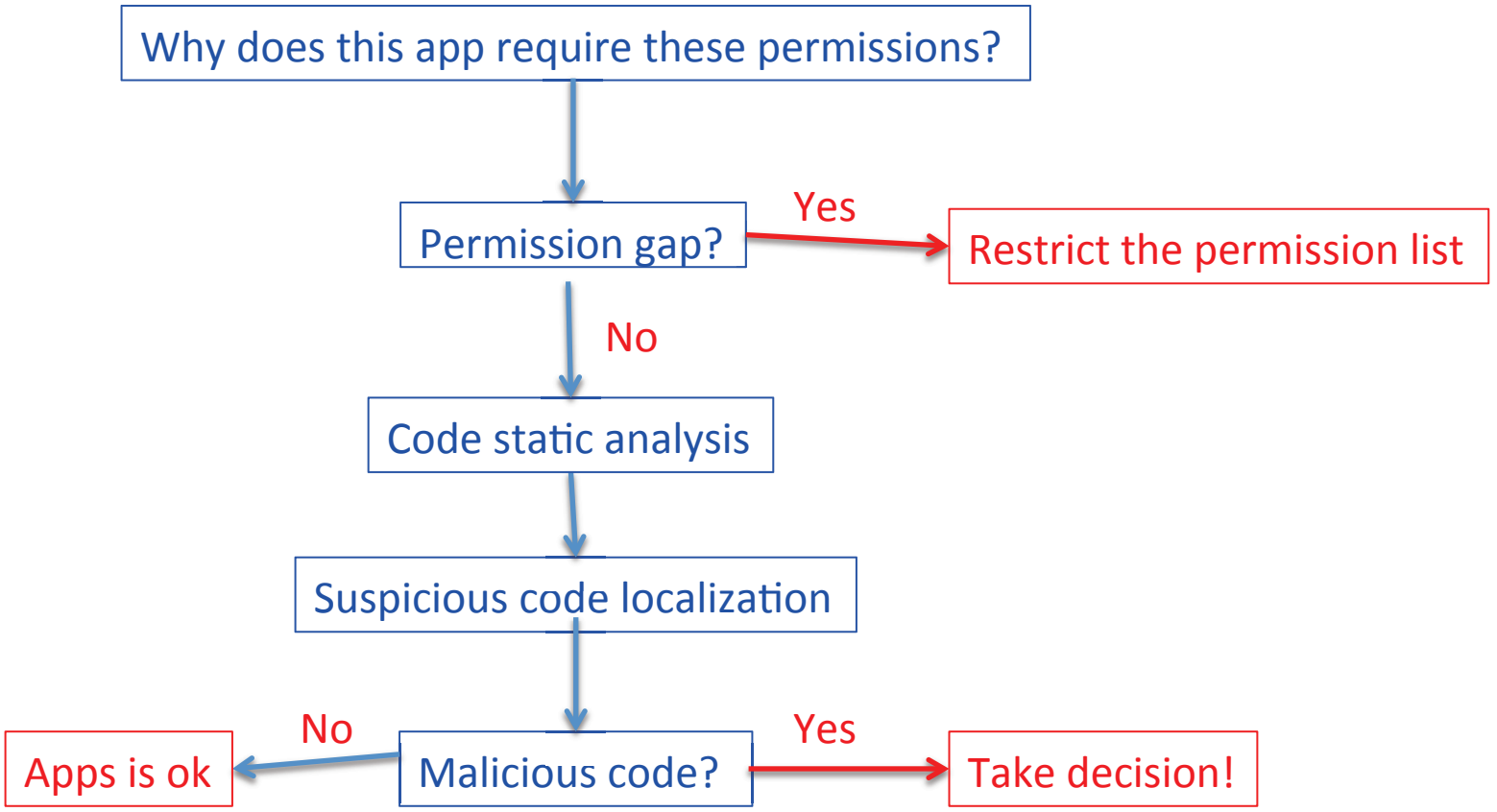
OK !

```
ublic void onActivityResult(int, int, android.content.Intent);
 Code:
    0:          iload_1
    1:          tableswitch{//999 to 999
                999: 24
                default: 20 }
   20:          aload_0
   21:          iload_1
   22:          iload_2
   23:          aload_3
   24:          invokespecial   #378; //Method android/app/Activity.onActivityResult:(IILandroid/content/Intent;)V
   27:          return
   28:          iload_2
   29:          bipush          -1
   31:          if_icmpne       20
   34:          aload_3
   35:          ldc_w           #380; //String android.intent.extra.ringtone.PICKED_URI
   38:          invokevirtual   #384; //Method android/content/Intent.getParcelableExtra:(Ljava/lang/String;)
Landroid/os/Parcelable;
   41:          checkcast       #386; //class android/net/Uri
   44:          astore          4
   46:          aload           4
   48:          ifnull          20
   51:          aload_0
   52:          iconst_1
   53:          aload           4
   55:          invokestatic    #96; //Method android/media/RingtoneManager.setActualDefaultRingtoneUri:
(Landroid/content/Context;ILandroid/net/Uri;)V
   58:          goto            20

public void onAdClick(com.mobclix.android.sdk.MobclixAdView);
 Code:
    0:          ldc             #43; //String SoundboardActivity
    2:          ldc_w           #393; //String Ad Clicked!
    5:          invokestatic    #393; //Method android/util/Log.v:(Ljava/lang/String;Ljava/lang/String;)I
    8:          istore_2
                return

public void onCreate(android.os.Bundle);
 Code:
    0:          aload_0
    1:          aload_1
    2:          invokespecial   #397; //Method android/app/Activity.onCreate:(Landroid/os/Bundle;)V
    5:          aload_0
    6:          ldc_w           #398; //int 2130903044
    9:          invokevirtual   #401; //Method setContentView:(I)V
   12:          aload_0
   13:          aload_0
   14:          putfield        #67; //Field mContext:Landroid/app/Activity;
   17:          aload_0
   18:          invokestatic    #406; //Method com/mobclix/android/sdk/Mobclix.onCreate:(Landroid/app/Activity;)
V
   21:          aload_0
   22:          ldc_w           #398; //int 2130903044
```

Testing and localizing suspicious byte code



Android Inter component Communication

- Given one Android app :
- We use data-flow analysis to
  - Compute a list of components the app. Communicate with

λHaving this map opens the door to :

-Detect Intents which can be intercepted by other applications

-In general the map can be used to detect all kinds of ICC vulnerabilities

-Detect Application Collusion (apps which share permissions)

Application Collusion

Testing how Android malware detection techniques are validated: the generalization problem

- Empirical Software Engineering is becoming pretty mature
  - Reproducibility
  - Data, assumptions, hypotheses and experimental set-up
  - Explicit research questions
  - Internal/external validity. Threats to validity/limitations
  - Key aspect: Generalization
- What a surprise! In the case of malware detection
  - Security people hide their data
  - Non reproduceable
  - Non diverse and large enough
  - The generalization problem is hidden too!

Examples

Full Ref	Authors	Domain	# Malware	# Good ware	Larger test set size	Note
Sahs, J.; Khan, L., "**A Machine Learning Approach to Android Malware Detection**," Intelligence and Security Informatics Conference (EISIC), 2012 European , pp.141,147, 22-24 Aug. 2012	**J. Sahs & L. Khan**	**Android**	91	0-2100	**None**	Train on 90%, test on 10%
Tian, R.; Batten, L.M.; Versteeg, S.C., "**Function length as a tool for malware classification**," Malicious and Unwanted Software, 2008. MALWARE 2008. 3rd International Conference on , vol., no., pp.69,76, 7-8 Oct. 2008	**Tian & al**	Windows	<200	<3500	**None**	Train on 80%, test on 20%
J. Zico Kolter and Marcus A. Maloof. 2006. **Learning to Detect and Classify Malicious Executables in the Wild**. J. Mach. Learn. Res. 7 (December 2006), 2721-2744.	**J. Z. Kolter & M. A. Maloof**	Windows	476	561	**1651 Malware + 1971 Goodware, evaluated with 10-fold**	Train on 90%, test on 10%
Gil Tahan, Lior Rokach, and Yuval Shahar. 2012. **Mal-ID: Automatic Malware Detection Using Common Segment Analysis and Meta-Features**. J. Mach. Learn. Res. 98888 (June 2012), 949-979.	**Tahan & al**	Windows	849	2627	**None**	Train on 80% or 90% (Not specified, inferred from experimental setup)
	Serval	**Android**	1247	620-3500	**52000 (16% of which (=8500) are malware**	

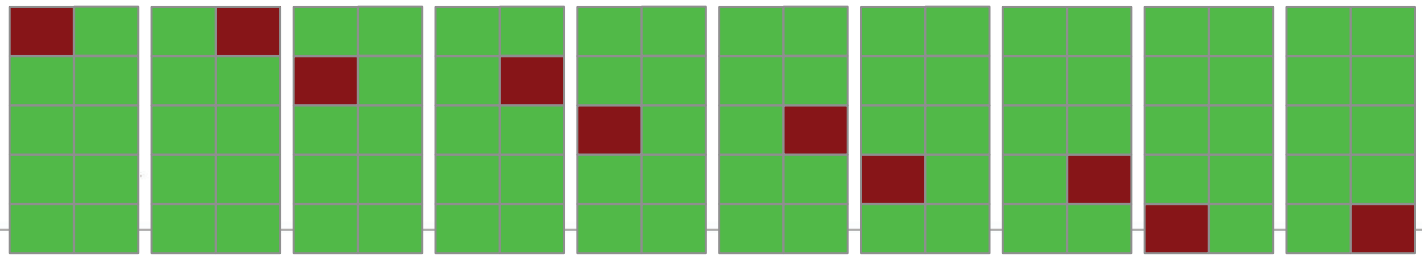
Ten-Folds classification

- Start with a set of “things to classifiy”
  (App1 is a malware; App2 is a goodware; …)
- Divide this set in 10 pieces
- For each piece:
  - Pretend we don't know this piece
  - Build a model based on the 9 other pieces

    (Machine learning magic goes here)
  - Predict the piece we'd like to know

We'd like to Know

We Know



Ten-Folds classification

- Every piece has been predicted exactly once
  - predicting 10% while knowing 90%, 10 times
- Since we actually know them, we can check the prediction *accuracy*
- Standard measures:
  - **Precision**: Percentage of predicted malware that are actually malware
  - **Recall**: Percentage of actual malware that were predicted as malware
  - $\mathbf{F_1}$: $\frac{2 \times precision \times recall}{precision + recall}$

Example

Sahs, J.; Khan, L., "A Machine Learning Approach to Android Malware Detection," Intelligence and Security Informatics Conference (EISIC), 2012 European , pp.141,147, 22-24 Aug. 2012

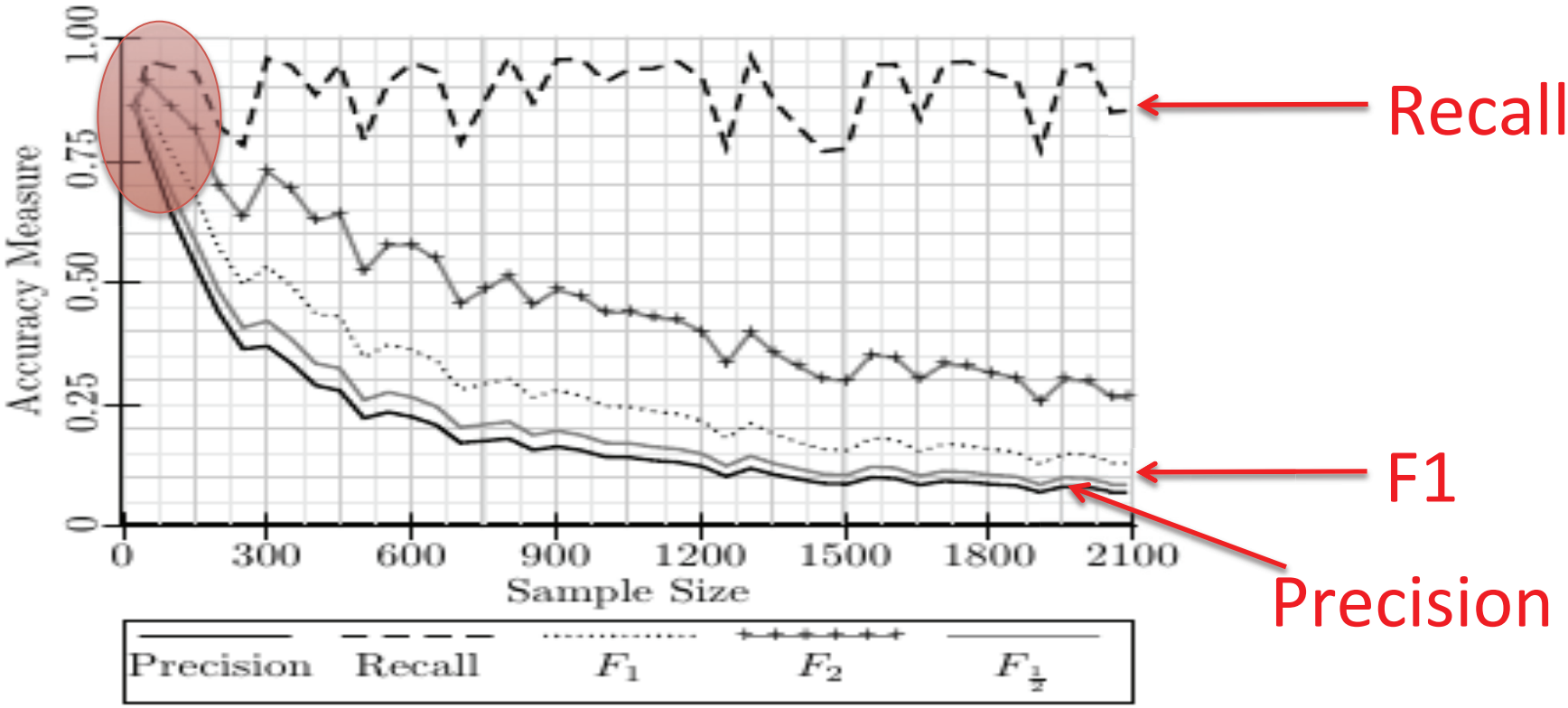
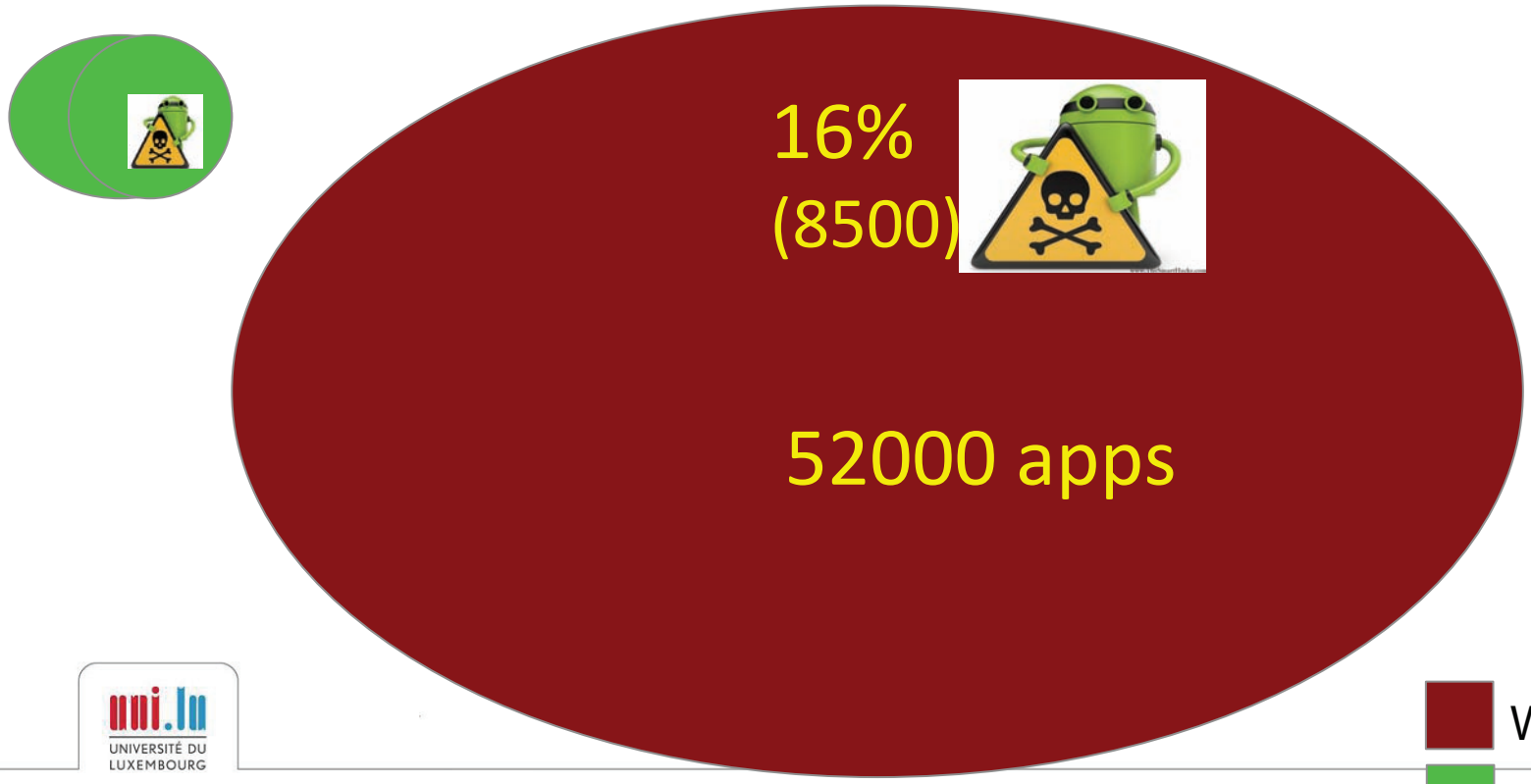


Fig. 3. Accuracy Measures vs Sample Size

Can we generalize?

- If predictions were good for a small unknown set, they should be good for a big one as well, right?



Experimental setup

- Classification techniques
  - SVM (Support Vector Machine)
  - JRIP
  - J48
  - RandomForest
- 960 experiments=

  4 (values for number of Goodware)
  x 6 (values for number of features)
  x 4 (number of algorithms)
  x 10 runs

  30 days of CPU time, 60 millions of predictions
- Analysis the bytecode of:
  - 143 Go from Google Market
  - 1.6 Go for the Malware
- Big server 96 G of RAM, …

Sahs, J.; Khan, L., "A Machine Learning Approach to Android Malware Detection," Intelligence and Security Informatics Conference (EISIC), 2012 European , pp.141,147, 22-24 Aug. 2012

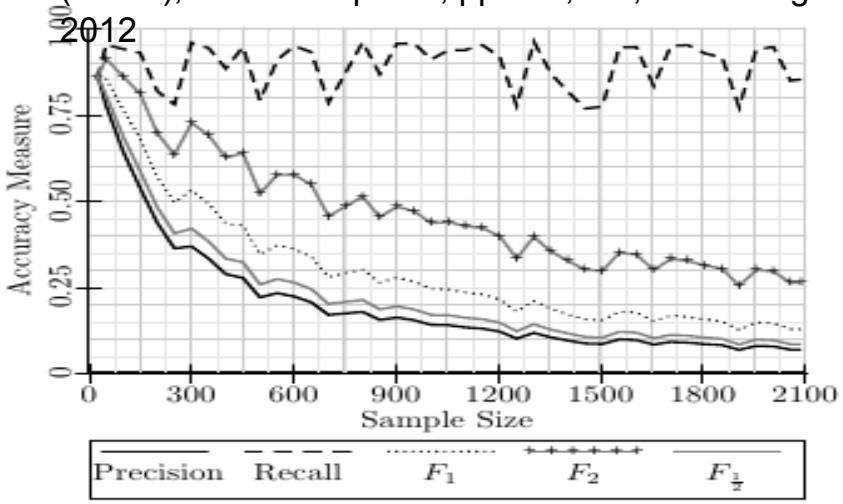
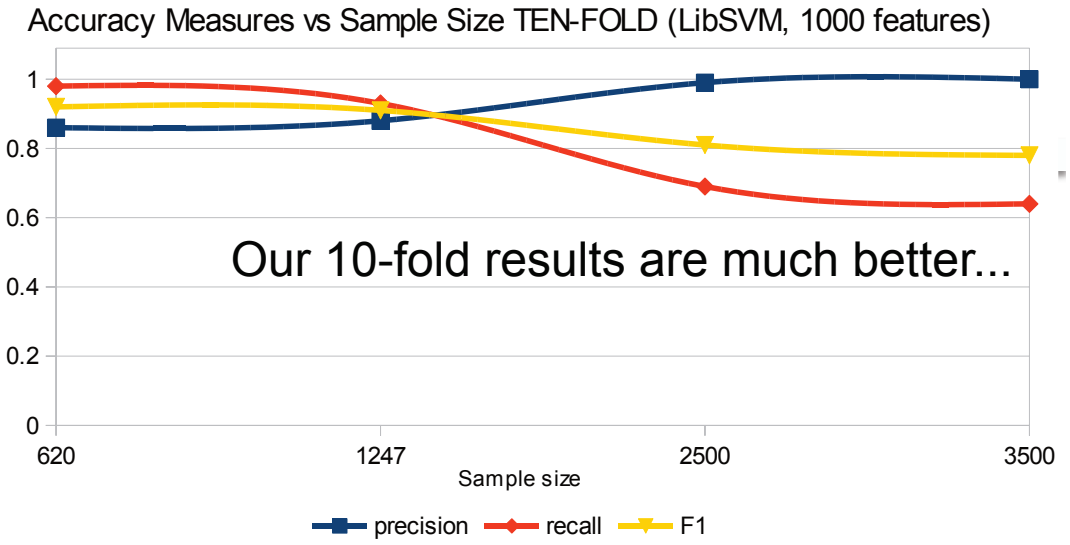
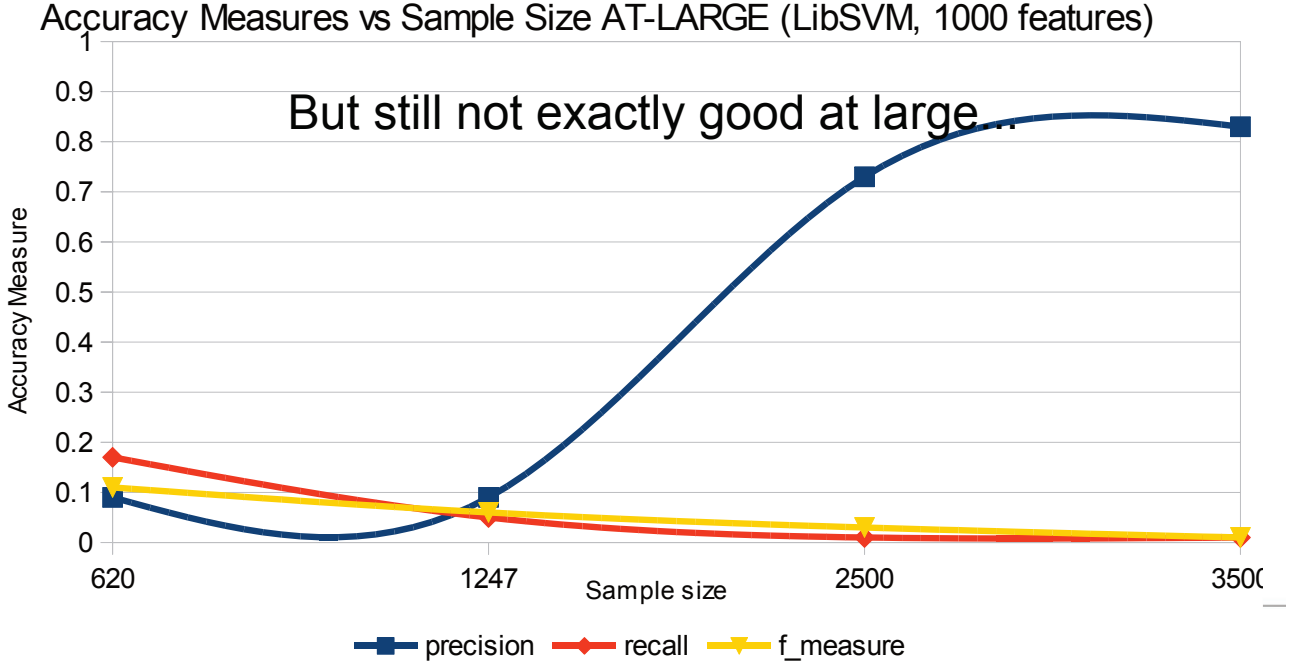


Fig. 3. Accuracy Measures vs Sample Size

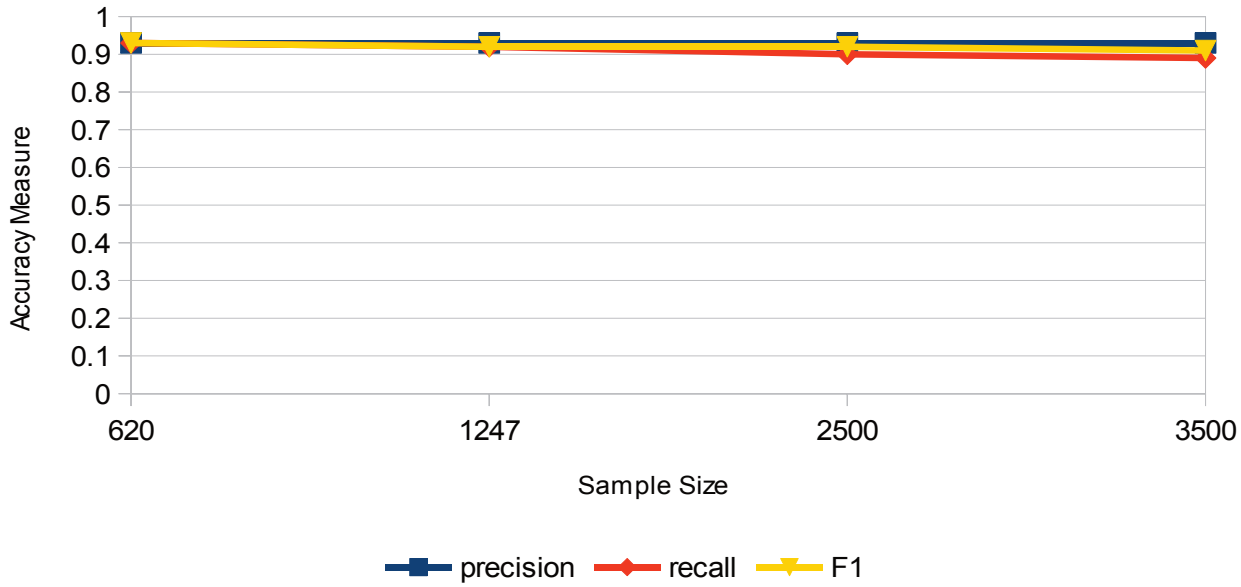
One of our own experiments, very similar



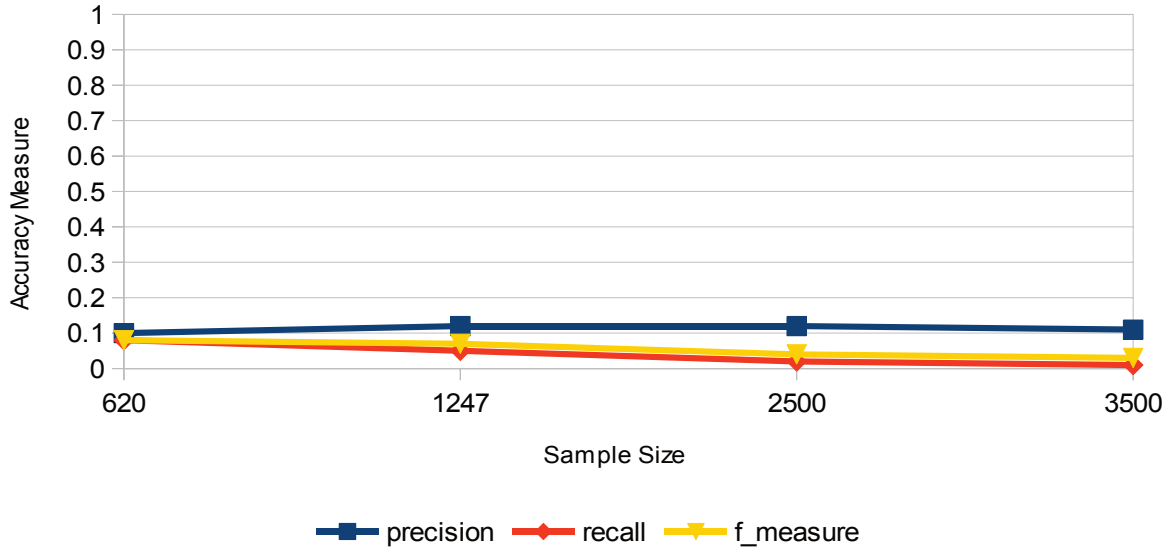
?



Accuracy Measure vs Sample Size TEN-FOLD (JRIP, 1000 features)



Accuracy Measure vs Sample Size AT-LARGE (JRIP, 1000 features)



Ten-fold is bad!

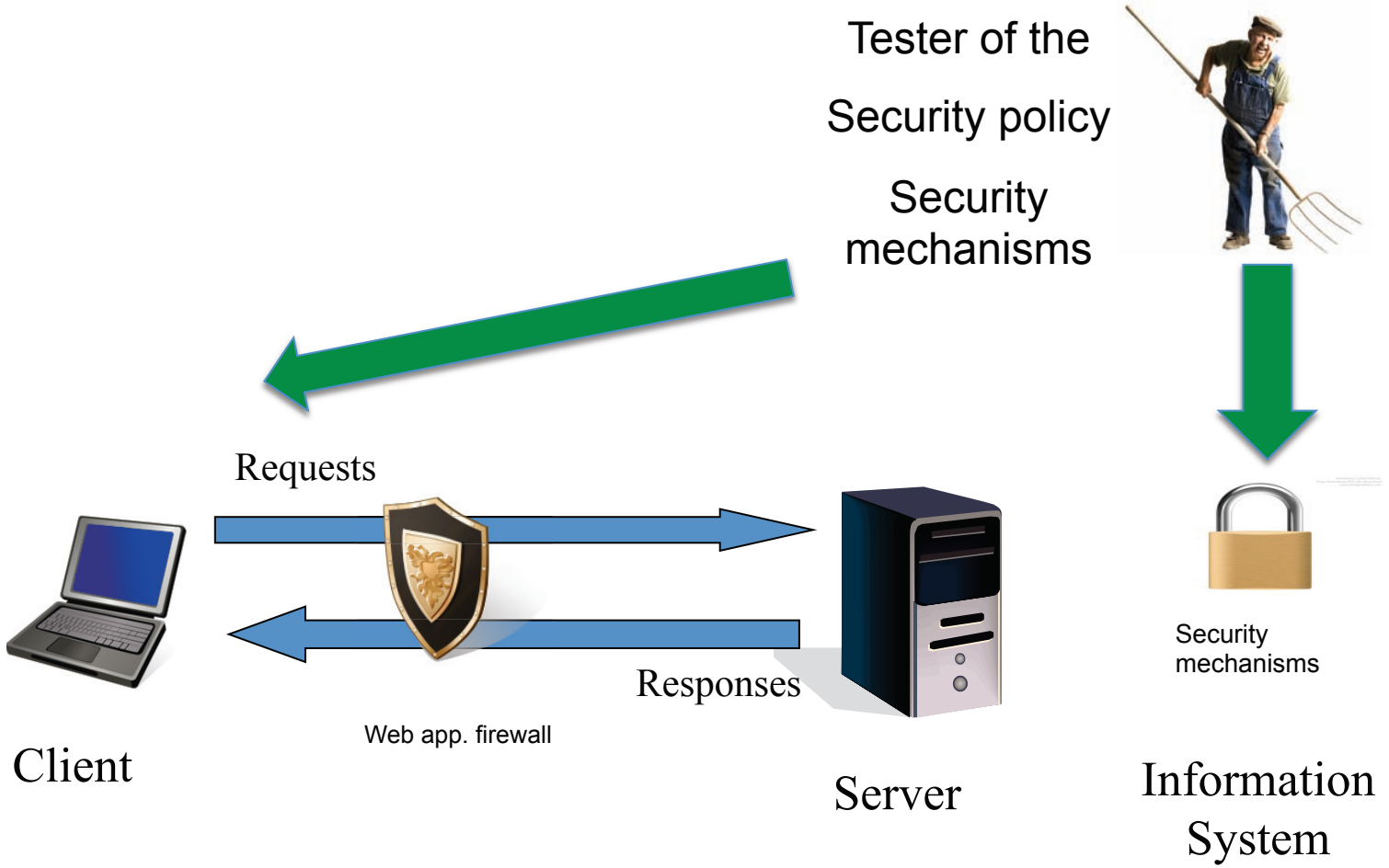
- very good ten-folds metrics (F > 0.9)
- Generalization: 0.23 max, most < 0.1
- Conclusion:
  - Ten-folds metrics provide little insight on how a classifier will perform at large
  - Do not trust security such papers :-)
  - Need to push ESE techniques into the security community

Conclusion Android security and research challenges

- Could we understand how to improve Android apps security?
  - Static analysis allows reducing the attack surface
  - Combining testing and static analysis
- Locate the suspicious byte code?
- Detect malware
  - Combining testing with static analysis
- Having sound and rigorous

About internal information system security

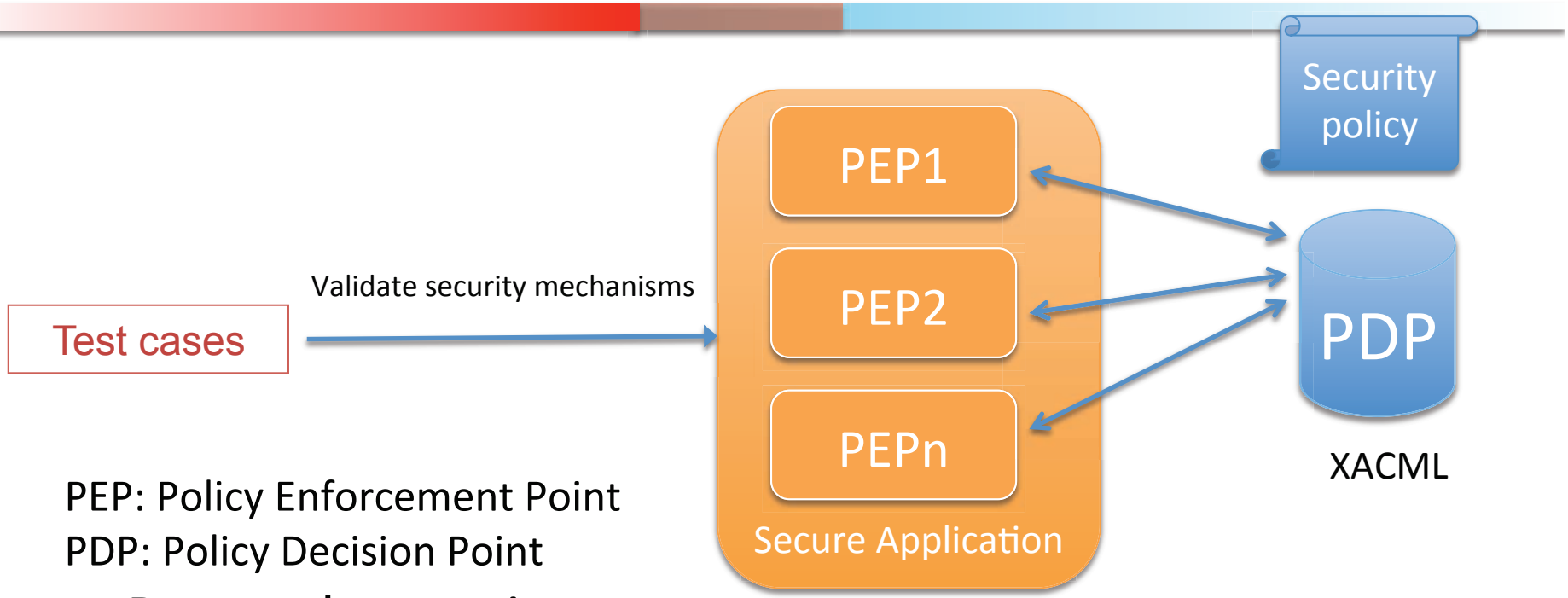
Security testing is two fold



Ideal architecture for a secure system

- Low coupling
  - security mechanisms
  - business objects
- Traceability
  - security requirements <-> security mechanisms
- Separate security components:
  - Control and correction
  - More evolvable
  - BUT : the business logic design must be flexible

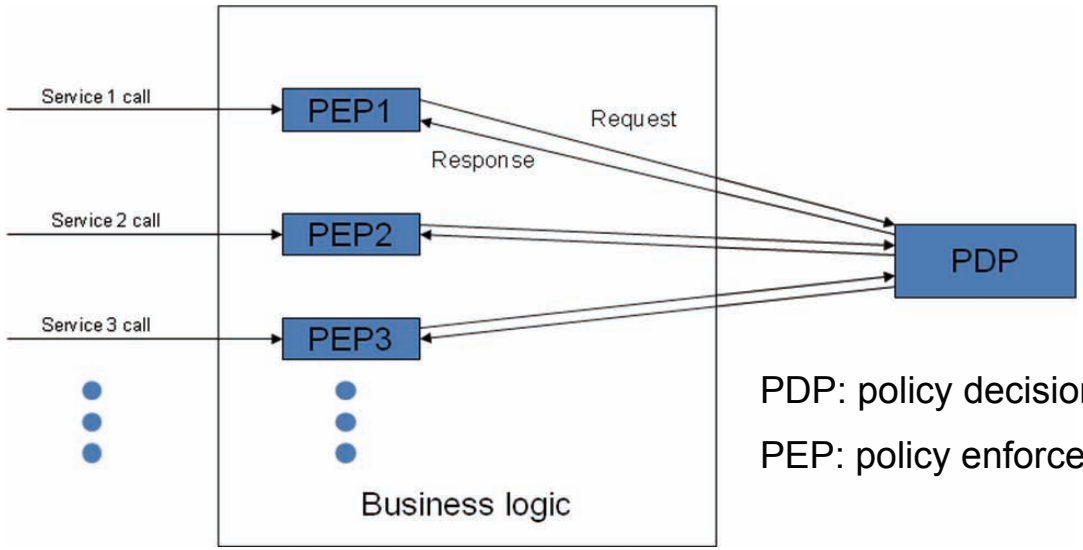
In a nutshell…security policy architecture



PEP: Policy Enforcement Point
PDP: Policy Decision Point

- Research questions:
  - How to evaluate test quality?
  - How to generate security test cases?

Security mechanism architecture



PDP: policy decision point – contains the policy

PEP: policy enforcement point – enforces the policy

Example of PEP

```
Public void borrowBook(Book b, User user) {
// Call to the PDP
SecurityPolicyService.check(user,
SecurityModel.BORROW_METHOD, Book.class, SecurityModel.DEFAULT_CONTEXT);

// borrow book for user
User.borrow(b);
DBUtils.updateBookState(b);
DBUtils.updateBorrowedList(user);
...}
```

XACML Policy

```
<PolicySet PolicySetId="n" PolicyCombiningAlgId="Permit-Overrides">
  <Policy PolicyId="n1" RuleCombinationAlgId="Deny-Overrides">
    <Target/>
    <Rule RuleId="1" Effect="Deny">
      <Target>
        <Subjects><Subject>      Student    </Subject>
                  <Subject>      Secretary </Subject></Subjects>
        <Resources><Resource> Grades    </Resource></Resources>
        <Actions><Action>        Change   </Action></Actions>
      </Target>
    </Rule>
    <Rule RuleId="2" Effect="Permit">
      <Target>
        <Subjects><Subject>      Professor  </Subject>
                  <Subject>      Lecturer   </Subject>
                  <Subject>      Secretary  </Subject></Subjects>
        <Resources><Resource> Grades     </Resource>
                    <Resource>  Records   </Resource></Resources>
        <Actions><Action>        Change    </Action>
                 <Action>        Read      </Action></Actions>
      </Target>
    </Rule>
  </Policy>
  <Policy PolicyId="n2" RuleCombinationAlgId="First-Applicable">
    <Target/>
    <Rule RuleId="3" Effect="Permit">
      <Target>
        <Subjects><Subject>      Student   </Subject></Subjects>
        <Resources><Resource> Records  </Resource></Resources>
        <Actions><Action>        Change   </Action>
                 <Action>        Read     </Action></Actions>
      </Target>
    </Rule>  </Policy>
</PolicySet>
```

R_1: Student or secretary cannot change grades

R2: Professor, Secretary or Lecturer can change or read grades or records

R3: Student can Change or read records

Security policy: "rights and duties"

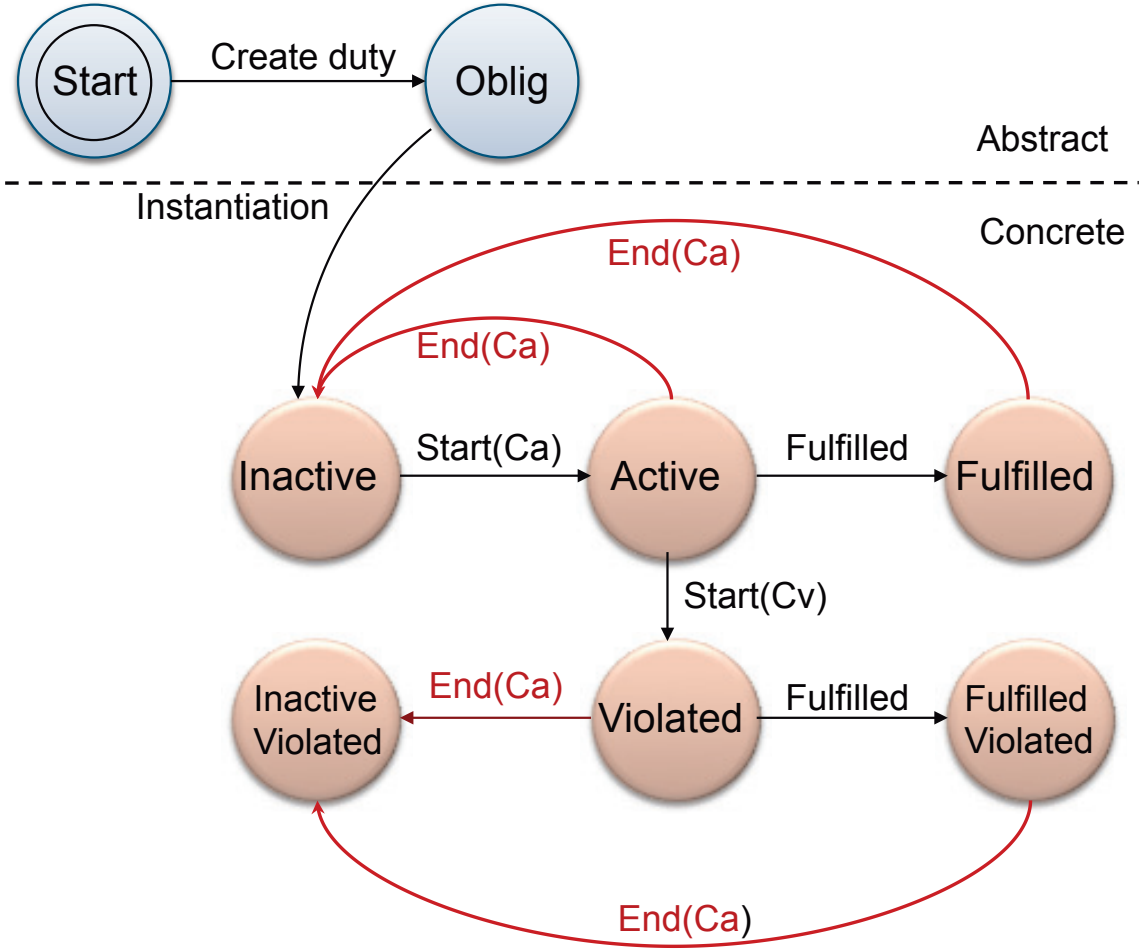
- Access Control
  - Rules
    - Express permissions or prohibitions for users to access some resources of the system
    - Based on an Access control models (RBAC, OrBAC, MAC, DAC, ...)

  "Permission(Library, Teacher, Borrow, Book, WorkingDays)"
- Obligation policies
  - About usages/duties
  - "The doctor should examine a patient within 20 minutes"

Obligation policy/usage control

- Aims at specifying what actions users should do to fulfill security requirements:
  - What user must do
    - Before getting the access
    - During the access
    - After getting the access
- Obligation policy:
  - A set of rules
  - Based on a model

Obligation Management

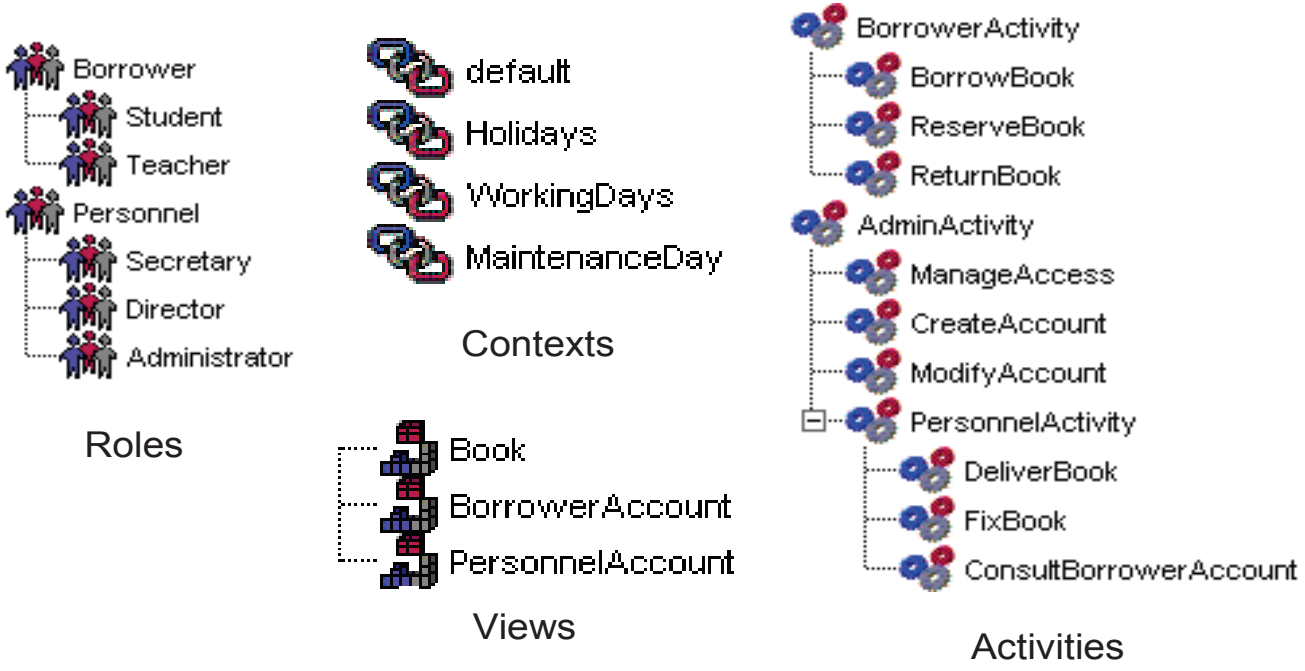


Security requirements: a library management system

- offer services to manage books in a public library
- books can be borrowed and returned on working days. When the library is closed, users can not borrow books. When a book is already borrowed, a user can make a reservation for this book. When the book is available, the user can borrow it.
- managed by an administrator (create, modify and remove accounts for new users).
- A secretary who can order books, add them in the LMS when they are delivered. The secretary can also fix the damaged books (maintenance days)
- The director of the library has the same accesses than the secretary and he can also consult the accounts of the employees
- The administrator and the secretary can consult all accounts of users.
- All users can consult the list of books in the library
- Three types of users: public users who can borrow 5 books for 3 weeks, students who can borrow 10 books for 3 weeks and teachers who can borrow 10 books for 2 months

Security is mixed with functional aspects

The LMS case study



- Examples of rules
  - *Permission(Library,Administrator, ModifyAccount, BorrowerAccount, WorkingDays).*
  - *Permission(Library,Personnel, FixBook, Book, MaintenanceDay)*
  - *Prohibition(Library,Personnel, FixBook, Book, WorkingDay)*

Security policy mutation analysis

Access control Policy:

R1…

R2…

Ri….

Initial policy

Access control Policy:

R1…

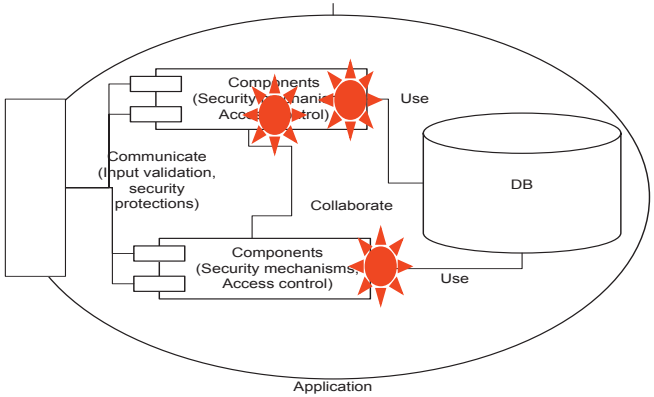
R2…

R3…

Ri….

Mutants implementation

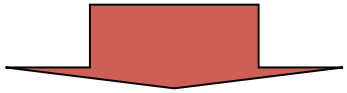
Test cases



Mutation operators

- Simulating access control flaws
  - Permission and prohibition – Rule's type errors (PRP-PPR)

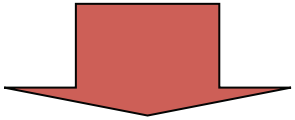
Permission(Library, Administrator, ModifyAccount, BorrowerAccount, WorkingDays)



Prohibition(Library, Administrator, ModifyAccount, BorrowerAccount, WorkingDays

  - Role and context – Parameter errors (RRD-CRD).

Permission(Library, Administrator, ModifyAccount, BorrowerAccount, WorkingDays)

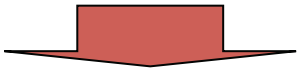


Permission(Library, Teacher, ModifyAccount, BorrowerAccount, WorkingDays)

Mutation operators

– Hierarchy errors on roles and activities (RPD-ADP)

Permission(Library, Student, BorrowerActivity, Book, WorkingDay)



Permission(Library, Student, Reserve, Book, WorkingDay)

– Rule addition for checking tests robustness (ANR)

Prohibition(Library, Personnel, reserve, Book, WorkingDay)

Property : A mutated rule has a higher priority than the other rules

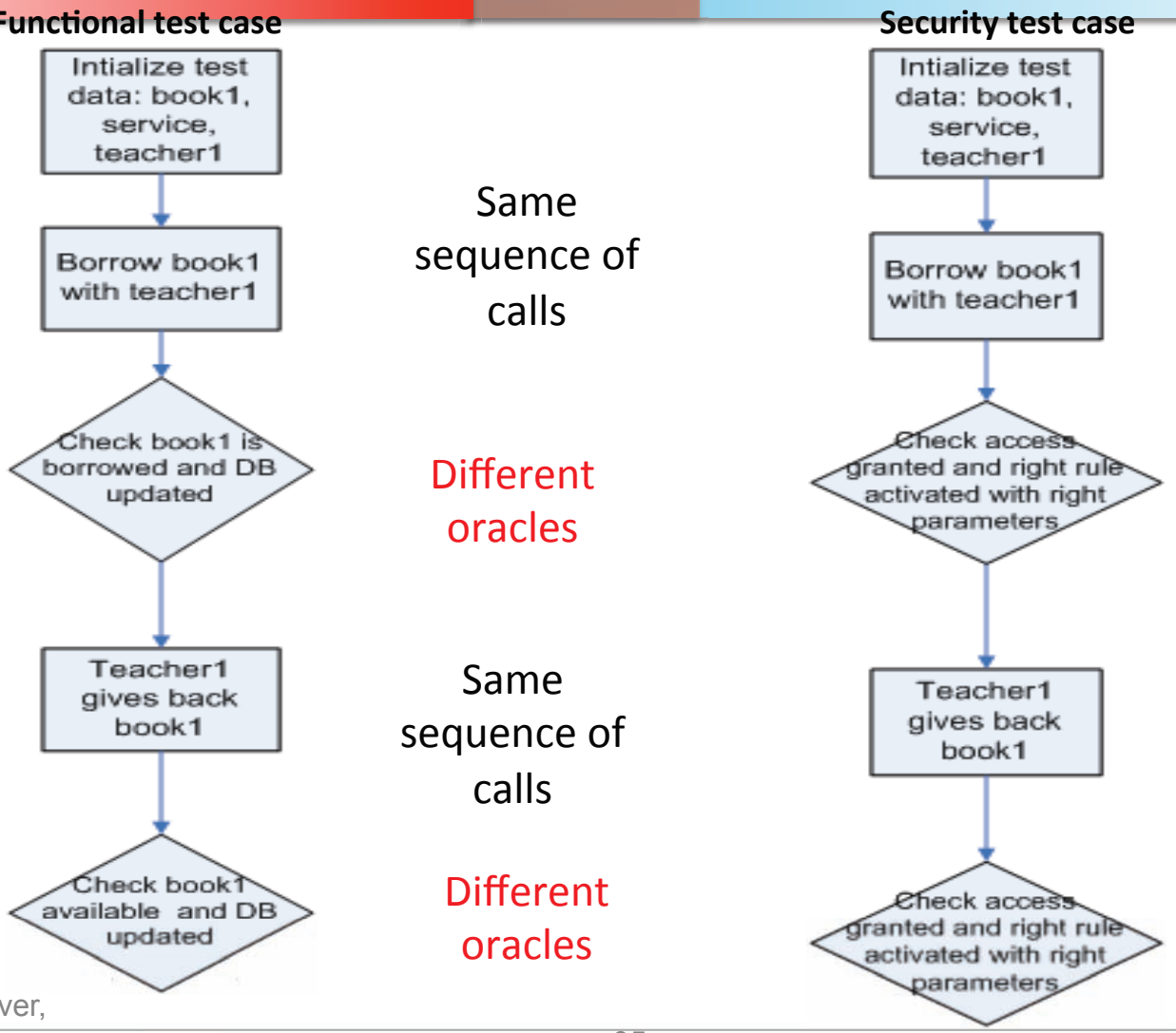
Number of mutants

Operator category		Op.	LMS	ASMS	VMS
Basic Mutation operators	Type changing	PPR	22	89	36
		PRP	19	41	70
	Parameter changing	RRD	60	650	530
		CRD	60	520	318
	Hierarchy changing	RPD	5	20	20
		APD	5	0	20
Rule adding operator		ANR	200	736	432
Total			**371**	**2056**	**1426**

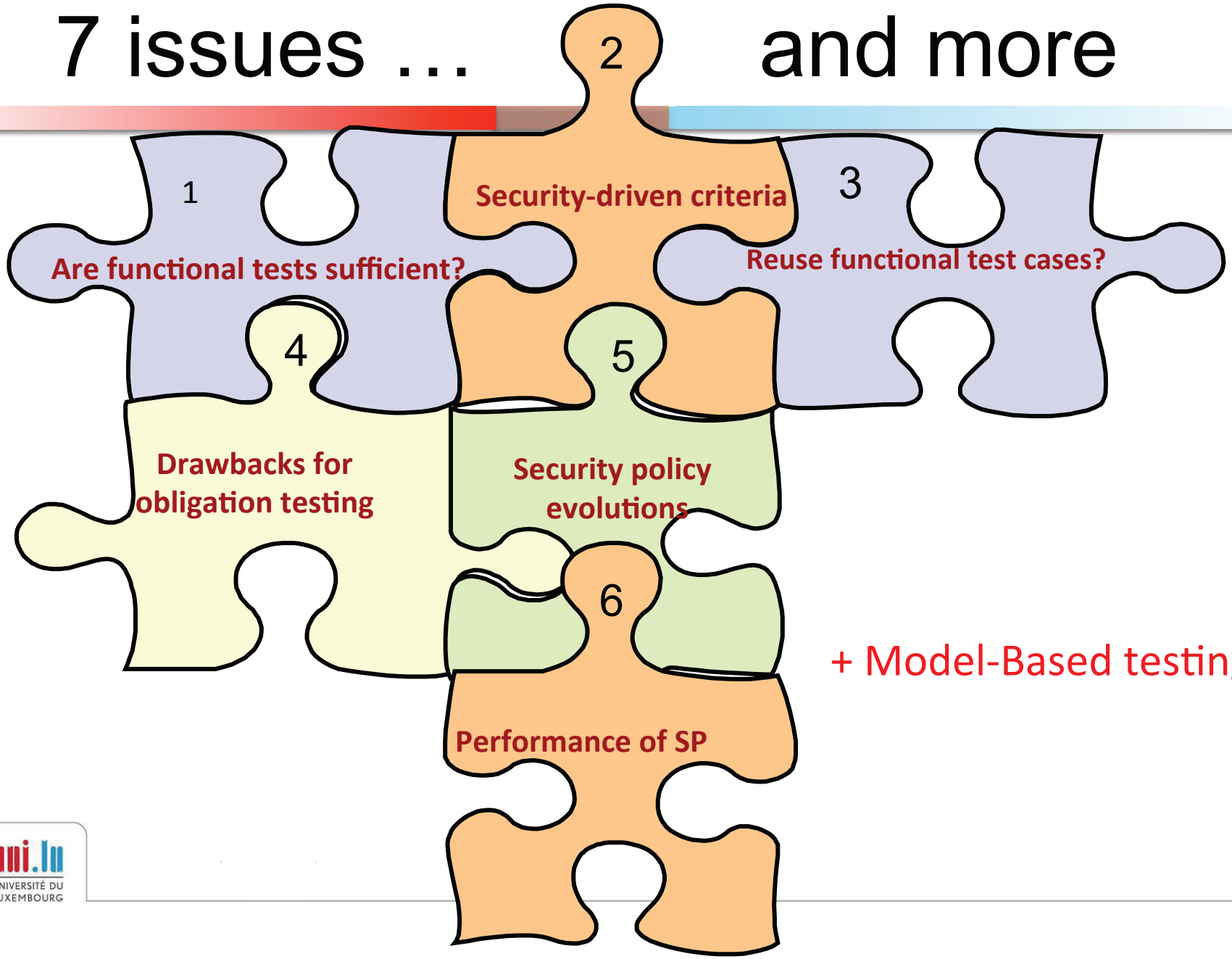
Testing Access Control

- Reuse functional tests
  - Code coverage
- CR1: 1 test case per declared rule
- CR2: 1 test case per concrete rule

Functional vs. security test



7 issues … and more



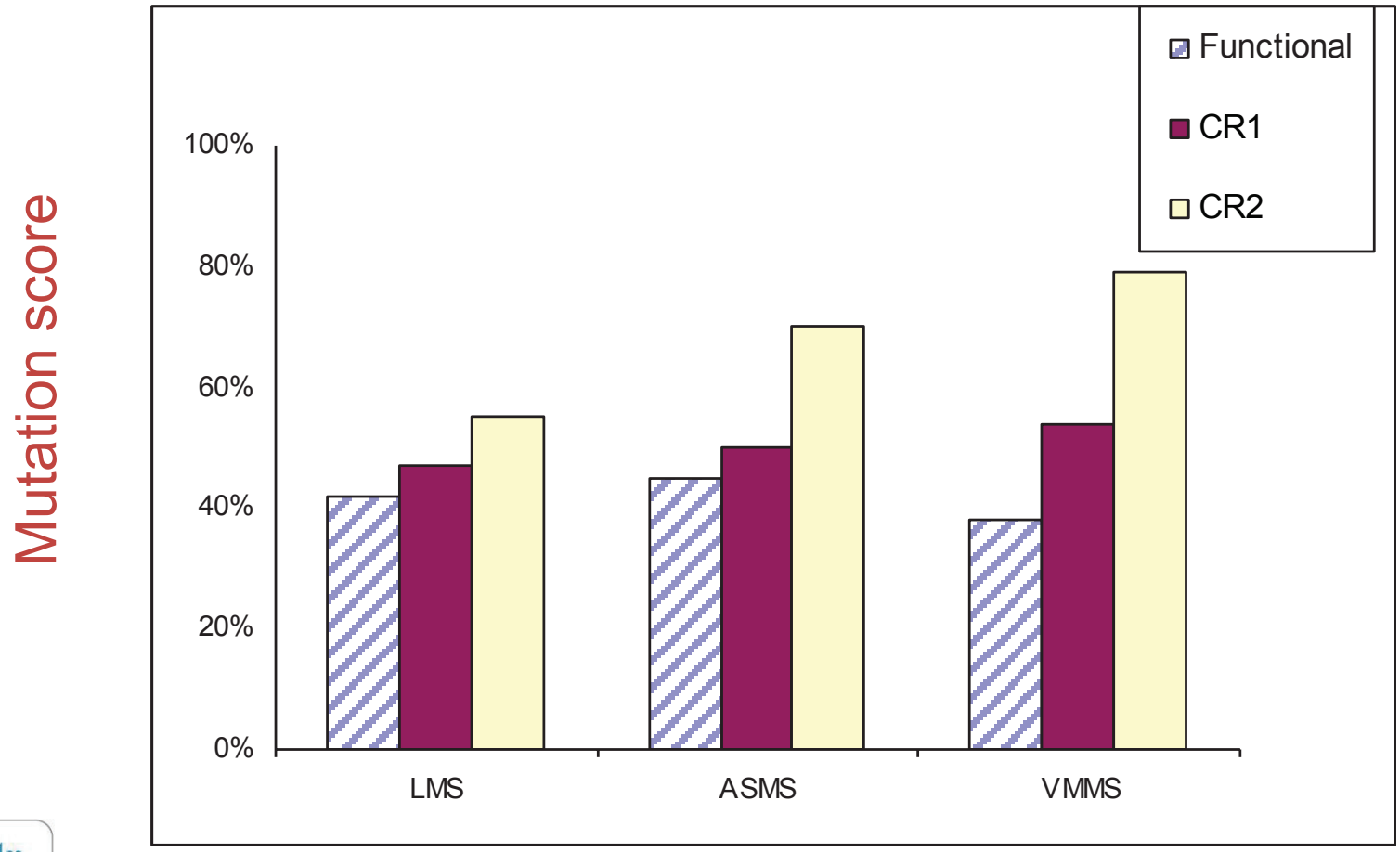
+ Model-Based testing

Issue1 and 2: security testing is a specific target

		#test cases	Basic security mutants	ANR
LMS	CR2	35	100%	**17%**
	Adv. tests	154	**59%**	100%
ASMS	CR2	110	100%	**16%**
	Adv. tests	614	**69%**	100%
VMS	CR2	106	100%	**32%**
	Adv. tests	384	**72%**	100%

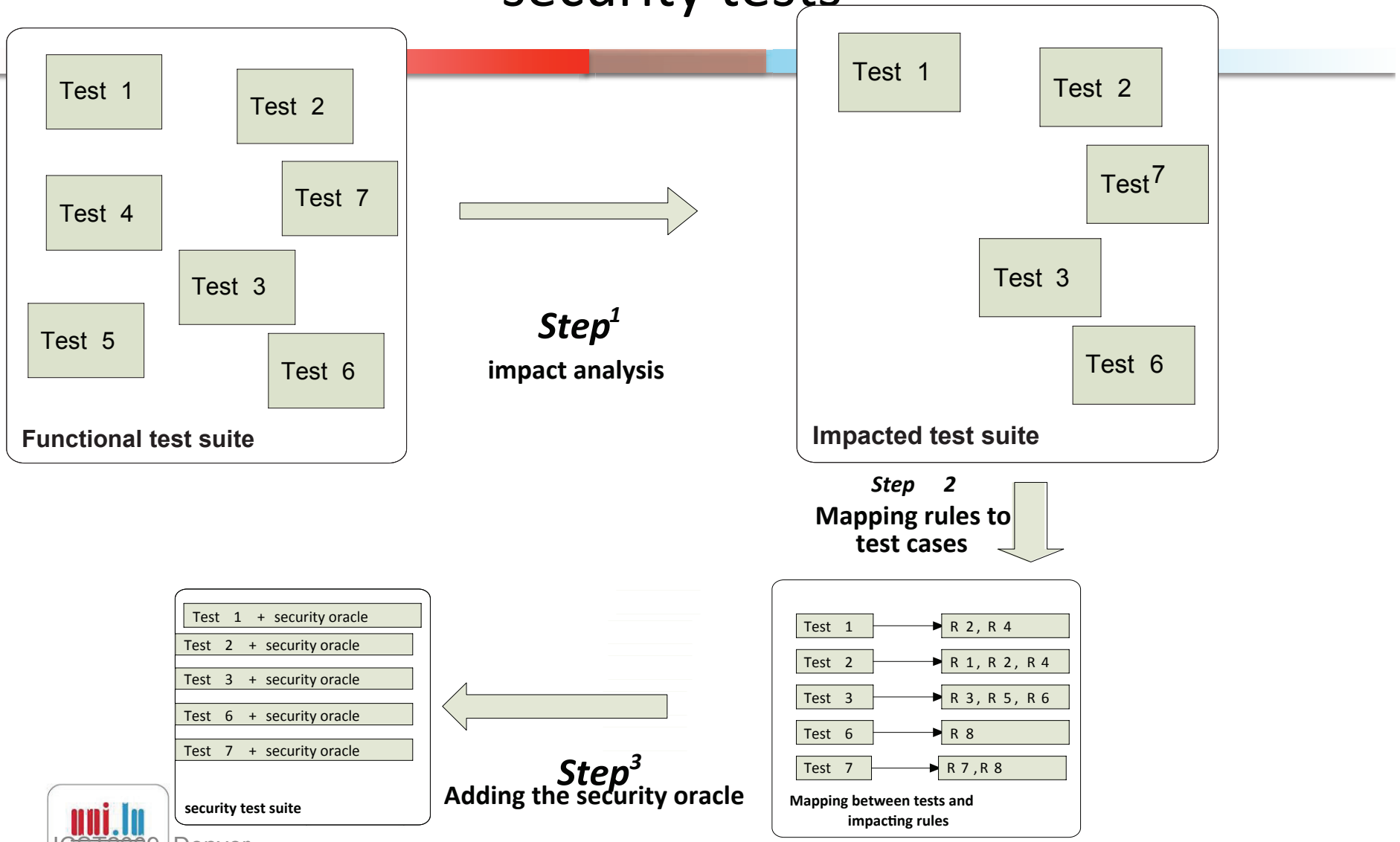
- Advanced security test cases kill all ANR mutants
  - a costly task
- But fail in killing all basic security mutants
- Both criteria are thus needed to be fully efficient in testing the security mechanisms

Issue1 and 2: security testing is a specific target



Mutation scores with all mutants

Issue 3: Reuse tests: Transforming func. Into security tests

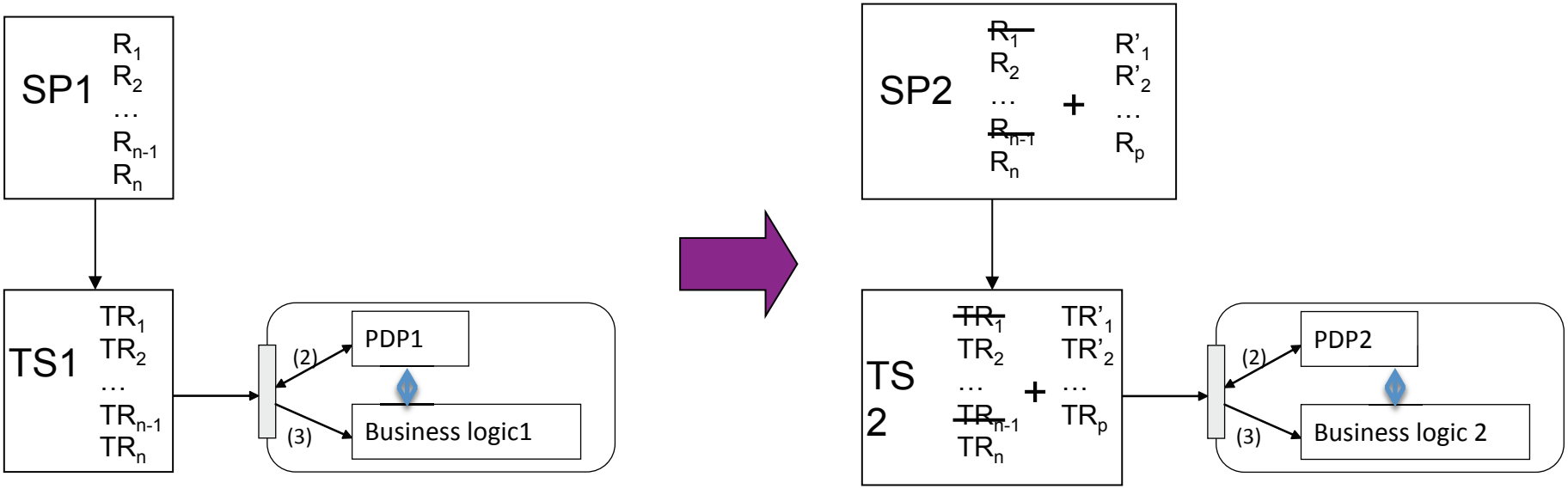


Issue 4: Drawbacks for Obligation testing

Manual implementation of mutation operators

- 54 Mutants were created for this small policy
- 46 were killed
  - All Obligation management, Rule Deletion mutants
- 8 alive mutant:
  - 1 hierarchy enlarging mutant
  - 4 context reduction mutants
  - 3 context extension mutants

Issue 5: Dealing with security policy evolutions (with Alexander Pretschner)



Ideal regression testing scheme

SPi: Security Policy i
TSi : Test suite i
PDPi: Policy Decision Point i
TRi:Test rule i

Issue 5: Explicit/hidden security mechanisms

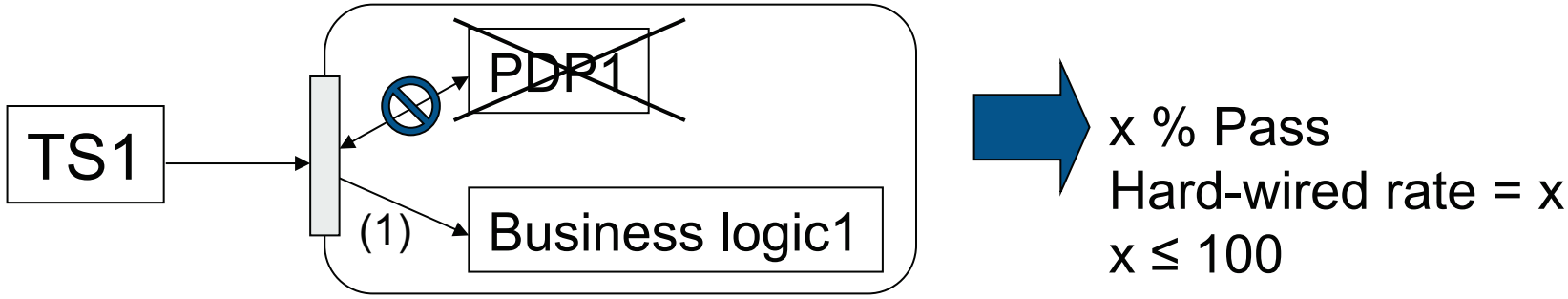
```
public  void borrowBook(User user, Book book) throws SecuritPolicyViolationException {
// call to the security service
ServiceUtils.checkSecurity(user,
LibrarySecurityModel.BORROWBOOK_METHOD,
LibrarySecurityModel.BOOK_VIEW,
ContextManager.getTemporalContext());
// call to business objects
// borrow the book for the user
book.execute(Book.BORROW, user);
// call the dao class to update the DB
bookDAO.insertBorrow(userDTO, bookDTO);}
Public void borrowBook(Book b, User user) {
// visible mechanism, call to the security policy service
SecurityPolicyService.check(user,
SecurityModel.BORROW_METHOD, Book.class, SecurityModel.DEFAULT_CONTEXT);
// do something else
// hidden mechanism
If(getDayOfWeek().equals("Sunday") || getDayOfWeek().equals("Saturday")) {
// this is not authorized throw a business exception
Throw new BusinessException("Not allowed to borrow in week-ends);}   …}
```

explicit

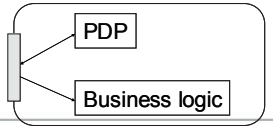
hidden

Issue 5: Test driven audit of the current system

Reapply test cases to check the security policy « hard-coding » rate



SPi: Security Policy i
TSi : Test suite i
PDPi: Policy Decision Point i



: a legacy system

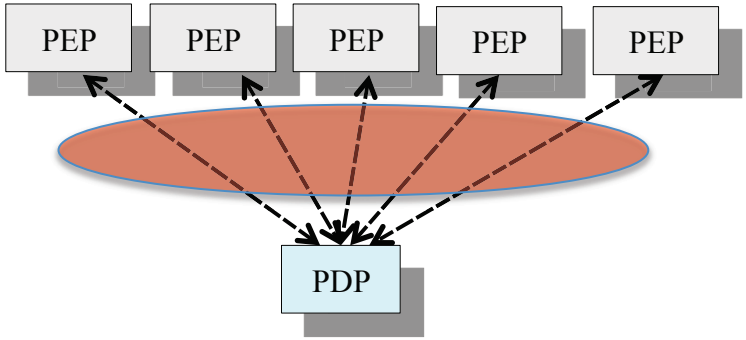
Issue 5: Flexibility of the VMS case study

	Flexible	Rigid rules	System flexibility
results	20	36	0.35

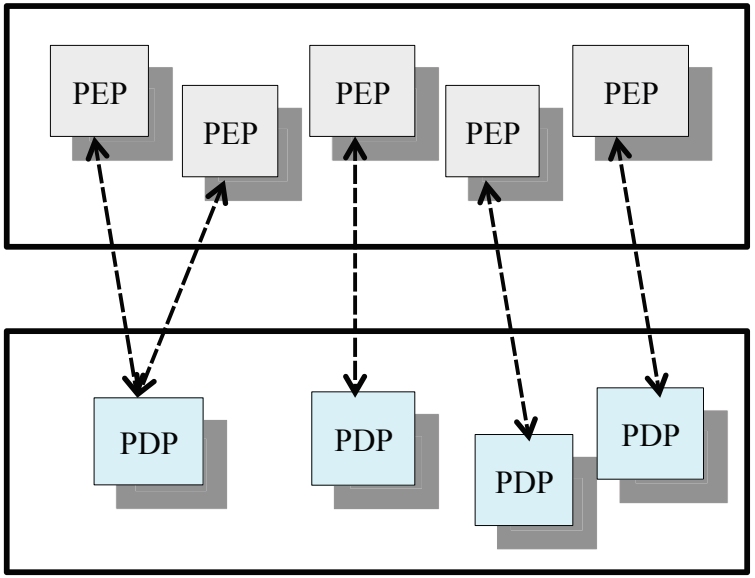
Resource\View	Flex. rules	Hard-wired rules	All	Flexibility
Meeting	12	36	48	0.25
PersonnelAccount	6	0	6	1
UserAccount	2	0	2	1

Function/Activity	Flexibility
updatePersonnelAccount	1
updateUserAccount	1
askToSpeak	0.13
leaveMeeting	0.14
overSpeaking	1
closeMeeting	1
setMeetingAgenda	0.14
setMeetingModerator	0.14
speakInMeeting	0.14
setMeetingTitle	0.14
deleteUserAccount	1
openMeeting	1
handOver	1
deletePersonnelAccount	1

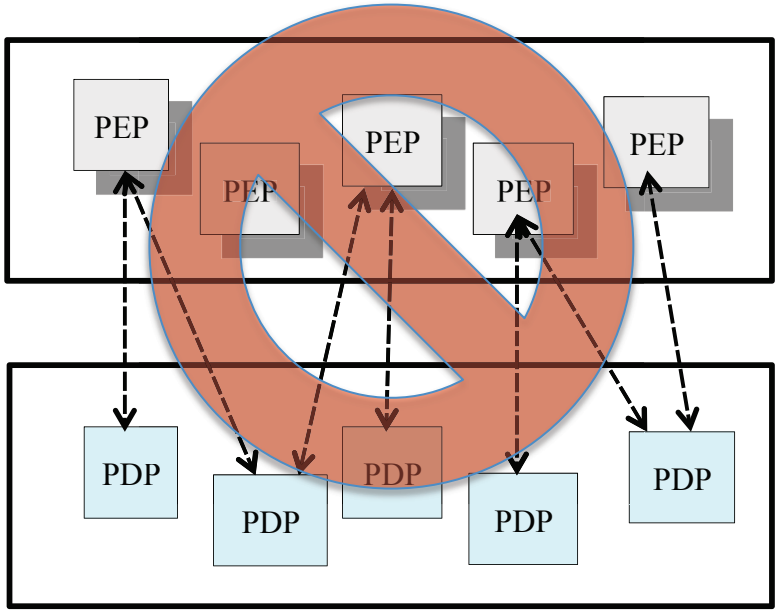
Issue 6. Performance - Synergic vs Non-Synergic Architecture



Initial Architecture



Synergic Architecture



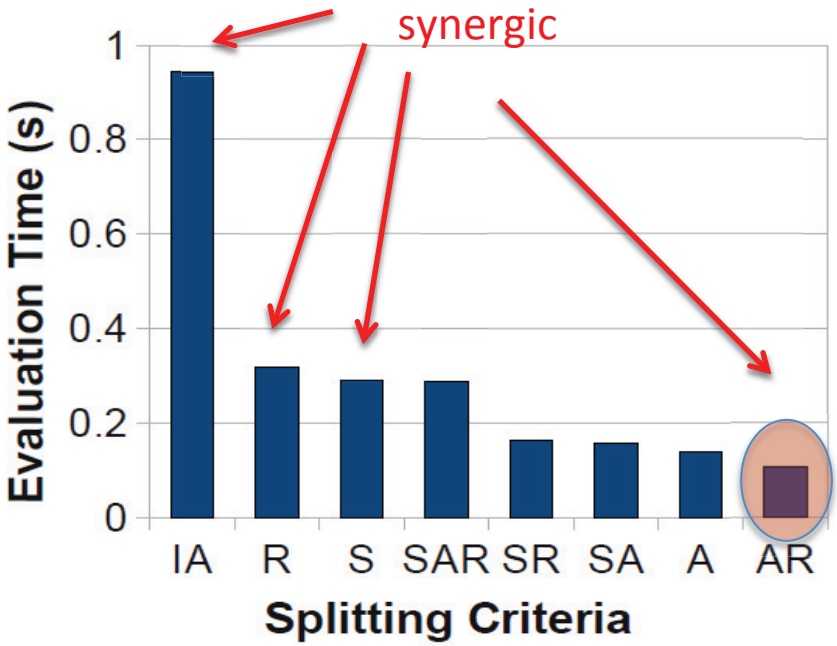
Non-Synergic Architecture

Issue 6: Performances improvements of security policies (with Tao Xie's team)

- A centralized PDP architecture is slow
  - Bottleneck when many requests have to be managed
- Two improvements
  - Split the centralized PDP so that every smaller PDP is associated to a set of PEPs (e.g. a PDP per resource to protect)
  - Since a request to the PDP is treated iteratively, push the most frequent access rules on top of the PDP. Requests are more quickly processed.

Performance results

Performance Improvement: Sun PDP with LMS system



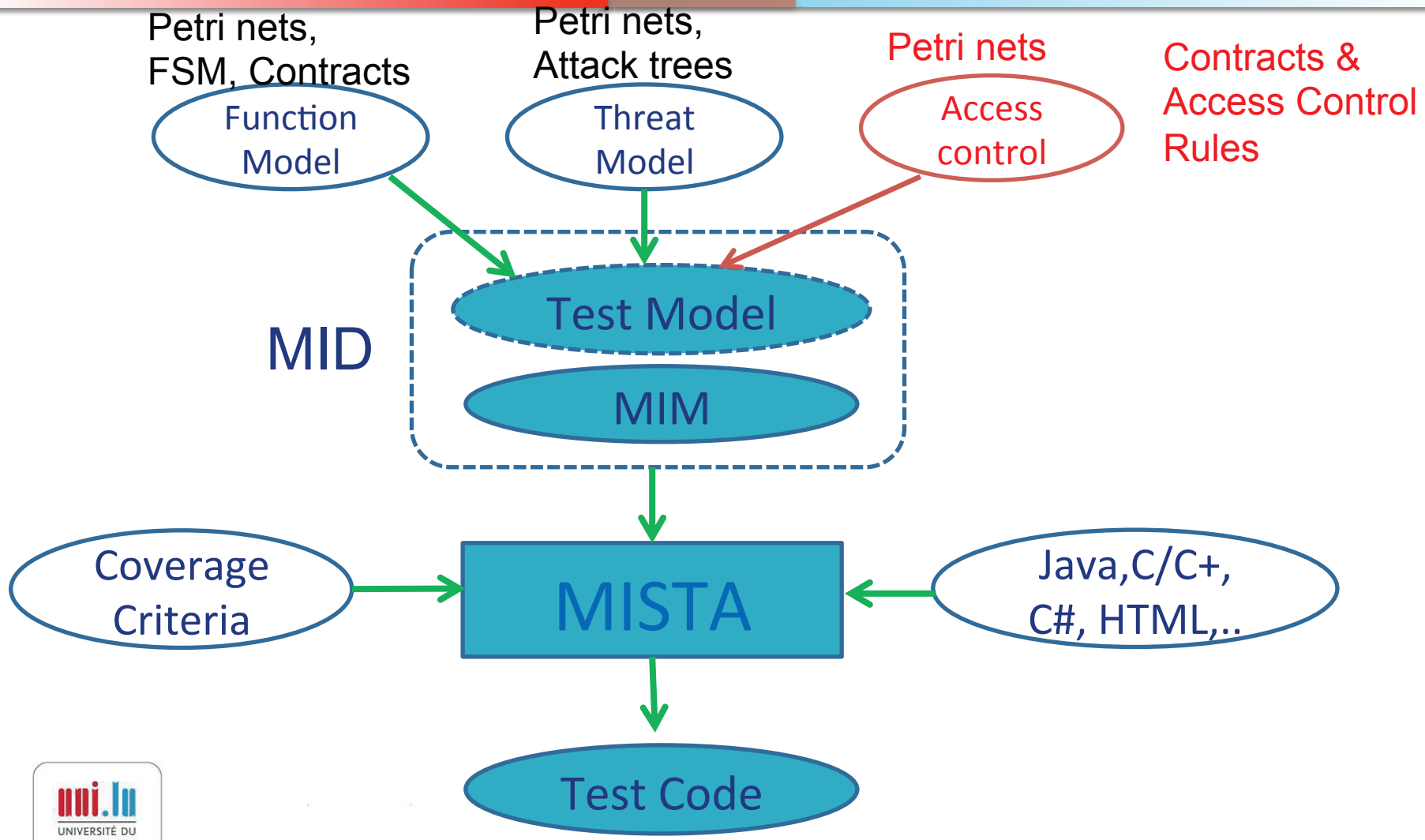
Issue8: A Model-Based Testing for Access Control Policies

Dianxiang Xu, Lijo Thomas, Michael Kent
Dakota State University, USA
Tejeddine Mouelhi, Yves Le Traon
University of Luxembourg, Luxembourg

Model-Based Testing

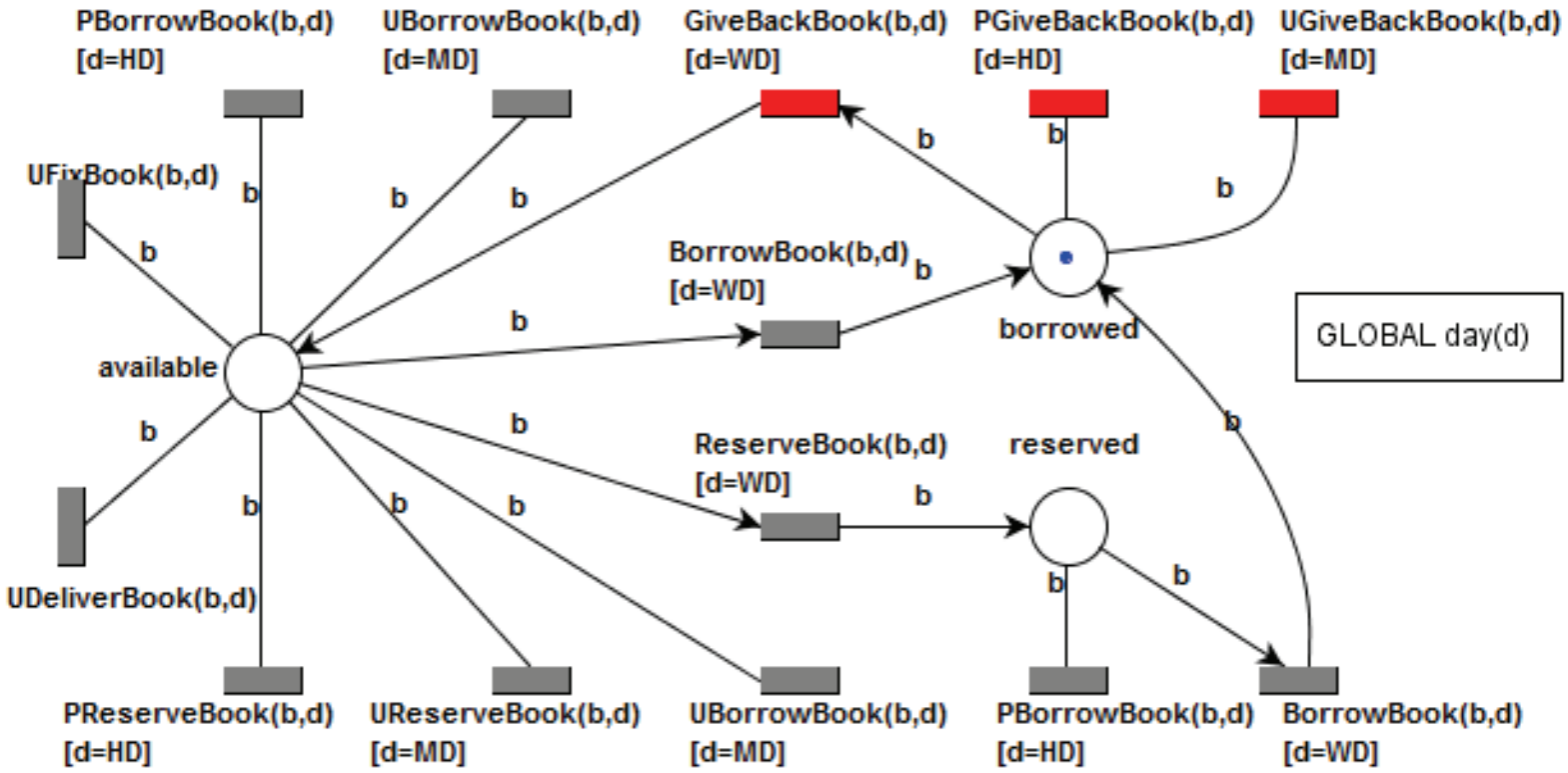
- Uses models of a SUT for generating tests
  - Clarification of requirements
  - Automated test generation and execution
  - Improved fault detection capability
- MBT for Access Control Testing: Issues
  - How to build access control test models in a structured, repeatable process?
    - Declarative vs. operational
  - How to execute abstract model-level tests?
  - How to evaluate testing effectiveness?

MISTA


Petri nets,
FSM, Contracts
Function Model
Petri nets,
Attack trees
Threat Model
Petri nets
Access control
Contracts &
Access Control
Rules
MID
Test Model
MIM
Coverage Criteria
MISTA
Java,C/C+,
C#, HTML,..
Test Code

Constructing Test Models

- Composition of multiple nets
- Test data and initial settings



Empirical Studies: Results

	Models		Tests		Mutation Analysis		
	T	***P***	***TC***	***LOC***	***M***	***K***	***FDR***
LMS	73	27	207	3,086	243	233	95.9%
ASMS	126	30	179	4,680	914	914	100%

- The live mutants of LMS do not violate the required security policies
  - e.g., an added rule allows secretary to return books on any day, but no way to borrow books

Conclusion and research challenges

- A qualification process
- Some challenges
  - Test generation for security policies
    - Formal models
    - MBT
    - Combinatorial testing
  - Test performances vs. security
  - Regression testing when the security policy evolves

Emerging research challenge
Putting the cloud under pressure

Tests must scale too: Peer-to-peer Load Testing / elasticity testing

- Test methodology
- Normal load
- Distributed Denial of Service Attacks (DDoS)

Tests must scale too: Peer-to-peer Load Testing

- Test methodology
- Test language for the cloud / test elasticity vs. Fault tolerance
- Distributed Denial of Service Attacks (DDoS)

Overall conclusion

- Many open challenges
  - Testing "as a hacker"
    - Models for generating new vectors
    - No test adequacy criteria
    - Regression testing
    - Testing IDS and security components
  - Testing a security policy
    - Already some adequacy criteria
    - Formal modelling
    - MBT
  - Testing a cloud robustness
- Design for testable security
  - Model-driven security

« intelligently react to abnormal situations and ensure the quality of the information » (P1 conclusion)

Questions ?