A clustering Approach for Web Vulnerabilities Detection

Mohamed Kaâniche

Rim Akrout, Eric Alata, Yann Bachy,
Anthony Dessiatnikoff, Vincent Nicomette

LAAS-CNRS

Design and
Assessment of application
Level
Intrusion detection systems

IFIP WG 10.4 meeting, Martinique, 26-30 January 2012

Outline

- Context and motivation
- Web vulnerability scanners
- Proposed approach
- Experimental results
- Conclusion

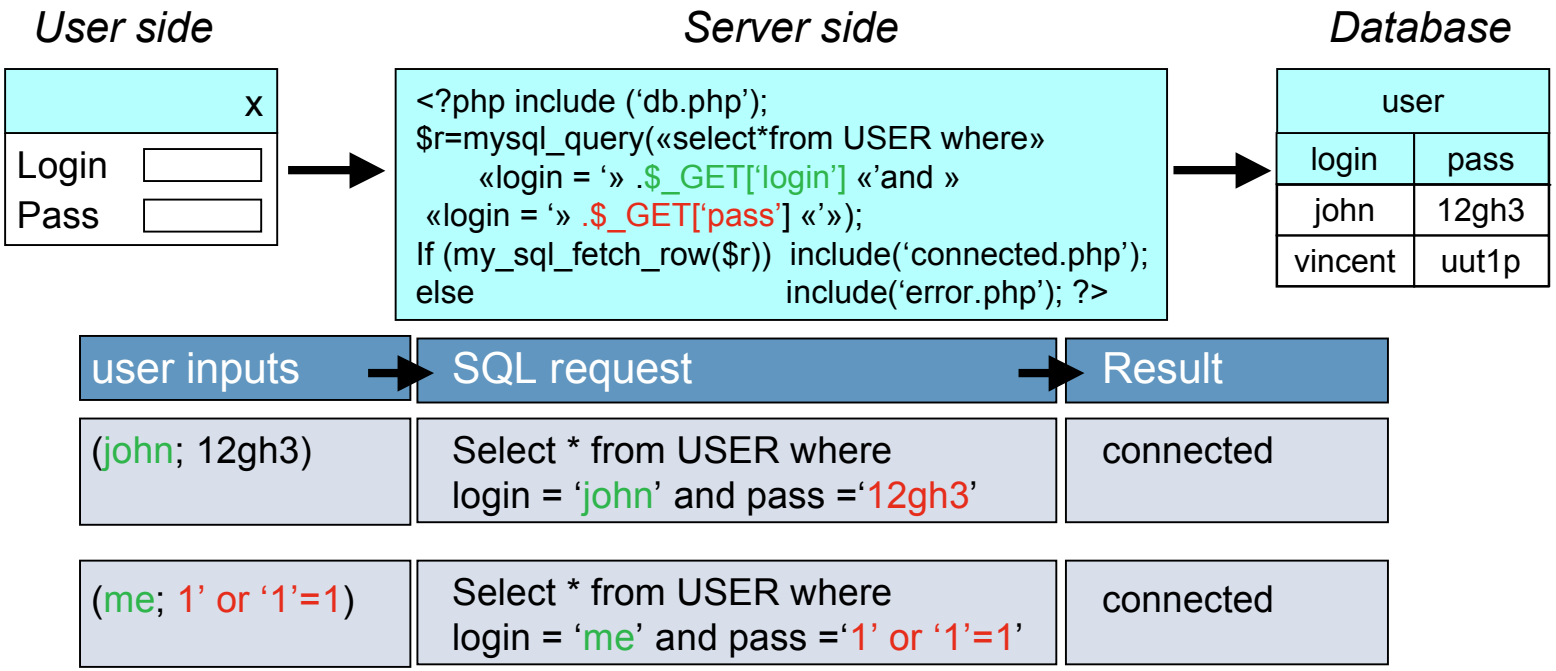
Context

- Web vulnerabilities have become a major threat to information systems security
  - SQL Injections, XPATH, OS commanding, XSS,…
  - Lack of sanitization of URL parameters, html form inputs, ...
- Attackers can gain unauthorized access, read/modify sensitive data, perform DoS attacks, …

Context

- Web vulnerabilities have become a major threat to information systems security
  - SQL Injections, XPATH, OS commanding, XSS,…
  - Lack of sanitization of URL parameters, html form inputs, ...
- Attackers can gain unauthorized access, read/modify sensitive data, perform DoS attacks, …

SQL injection



Techniques to cope with vulnerabilities

- Identification and possible sanitization at runtime of malicious requests
  - Web application Firewalls, IDS
- Static analysis of the source code
- Web vulnerability scanners
  - Black box security testing
  - identify injection points and generate specially crafted inputs to detect vulnerabilities

Web vulnerability scanners

- Commercial tools
  - WebInspect, AppScan, Acunetix, …
- Open source and publicly available tools
  - Skipfish: *http://code.google.com/p/skipfish*
  - W3aF: *http://w3af.sourceforge.net*
  - Wapiti: *http://wapiti.sourceforge.net ; …*
- Several experimental analyses point out the need to improve the detection efficiency and the automation capabilities of existing tools
  - [Fonseca et al. 2007; PRDC-2007], [Bau et al. 2010; Symp. Security & Privacy]; [Doupé et al. 2010, DIMVA], etc.
- Research objectives
  - contribute to fulfill this gap

Existing approaches

- General principle
  - Submit through injection point a few crafted inputs and conclude about vulnerability existence based on server reponses analysis
- Error pattern matching: W3aF, Wapiti, Secubat
  - Search for some predefined error messages
- Similarity analysis of server reponses: skipfish
  - 3 requests for each injection point

  - a vulnerability exists if the responses associated to R_1 and R_2 are different and those associated to R_1 and R_3 are different
- General comments
  - Only a few requests are generally submitted (two or three)
  - None of the investigated tools provides the requests that successfully exploit the identified vulnerability

Discussion

- Only a few requests are generally submitted
  - two or three
- None of the investigated tools provides the requests that successfully exploit the identified vulnerability
- Contribution: a new vulnerability detection algorithm based on the similarity analysis approach:
  - Generation of a large number of requests that can be tuned by the user to potentially achieve a higher coverage of the server responses space
    - Grammars specific to vulnerability classes
  - Automatic identification of successful injections based on the hierarchical clustering of similar server response pages

Proposed approach

- Generate requests that would likely produce failed execution pages (*rejection* pages)
  - R1: Randomly generated login-passwords

```
http://address/directory/page.php?
        login=ABCDEF&pass=ABCDEF
```

Proposed approach

- Generate requests that would likely produce failed execution pages (*rejection* pages)
  - R1: Randomly generated login-passwords
  - R2: Syntaxically invalid injections

```
http://address/directory/page.php?
        login='''&pass='''
```

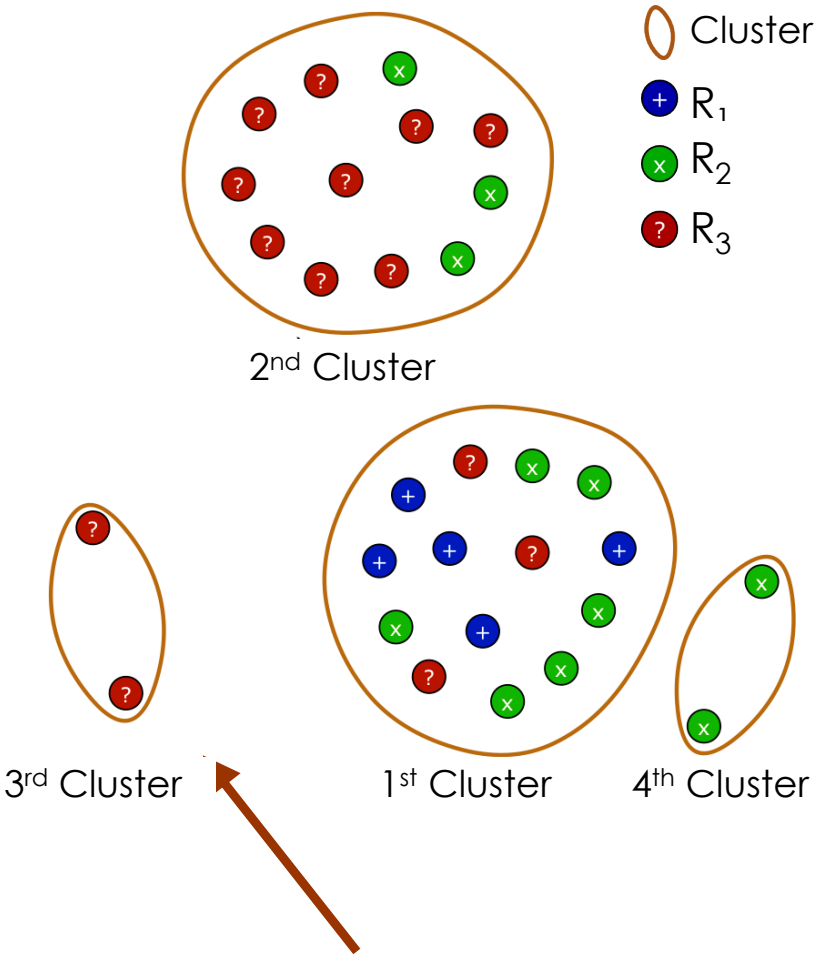
Proposed approach

- Generate requests that would likely produce failed execution pages (*rejection* pages)
  - R1: Randomly generated login-passwords
  - R2: Syntaxically invalid injections
- Generate syntaxically valid injections : R3

```
INJECTION   :=   WORD ‘ POR TAU [‘ POR TAU
            |    WORD " POR TAU [" POR TAU
POR         :=   ‘ or ‘
            |    ‘)’ POR ‘(‘
TAU         :=   Hex(‘A’)=’41
            |    ‘1’=‘1
            |    ’[f-m]’ between ’[a-e]’ and ’[n-z]’
WORD        :=   [0-9a-zA-Z]*
…
```

Proposed approach

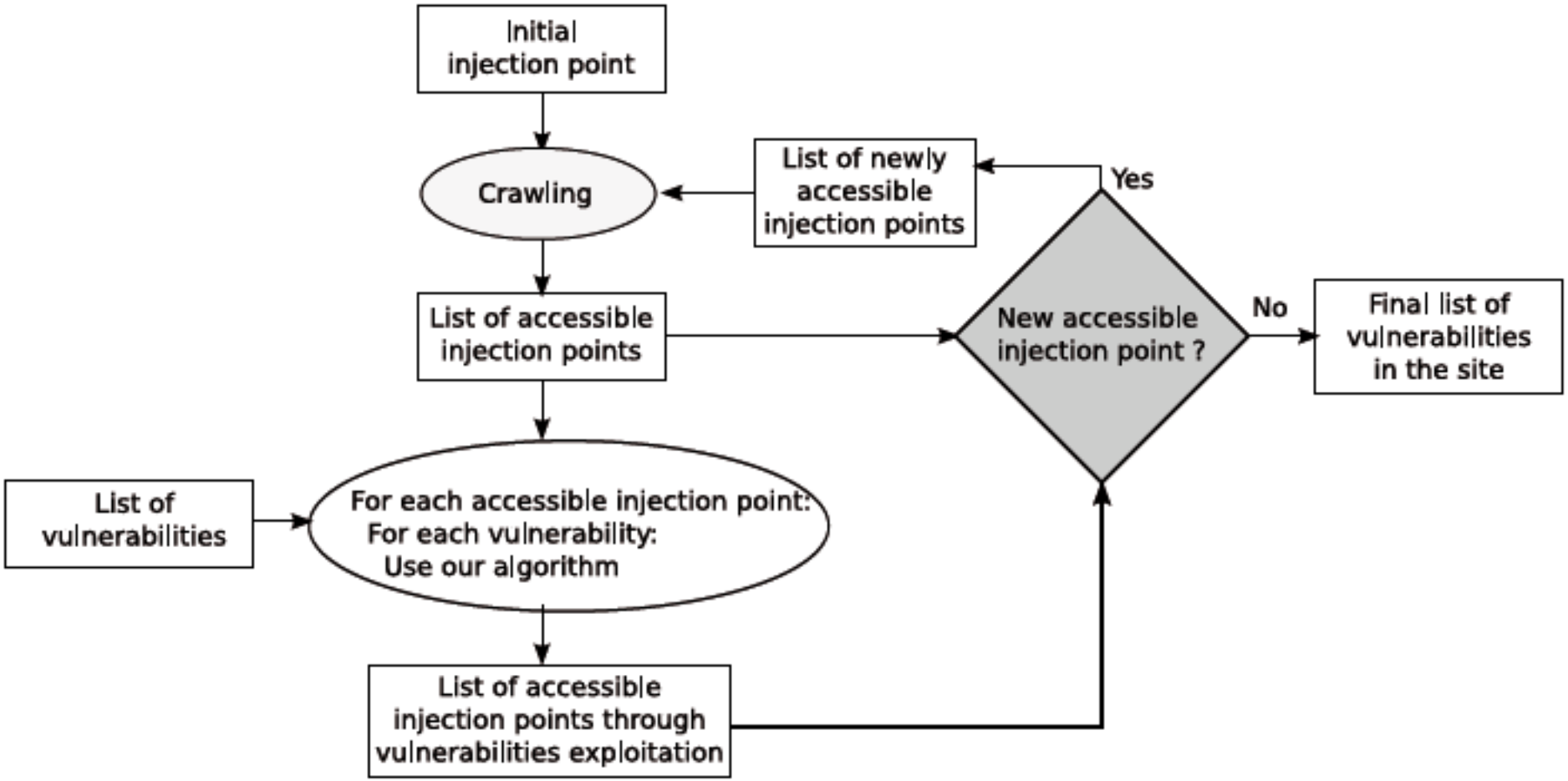
- Generate requests that would likely produce failed execution pages (*rejection* pages)
  - R1: Randomly generated login-passwords
  - R2: Syntaxically invalid injections
- Generate syntaxically valid injections : R3
- Identify successful injections based on the similarity analysis of R3 responses compared to R1 and R2 responses



Clusters with only syntactically valid SQL injections (R3) identify successfull injections

Algorithm

- Entry point: URL of the web application



Experimental assessment

- WASAPY tool
  - Web Application Security Assessment in Python
- Comparative analysis with open source vulnerability scanners: skipfish, Wapiti, W3af
- Two types of experiments
  - Modified applications including specific injected vulnerabilities
  - Publicly available vulnerable applications without modification
    - For some of these, results available for commercial tools in the literature: AppScan,WebInspect, Acunetix
- Experimental environment
  - Scanners: skipfish 1.9.6b; Wapiti 2.2.1, W3af 1.1
  - Gnu/Linux (2.6 kernel) host running several virtual machines
  - Apache Web server (1.3.37/ 2.2.8/4.0.0/5.0.0)
  - MySQL database server 5

Modified applications

Vulnerable applications

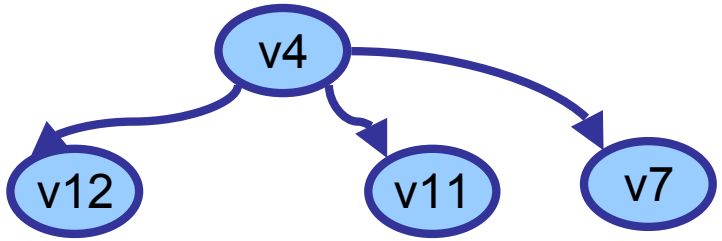
- phpBB3 (PHP/MySQL)
- SecurePages (PHP/MySQL)
- HardwareStore (PHP/MySQL)
- Insecure (Ruby On Rails)
- Damn Vulnerable Web Application (PHP/MySQL)

Legend

- ✓ Tested & detected vulnerability
- ✗ Tested & not detected vulnerability
- – Injection point not tested by the scanner

Vulnerabilities			Scanners			
Type	Application	*ID*	Skipfish	W3af	Wapiti	Wasapy
SQL*i*	phpBB3	v1	✗	✗	✓	✓
	SecurePages	v2	✗	✗	✓	✓
	HardwareStore	v3	✓	✓	✓	✓
		v4	✓	✓	✗	✓
		v5	✓	✗	✗	✓
		v6	✗	✗	✗	✓
		v7	–	–	–	✓
	Insecure	v8	✓	✓	✗	✓
	DVWA	v9	✓	✓	–	✓
XPa	HardwareStore	v10	✗	✗	✗	✓
OsC	HardwareStore	v11	–	–	–	✓
FIn	HardwareStore	v12	–	–	–	✓
Nombre de détection			5	4	3	12

- *v2 functionally similar to v1*
- *v8 functionally similar to v4*
- *v9 functionally similar to v3*



Non modified vulnerable applications

Vulnerability			Skipfish	W3af	Wapiti	Wasapy
Type	CVE	Location				
SQL*i*	*NR*	search.php	✓	✓	✓	✓
	2005-3236	lostpwd.php	✓	✓	✓	✓
	2005-3236	newmsg.php	✓	✓	✓	✓
	2005-3575	show.php	✓	✓	✓	✓
False positive			1	0	0	0

Cyphor

Vulnerability			Skipfish	W3af	Wapiti	Wasapy	AppScan	WebInspect	Acunetix
Type	CVE	Location							
OsC	*NR*	index.php	✗	✓	✗	✓	✗	✗	✗
False positive			0	0	0	0	0	0	0

Ftss

Vulnerability			Skipfish	W3af	Wapiti	Wasapy	AppScan	WebInspect	Acunetix
Type	CVE	Location							
SQL*i*	*NR*	edit_post.php	✗	✗	✗	✓	✗	✗	✗
	NR	edit_post_script.php	✗	✗	✗	✗	✗	✗	✗
	NR	index.php	✗	✗	✗	✗	✗	✗	✗
	NR	message.php	✗	✗	✗	✓	✗	✗	✗
	NR	reader.php	✓	✓	✗	✓	✗	✗	✗
False positive			0	0	0	0	0	0	0

Riotpix

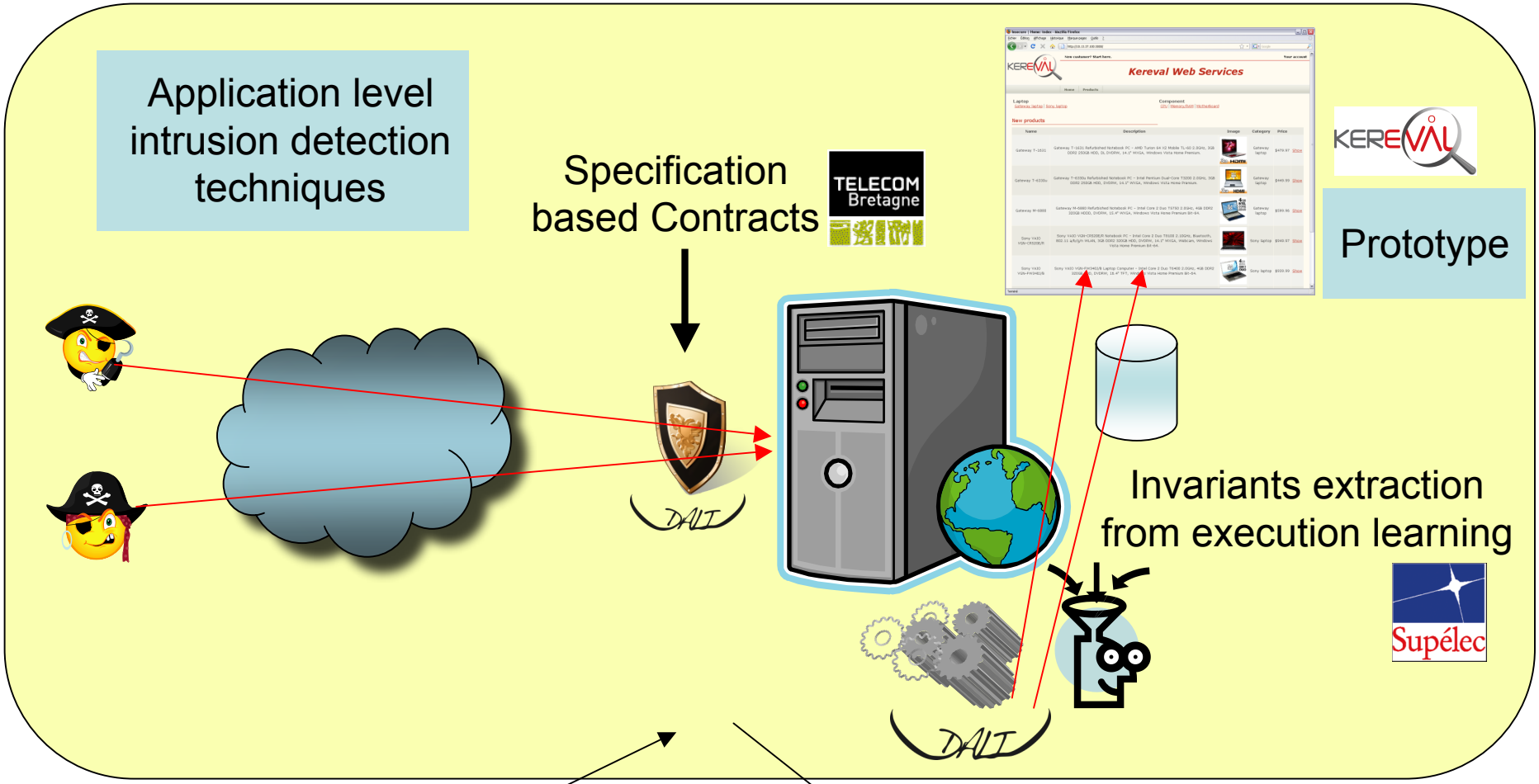
Vulnerability			Skipfish	W3af	Wapiti	Wasapy	AppScan	WebInspect	Acunetix
Type	CVE	Location							
SQL*i*	*2008-7091*	login.php	✗	✗	✗	✓	✗	✓	✗
	2008-7091	story.php	✓	✗	✓	✓	✓	✓	✓
	NR	userrss.php	✗	✗	✗	✗	✓	✓	✓
	2008-7091	out.php	✗	✗	✗	✗	✓	✗	✓
	2008-7091	trackback.php	✗	✗	✗	✗	✗	✗	✗
	2008-7091	cloud.php	✗	✗	✗	✗	✗	✗	✗
	2008-7091	cvote.php	✗	✗	✗	✗	✗	✗	✗
	2008-7091	recommend.php	✗	✗	✗	✗	✗	✗	✗
	2008-7091	submit.php	✗	✗	✗	✗	✗	✗	✗
	2008-7091	vote.php	✗	✗	✗	✗	✗	✗	✗
	2008-7091	edit.php	✗	✗	✗	✗	✗	✗	✗
False positive			0	0	0	2	1	1	0

Pligg

Conclusion and future work

- Contributions
  - Novel approach for web vulnerability detection
    - Automatic identification of successful attacks
    - SQL injections, XPATH, OS Commanding, File Include
  - Promising results that need to be confirmed by further experiments
  - Suitable for vulnerabilities that modify the response page returned to the user (not for XSS)
- Current and Future work
  - More extensive validation experiments
  - Generate attacks scenarios taking into account dependencies between vulnerabilities
  - Evaluation of web applications IDS (Dali project)

DALI project



Assessment methodology and tools

Experiments

- normal traffic
- attacks

Results

- IDS efficiency
- Comparative analysis