Metrics, Methods and Tools to Measure Security and Trustworthiness

Henrique Madeira,

University of Coimbra, Portugal

Workshop on Recent Advances
on Intrusion-Tolerant Systems (WRAITS)

Cascais, June 29th, 2009

University of Coimbra

Measuring trustworthiness

- Trustworthy ICT should be:
  - Dependable
  - Resilient
  - Secure

Require different metrics, methods and tools

to attacks, operational faults and changes

Quite different types of "animals"…

Measuring trustworthiness

- Trustworthy ICT should be:
  - Dependable
  - Resilient
  - Secure

  to attacks, operational faults and changes

Measuring, assessing, and benchmarking dependability and resilience is not solved…

Measuring, assessing and benchmarking security is even harder

Dependability and resilience metrics

- **Dependability**
  - **Availability** - readiness for correct service
  - **Reliability** - continuity of correct service
  - **Safety** - absence of catastrophic consequences
  - **Integrity** - absence of improper system alteration
  - **Maintainability** - ability to undergo modifications and repairs

  (Performance, measured as response time & throughput, is part of correct service)

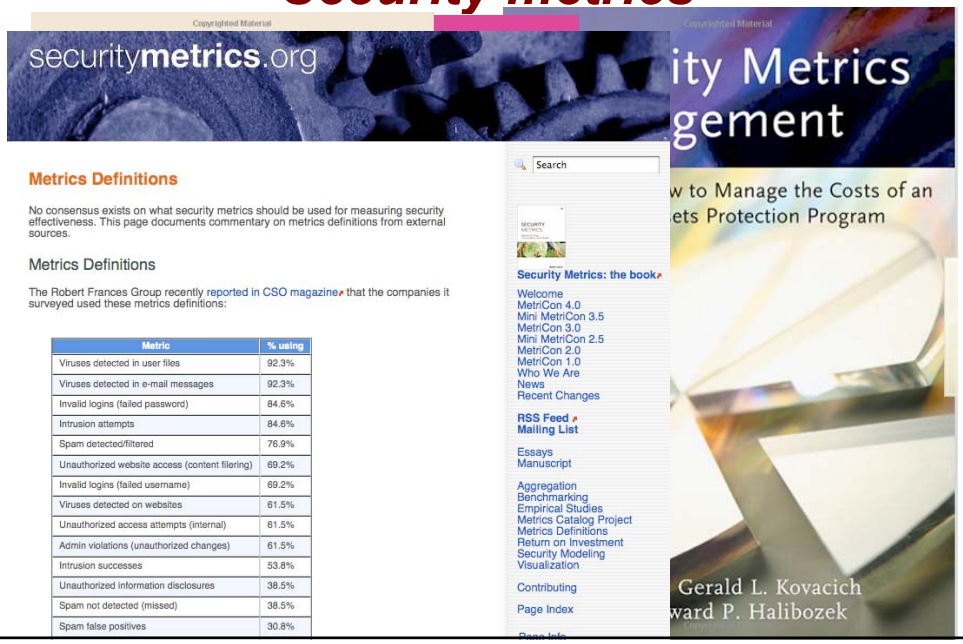
Fine! But just for faults!

- **Resilience**

  The persistence of delivery of justifiably **trusted service** in the presence of evolutionary changes. The changes can be planned, predictable, or totally unforeseen

(ReSIST - Resilience for Survivability in IST, www.resist-noe.org)

Security metrics



Security metrics

- The problem is not the lack of security metrics
- Different groups of metrics
  - Organizational Security Metrics
    - Measure the effectiveness of organizational programs and processes
    - Related to best practices
    - Specify levels of maturity
  - Technical Security Metrics
    - Access security attributes of ICT objects
    - Many ad hoc metrics… addressing quite specific aspects
  - Operational Security Metrics
    - Measure security risks based on metrics produced as a part of normal operations
    - Operational readiness
    - Security posture (e.g., measure antivirus protection, password strength,…)

Measuring trustworthiness

- Trustworthy ICT should be:
  - Dependable
  - Resilient
  - Secure

  to attacks, operational faults and changes

Measuring, assessing, and benchmarking dependability and resilience is not solved…

Measuring, assessing and benchmarking security is even harder

A single word is not enough…

Measuring, **assessing** and **benchmarking** trustworthiness

Measuring

The act of obtaining a proper measurement for a parameter or metric. It relies on a **quantitative** with well-known scale/reference

Measurement uncertainty can make the difference between good and bad measurements.

Quality of measurements is all about uncertainty evaluation.

Assessing

The act of classifying something with respect to its worth. It can just be **qualitative**.

Benchmarking

Agreement/contract and well-identified properties to ensure fairness in **comparison** *(DBench project)*.

What we have in mind when we talk about security measuring?

- Qualitative and quantitative measurements
- Quantitative measurement in two flavors:
  - **Relative sense** (benchmarking), to choose among alternatives
  - **Absolute sense**, to give guaranties to users
- Predictive value
- At different levels
  - **Organizational measurements**: measure effectiveness of organizational processes
  - **Technical measurements**: assess/compare technical objects
  - **Operational measurements**: produced as a part of normal operations
- Throughout systems lifecycle

Limitations of existing security measuring methods?

- In general, are focused on specific aspects of security.
- Lots of useful methods that help improving security, but none of them can quantify security.
  - Methods based on process guidelines and best practices
  - Red Teams, that may be effective in finding problems
  - Formal methods fail proving absolute security in realistic scenarios, although useful to find problems
  - Vulnerability detection approaches
  - Etc…
- Lack of predictive metrics and methods.
- Organizational-level and technical security metrics are not integrated to provide a comprehensive view

Measuring security; judging on trust

- Security is a feature of a system or a service

  (taking system/service in both the technical and organizational levels and including people as elements of the system)

- Trust is a relationship between two entities
  - Linked to confidence on something (a person, a machine, the related environment)
  - Dependent on the context
  - Varies along the time
  - Non-reflexive, non-symmetric, non-transitive
- Security measurements could provide support/evidences for the establishment of trust

Challenges

- **Challenge 1:** Define the right elements
  - Metrics
  - Evaluation methods
  - Tools
- **Challenge 2:** Validation
- **Challenge 3:** Impact and relevance

Challenge 1
Metrics, methods, and tools

- Appropriate metrics
- Methods for estimating metrics
- Effective evaluation tools

Challenge 1
Metrics, methods, and tools

- Appropriate metrics
- Methods f
- Effective e

- Which metrics? Just a small number? Specific of a given context? Metrics purpose (related to requirements)?
- Metrics at different levels: component, system, infrastructure, organization,…
- Product oriented and process oriented
- Applied throughout the system lifecycle
- Metric composition, namely how to relate organizational and technical metrics.

Challenge 1
Metrics, methods, and tools

- Appropriate metrics
- Methods for estimating metrics
- Ef

- Which methods? Formal, experimental, risk assessment, threat & vulnerability assessment, …
- Should be both model and experimentally based

Challenge 1
Metrics, methods, and tools

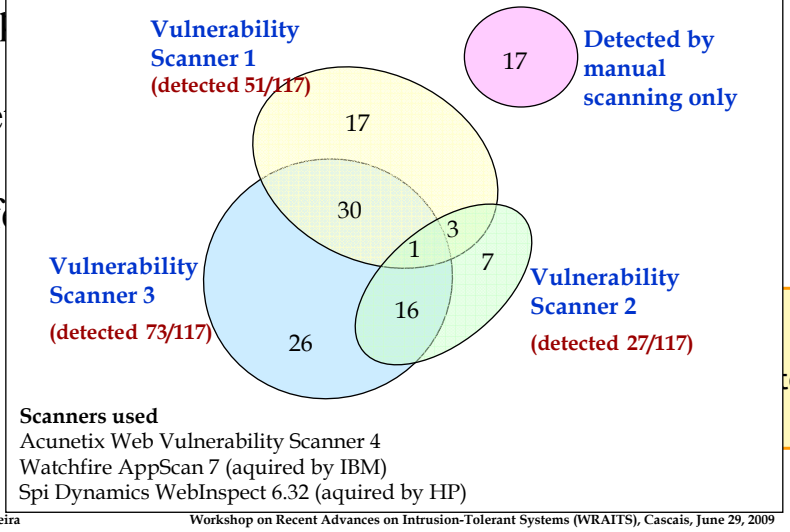
- Appropriate metrics?
- Methods for estimating metrics
- Effective evaluation tools

- Effective tools that can be used by practitioners
- Integration of different tools in a toolset, including tools for organizational and technical metrics

Challenge 1
Metrics, methods, and tools

- Ap
- Me
- Eff

Example of benchmarking vulnerability scanners tools



Scanners used
Acunetix Web Vulnerability Scanner 4
Watchfire AppScan 7 (aquired by IBM)
Spi Dynamics WebInspect 6.32 (aquired by HP)

tools

Challenge 2
Validation

- Properties
  - Representativeness
  - Repeatability
  - Reproducibility
  - Portability
  - Scalability
  - …
- Generalization
- Just being pragmatic is enough?

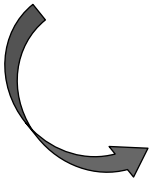
Challenge 3
Impact and relevance

- How to select the challenges and research problems that maximize the impact of research outcomes?
- Metrics that promote security improvement
  - User view
  - Industry
  - Metrics that show the return on investment
- Regulation policy
- Benchmarking

How can we give some contribution?
(back to 5 years ago)

Apply methods used in the dependability and resilience evaluation to the security field

Fault injection
Robustness testing
Dependability benchmarking



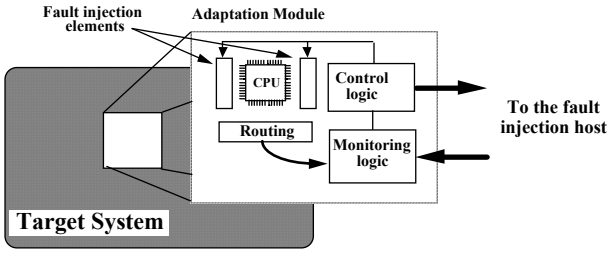
Vulnerability injection
Attack injection
Security benchmarking

What is fault injection?
(and a little bit of history…)

Deliberate insertion of upsets (faults or errors) in computer systems to evaluate its behavior in the presence of faults or validate specific fault tolerance mechanisms in computers.

Examples of fault injection approaches

- Pin-level fault injection **(e.g., RIFLE)**



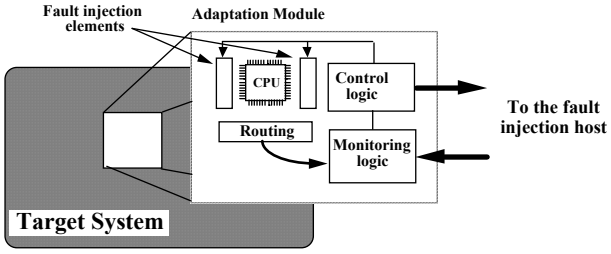
- Software Implemented Fault Injection **(e.g., Xception)**

  Reproduce pin-level fault injection by software

  Injection of hardware faults only!

Examples of fault injection approaches

- Pin-level fault injection **(e.g., RIFLE)**



- Software Implemented Fault Injection **(e.g., Xception)**

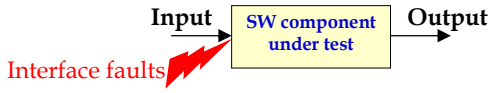
  Reproduce pin-level fault inject

  Injection of hardware faul

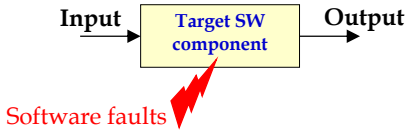
What about injecting software faults?

Two possible injection points (software faults)

1. Injection of interface faults in software components (classical robustness testing)



2. Injection of realistic software faults inside software components (new approach)



Example of results from interface faults (robustness testing)

Robustness failure in RTEMS 4.5.0

Excerpt of application code:

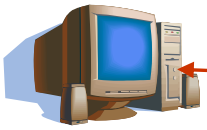
```
requestedSize1 = 4294967295;
returnStatus = rtems_region_get_segment (regionId,
                                         requestedSize1,
                                         option,
                                         timeout,
                                         ptsegment1);
```

Result:

```
Memory exception at fffffffc (illegal address)
Unexpected trap (0x09) at address 0x0200aaac
Data access exception at 0xfffffffc
```

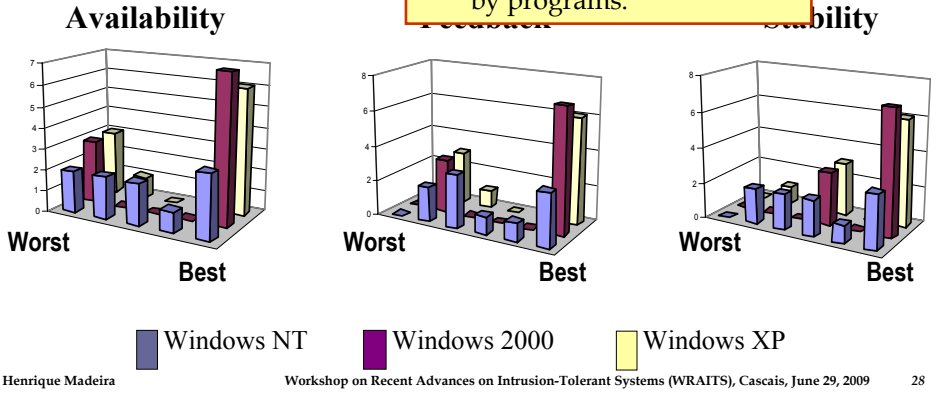
This software fault was discovered automatically by the Xception robustness testing tool!

Example of results obtained with injection of software faults

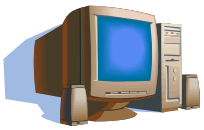


What h… g in the disk de… ve?

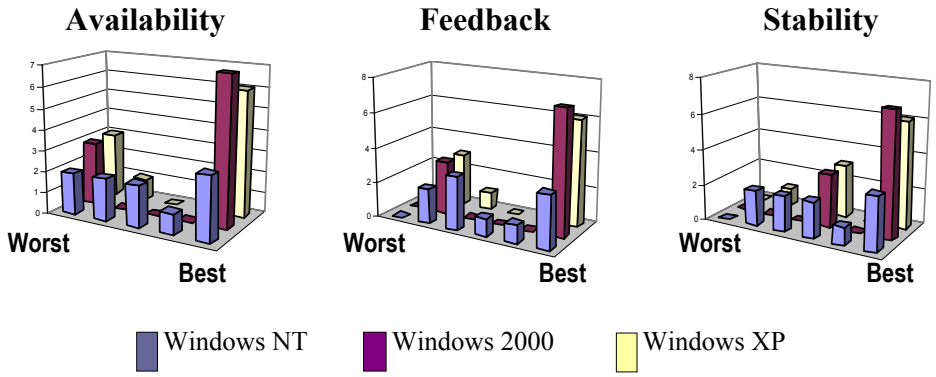
1. Software faults are injected in the disk device driver using the **G-SWFIT technique**.
2. The device is heavily used by programs.



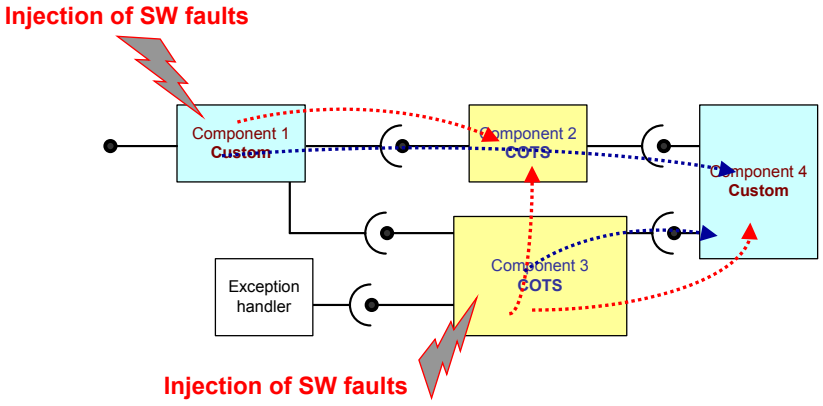
Example of results obtained with injection of software faults



What happens if a software bug in the disk device driver becomes active?



Why injection or real software faults?



- Error propagation through non conventional channels is a reality.
- Faults injected inside components are more representative.

Which are the most representative SW faults?

- Field data on real software errors is the most reliable information source on which faults should be injected
- Typically, this information is not made public
- Open source projects provide information on past (discovered) software faults

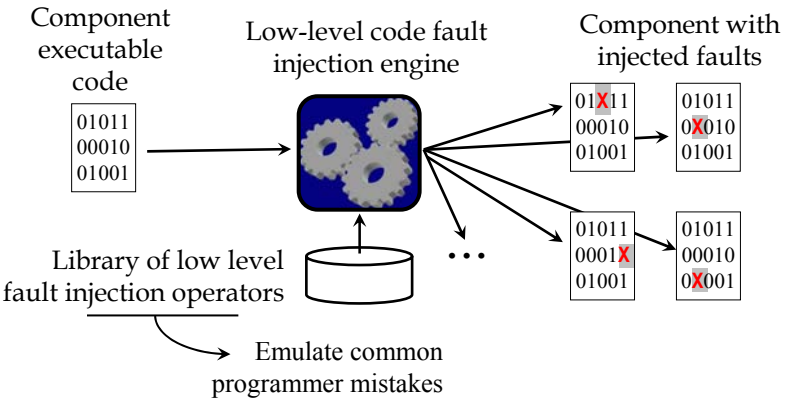
Open source field data survey

Programs	Description	# faults
CDEX	CD Digital audio data extractor.	**11**
Vim	Improved version of the UNIX vi editor.	**249**
FreeCiv	Multiplayer strategy game.	**53**
pdf2h	pdf to html format translator.	**20**
GAIM	All-in-one multi-protocol IM client.	**23**
Joe	Text editor similar to Wordstar®	**78**
ZSNES	SNES/Super Famicom emulator for x86.	**3**
Bash	GNU Project's Bourne Again SHell.	**2**
LKernel	Linux kernels 2.0.39 and 2.2.22	**93**
Total faults collected		**532**

The "Top-N" software faults

Fault types	Perc. Observed in field study	ODC classes
Missing "If (*cond*) { statement(s) }"	9.96 %	Algorithm
Missing function call	8.64 %	Algorithm
Missing "AND EXPR" in expression used as branch condition	7.89 %	Checking
Missing "if (*cond*)" surrounding statement(s)	4.32 %	Checking
Missing small and localized part of the algorithm	3.19 %	Algorithm
Missing variable assignment using an expression	3.00 %	Assignment
Wrong logical expression used as branch condition	3.00 %	Checking
Wrong value assigned to a value	2.44 %	Assignment
Missing variable initialization	2.25 %	Assignment
Missing variable assignment using a value	2.25 %	Assignment
Wrong arithmetic expression used in parameter of function call	2.25 %	Interface
Wrong variable used in parameter of function call	1.50 %	Interface
Total faults coverage	**50.69 %**	

G-SWFIT: Generic software fault injection technique



The technique can be applied to binary files prior to execution or to in-memory running processes

Software fault operators

- **Location pattern** – how to locate where to inject a software fault
- **Code change** – what to change in order to inject a software fault

Fault/operator ex 1: Missing and-expression in condition

Target source code (avail. not necessary)

```
if ( a==3 && b==4 )
{
  do something
}
```

→

Code with intended fault

```
if ( a==3 && b==4 )
{
  do something
}
```

Original target code (executable form)

```
cmp dword ptr off_a[ebp],3
jne short ahead
cmp dword ptr off_b[ebp],4
jne short ahead
; ... do something ...
ahead:
...

; remaining prog. code
```

→

Target code with emulated fault

```
cmp dword ptr off_a[ebp],3
jne short ahead
nop
nop
nop
; ... do something ...
ahead:
...
; remaining prog. code
```

The actual mutation is performed in executable (binary) code. Assembly mnemonics are presented here for readability sake

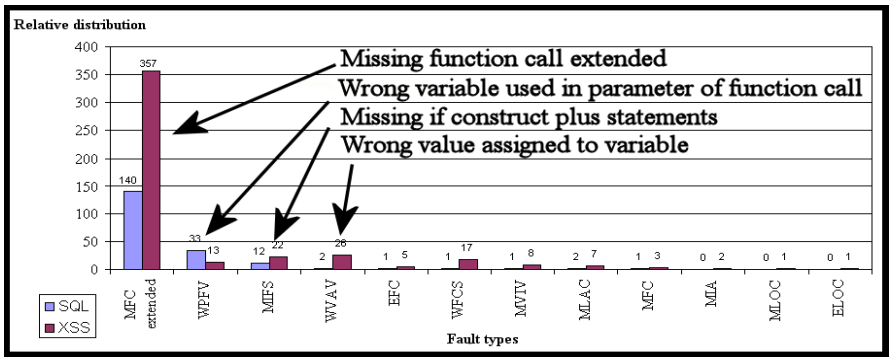
Vulnerability Injection (for Web Applications)

- Some possible scenarios:
  - Train security teams
  - Evaluate security teams
  - Estimation of vulnerabilities in the code
  - **Attack Injector**

    and with an attack injector we can evaluate security mechanisms…
  - …

Field Study on Vulnerabilities

- 6 LAMP web applications
- 655 security vulnerabilities (XSS and SQLi)

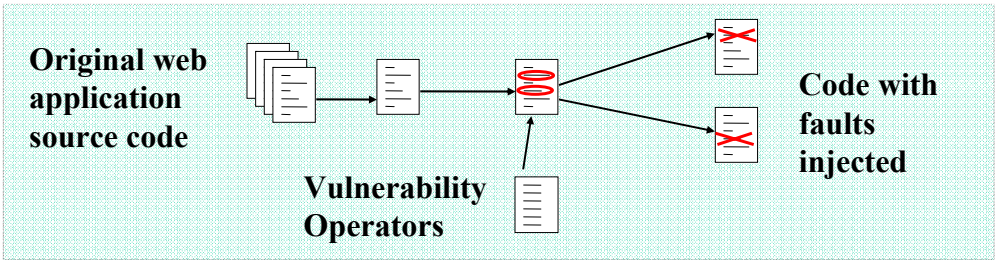


Vulnerability Operators

- **Location pattern** – how to locate where to inject a vulnerability
- **Vulnerability code change** – what to change in order to inject a vulnerability

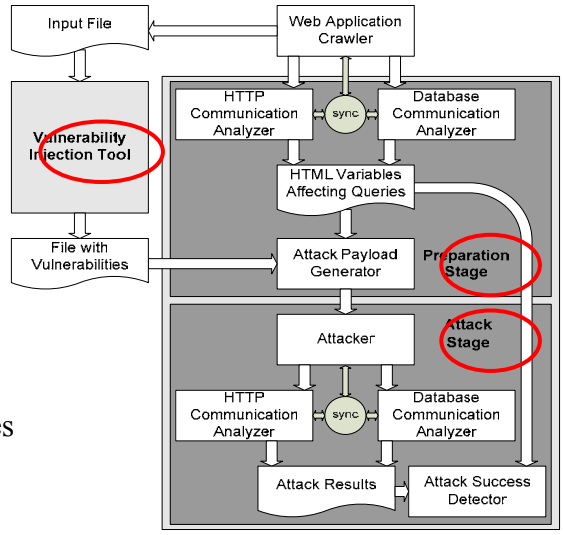
Vulnerability Injection methodology

1. Analysis of the source code of the web application
2. Search for the locations where a vulnerability may exist
3. Change the code to inject a vulnerability



Attack Injection Methodology

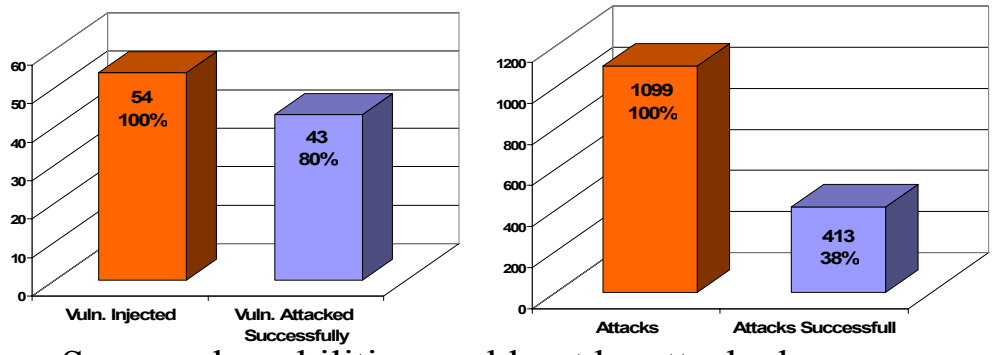
- Methodology to test security mechanisms:
  1. Injection of realistic vulnerabilities
  2. Controlled attack of the vulnerabilities
- Stages:
  1. Preparation
  2. Injection of Vulnerabilities
  3. Attack



Simple example

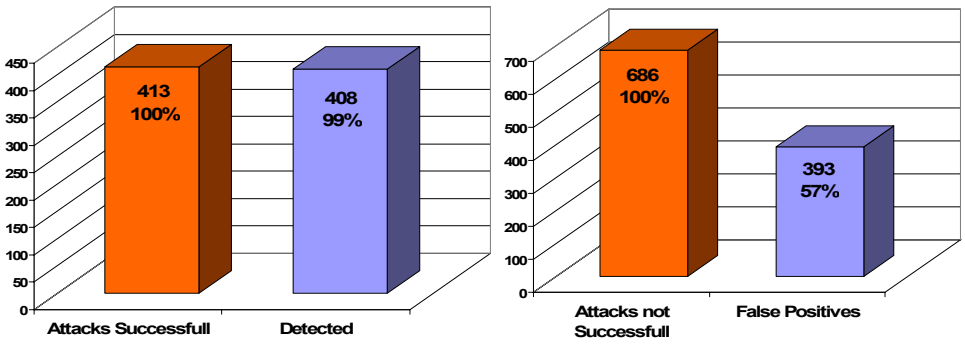
1. Verify the quality of the vulnerabilities and attacks injected
2. Test one IDS for databases

1: Verification of the Injection



- Some vulnerabilities could not be attacked:
  - Multiple protection of the variable
  - Different variables with the same name

2: IDS Evaluation



The IDS missed five attacks due to a bug in the code of one of its core functions

Conclusion

- Measuring, assessing and benchmarking security is a big challenge
- Metrics, methods, and tools
  - What is required?
  - Who is measuring and who is going to use the measurements? There are a lot of stakeholders…
  - Validation?
  - Trusting on the tools?
- Learn with "old" dependability evaluation techniques could be useful