Design and
Assessment of application
Level
Intrusion detection systems

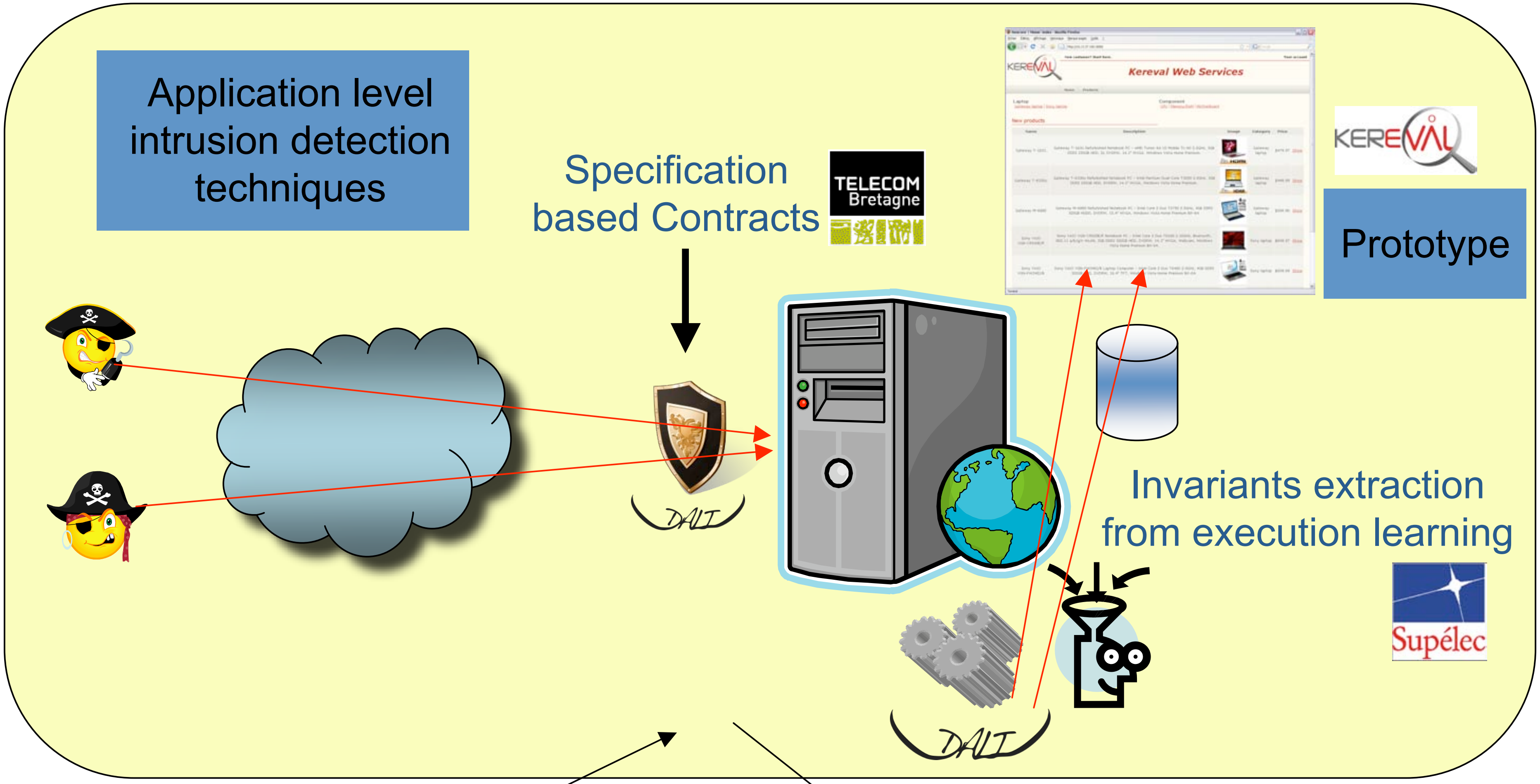
Web applications security assessment

Mohamed Kaâniche

Eric Alata, Rim Akrout, Anthony Dessiatnikoff,
Yves Deswarte, Karama Kanoun, Vincent Nicomette, Hélène Waeselynck

LAAS-CNRS

DALI: Context and Objectives



Application level intrusion detection techniques

Specification based Contracts

TELECOM Bretagne

KEREVAL

Prototype

Invariants extraction from execution learning

Supélec

Assessment methodology and tools

LAAS-CNRS

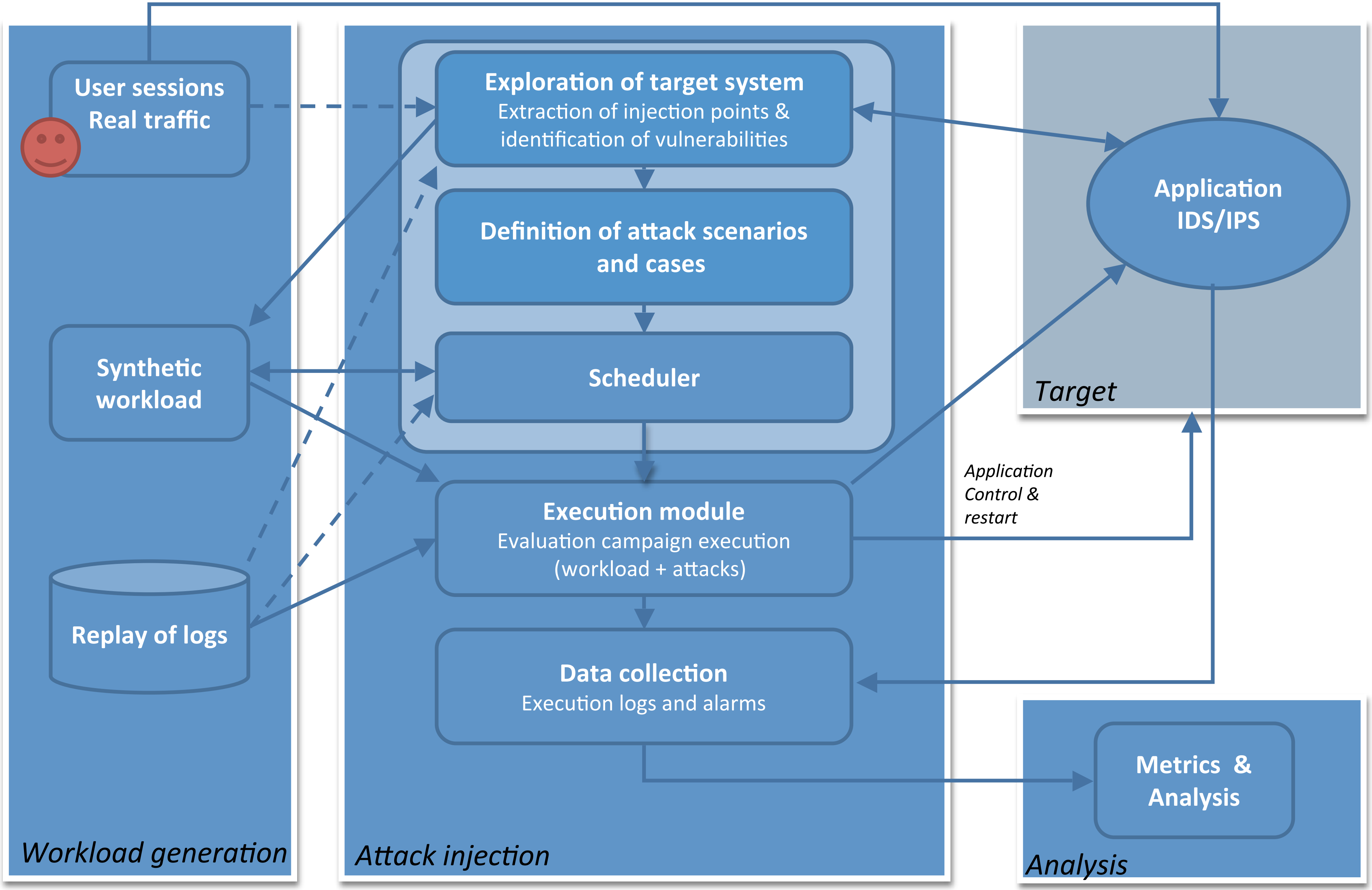
Experiments

- normal traffic
- attacks

Results

- IDS efficiency
- Comparative analysis

Assessment framework



User sessions
Real traffic

Synthetic workload

Replay of logs

Workload generation

Exploration of target system
Extraction of injection points & identification of vulnerabilities

Definition of attack scenarios and cases

Scheduler

Execution module
Evaluation campaign execution (workload + attacks)

Data collection
Execution logs and alarms

Attack injection

Application IDS/IPS

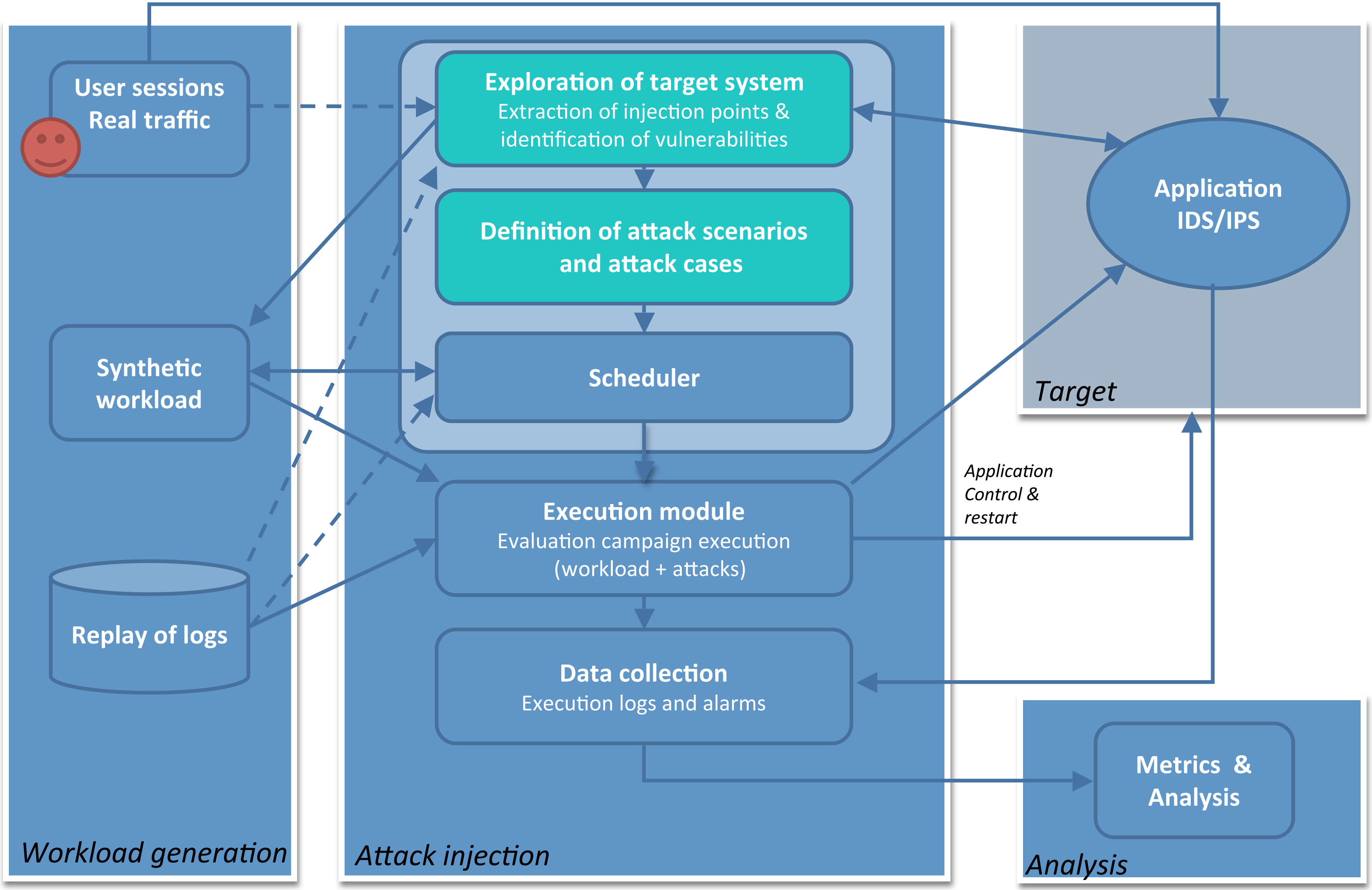
Target

Application Control & restart

Metrics & Analysis

Analysis

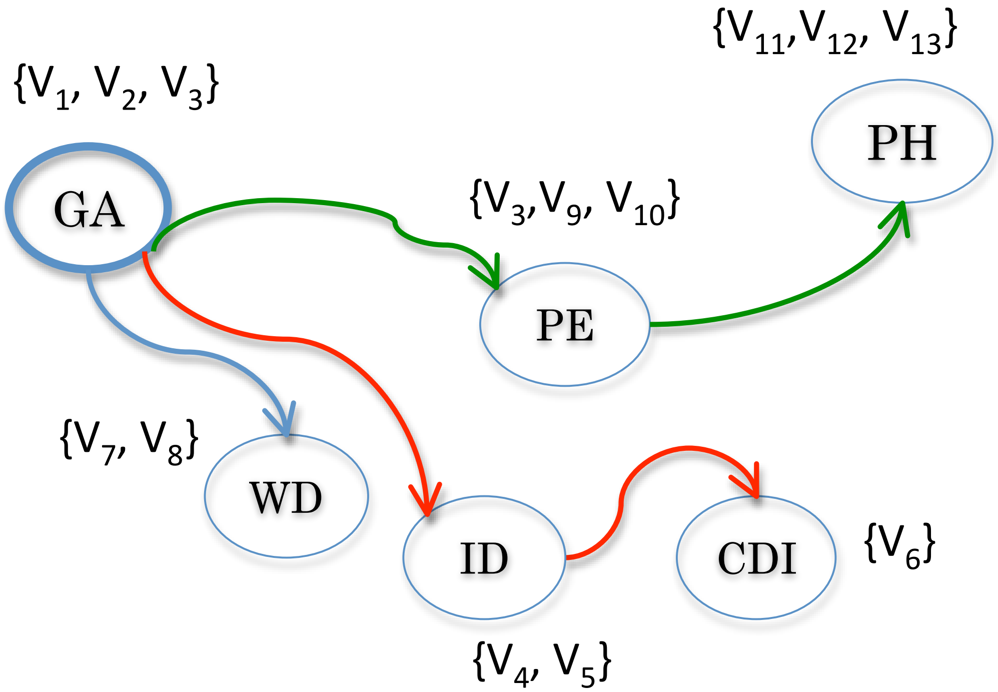
Assessment framework


User sessions
Real traffic
Synthetic workload
Replay of logs
Workload generation
Exploration of target system
Extraction of injection points & identification of vulnerabilities
Definition of attack scenarios and attack cases
Scheduler
Execution module
Evaluation campaign execution (workload + attacks)
Data collection
Execution logs and alarms
Attack injection
Application IDS/IPS
Target
Application Control & restart
Metrics & Analysis
Analysis

Attack goals and scenarios

Attack Goals:

- Gain Access (GA)
- Privilege escalation (PE)
- Information Disclosure (ID)
- Denial of Service (DoS)
- Compromise data integrity (CDI)
- Web site defacement (WD)
- Phishing (PH)
- …



Attack Scenarios:

Sc1: GA, ID, CDI

Sc2: GA, DW

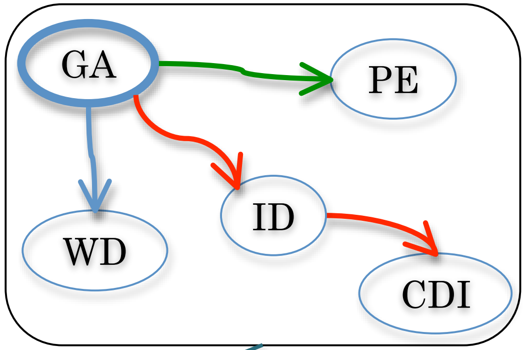
Sc3: GA, AP, TV

→ Several possible instantiations for each scenario

Generation of attack campaign

Vulnerabilities ↔ Goals
+ Dependencies

$\{V_1,\ldots,V_n\} \leftrightarrow \{G_1,\ldots,G_m\}$



Abstract Scenarios

	V_1	V_2	V_3	…	V_n
G_1: GA	X	X			X
G_2: VI			X	x	
….					
GO_m:CI D					X

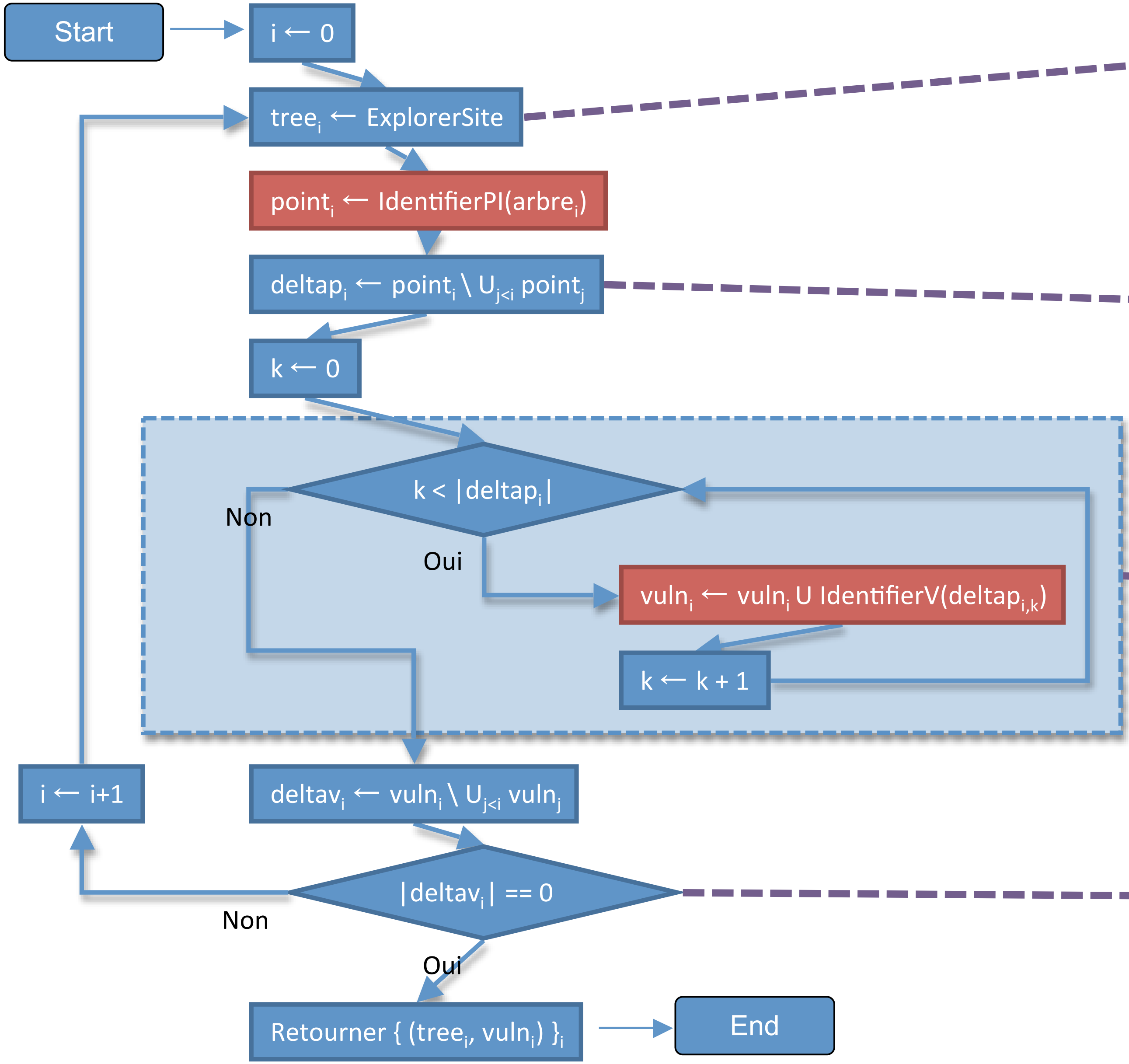
Definition of attack campaign

- Strategies: for each attack goal
  - Test at least one vulnerability
  - Test all vulnerabilities, …

Attack cases: Sequence of vulnerabilities instantiating each scenario $\{S_1, \ldots S_k\}$

S_1: GA (V_1), ID (V_4), CDI (V_6)
S_2: GA (V_2), ID (V_4), CDI (V_6)
S_3: GA (V_3), ID (V_5), CDI (V_6)

Injection points & Vulnerabilities Identification



Start
$i \leftarrow 0$
$tree_i \leftarrow$ ExplorerSite
$point_i \leftarrow$ IdentifierPI($arbre_i$)
$deltap_i \leftarrow point_i \setminus \cup_{j<i} point_j$
$k \leftarrow 0$
$k < |deltap_i|$
Non
Oui
$vuln_i \leftarrow vuln_i \cup$ IdentifierV($deltap_{i,k}$)
$k \leftarrow k + 1$
$deltav_i \leftarrow vuln_i \setminus \cup_{j<i} vuln_j$
$i \leftarrow i+1$
$|deltav_i| == 0$
Non
Oui
Retourner { ($tree_i$, $vuln_i$) }$_i$
End

Crawler: HTML pages structure

Identification of **new injection points**: *forms, URL parameters, …*

Vulnerability identification for each injection point. *Clustering algorithm*

Exploitation of new vulnerabilities can reveal new injection points

Vulnerability Identification Algorithm

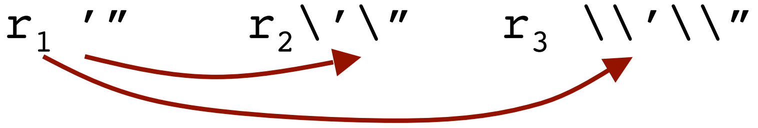
- input: injection point
- output: vulnerabilities associated to injection point

- Objective
  - automate identification
  - provide requests allowing exploitation of vulnerabilities
  - focus on SQL injections at a first step

- *open source* vulnerability scanners: *Skipfish*, *W3af*, Wapiti
  - *http://code.google.com/p/skipfish*
  - *http://w3af.sourceforge.net*
  - *http://wapiti.sourceforge.net*

Vulnerability detection algorithms

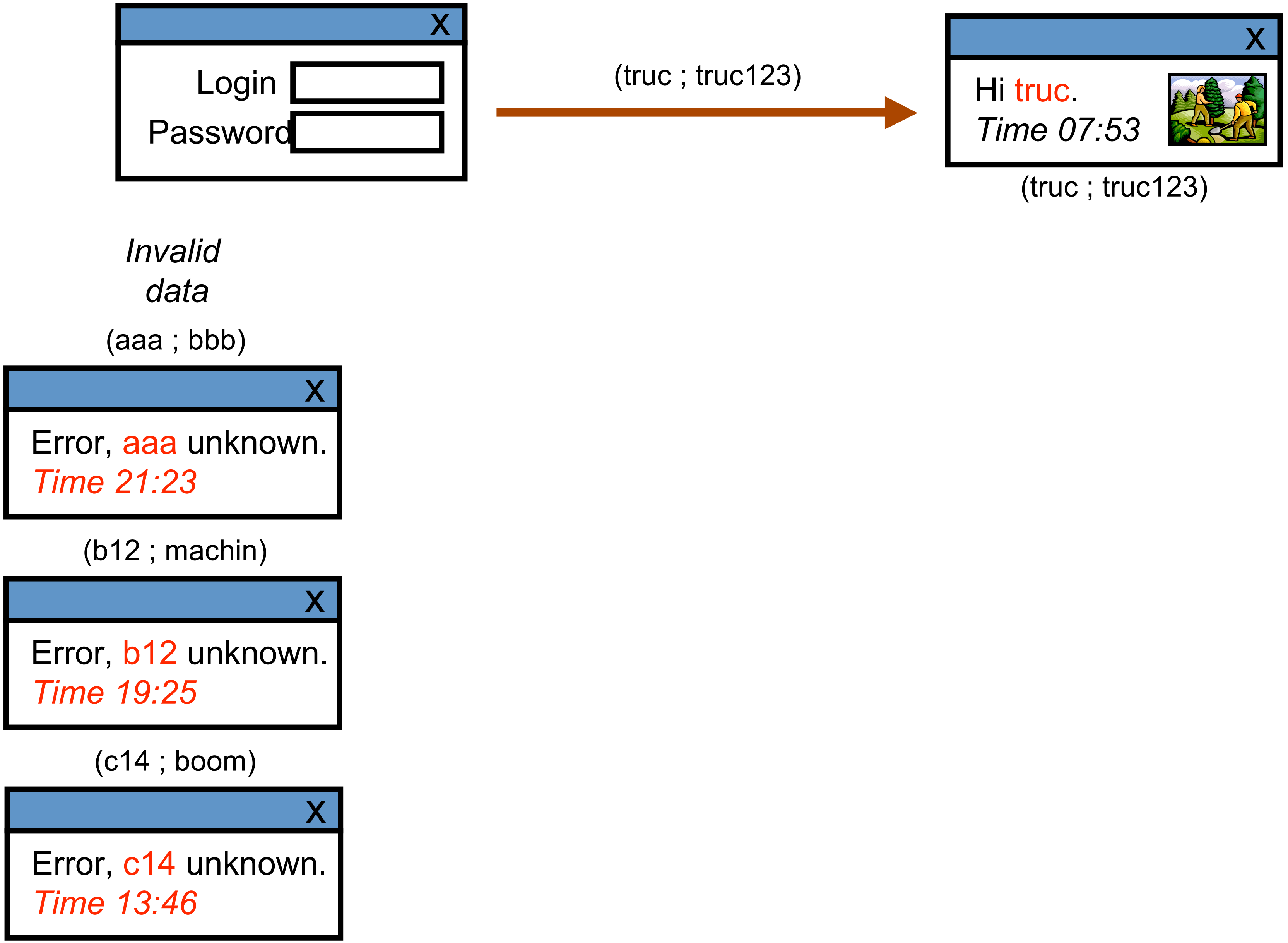
- ❑ *skipfish*
  - – 3 requests for each injection point



r_1 ’” r_2 \’\” r_3 \\’\\”

  - – a vulnerability exists whenever the responses associated to r_1 and r_2 are different and the responses associated to r_1 and r_3 are different
  - – Similarity analysis: frequency of words
  - – Assumption
    - • Different invalid requests return similar error responses
- ❑ *W3aF and Wapiti*
  - – Pattern matching of MySQL error messages

Intuition



(truc ; truc123)

Hi truc.
Time 07:53

(truc ; truc123)

Invalid data

(aaa ; bbb)

Error, aaa unknown.
Time 21:23

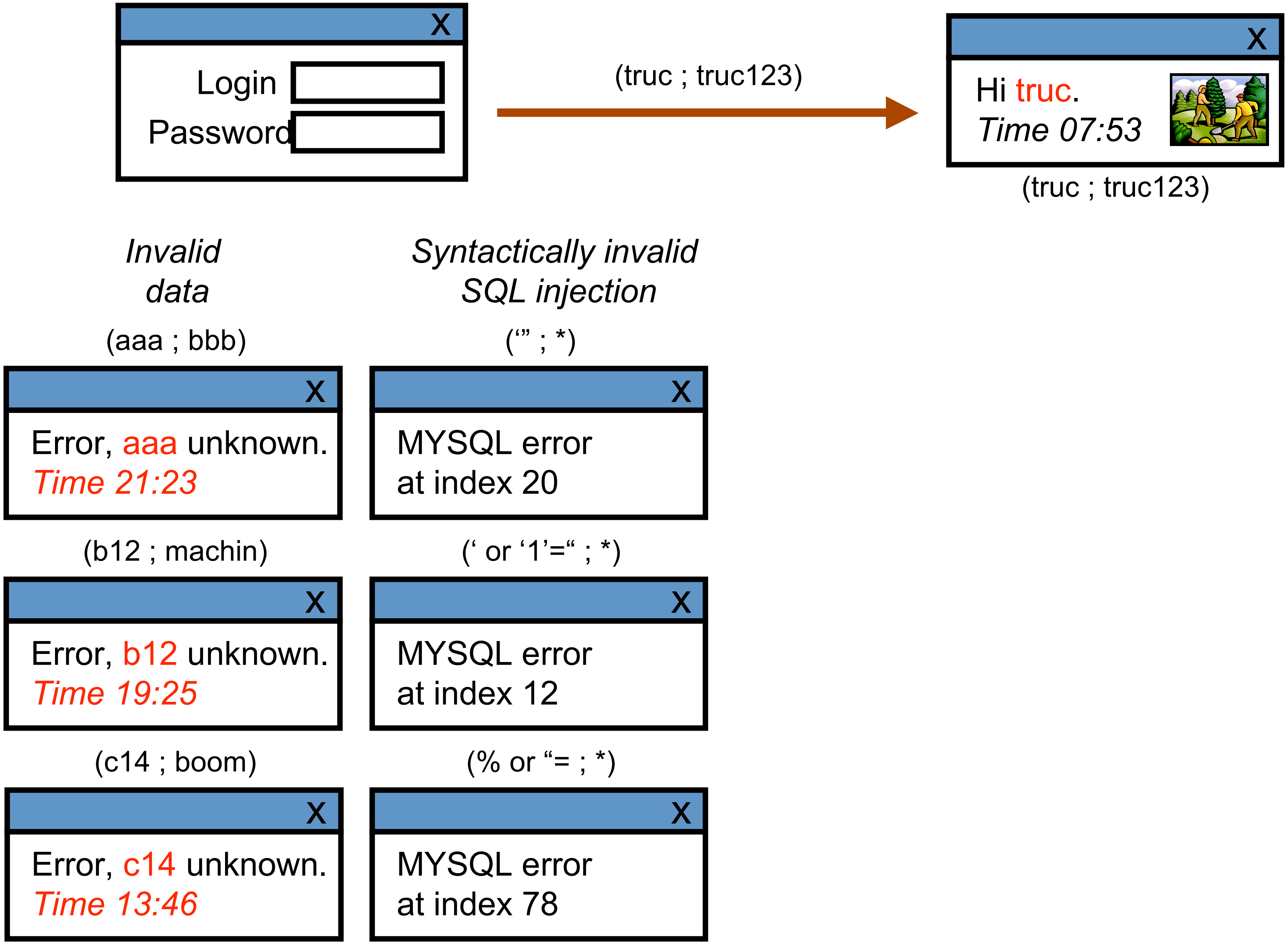
(b12 ; machin)

Error, b12 unknown.
Time 19:25

(c14 ; boom)

Error, c14 unknown.
Time 13:46

Intuition



(truc ; truc123)

Hi truc.
Time 07:53

(truc ; truc123)

Invalid data

(aaa ; bbb)

Error, aaa unknown.
Time 21:23

(b12 ; machin)

Error, b12 unknown.
Time 19:25

(c14 ; boom)

Error, c14 unknown.
Time 13:46

Syntactically invalid SQL injection

(“” ; *)

MYSQL error
at index 20

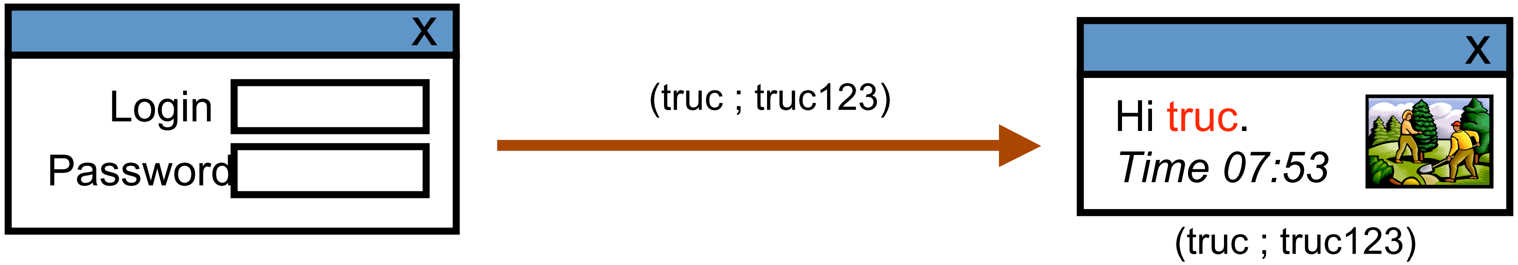
(‘ or ‘1’=“ ; *)

MYSQL error
at index 12

(% or “= ; *)

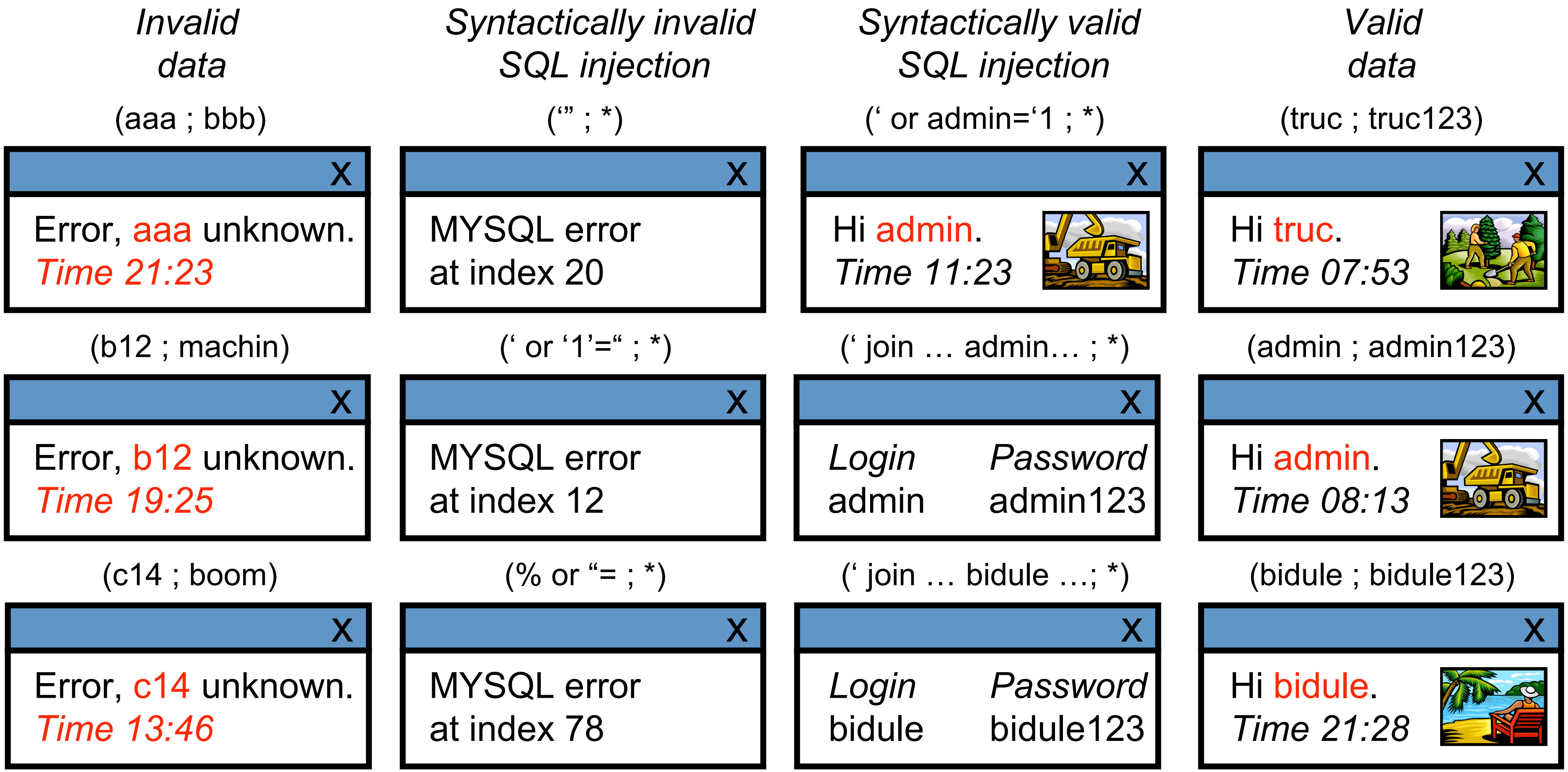
MYSQL error
at index 78

Intuition



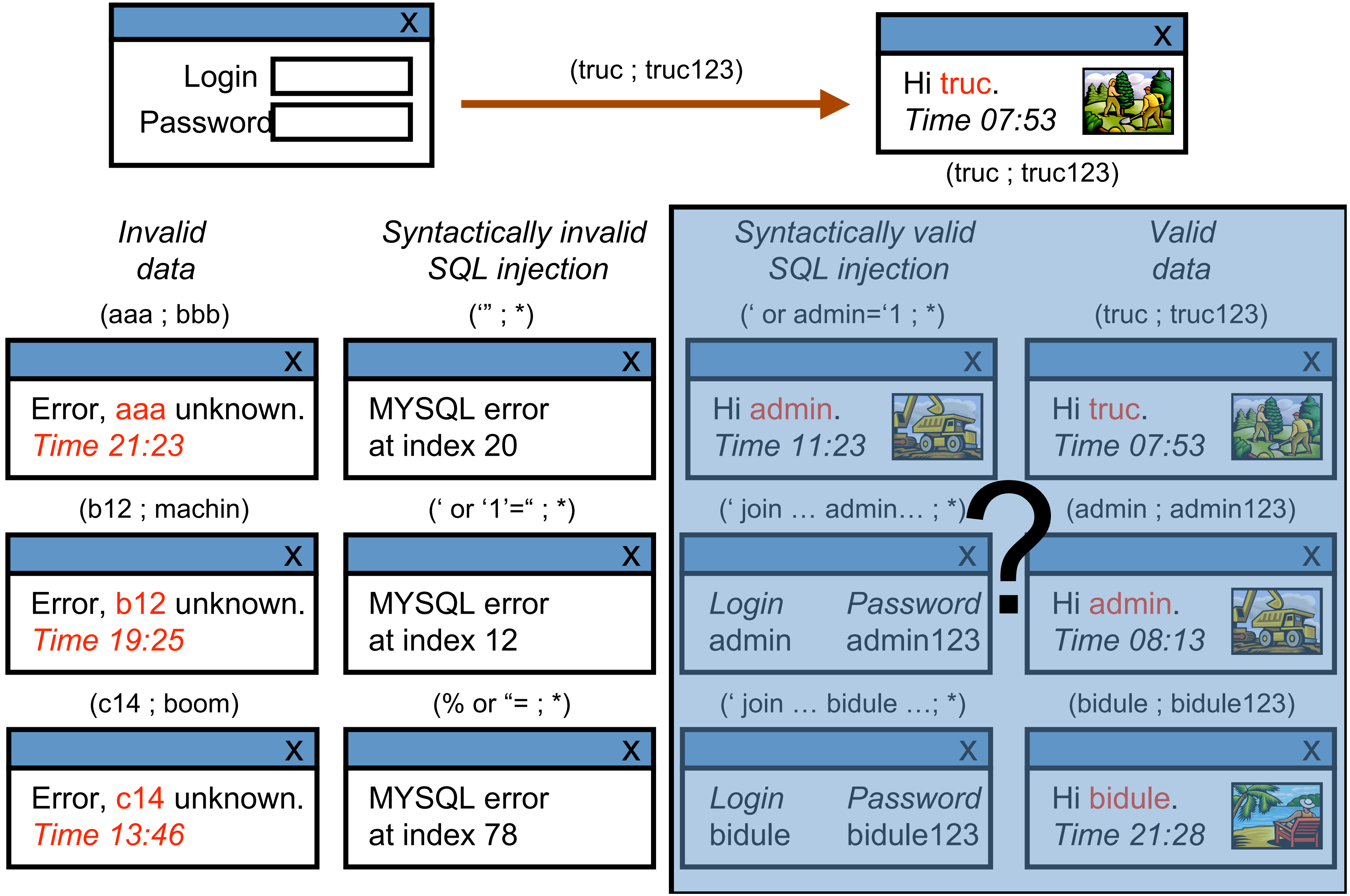
Hi truc.
Time 07:53

(truc ; truc123)



Invalid data	*Syntactically invalid SQL injection*	*Syntactically valid SQL injection*	*Valid data*
(aaa ; bbb)	(‘” ; *)	(‘ or admin=‘1 ; *)	(truc ; truc123)
Error, aaa unknown. *Time 21:23*	MYSQL error at index 20	Hi admin. *Time 11:23*	Hi truc. *Time 07:53*
(b12 ; machin)	(‘ or ‘1’=“ ; *)	(‘ join … admin… ; *)	(admin ; admin123)
Error, b12 unknown. *Time 19:25*	MYSQL error at index 12	*Login* *Password* admin admin123	Hi admin. *Time 08:13*
(c14 ; boom)	(% or “= ; *)	(‘ join … bidule …; *)	(bidule ; bidule123)
Error, c14 unknown. *Time 13:46*	MYSQL error at index 78	*Login* *Password* bidule bidule123	Hi bidule. *Time 21:28*

Intuition



Login

Password

(truc ; truc123)

Hi truc.
Time 07:53

(truc ; truc123)

Invalid data

(aaa ; bbb)

Error, aaa unknown.
Time 21:23

(b12 ; machin)

Error, b12 unknown.
Time 19:25

(c14 ; boom)

Error, c14 unknown.
Time 13:46

Syntactically invalid SQL injection

(‘” ; *)

MYSQL error
at index 20

(‘ or ‘1’=“ ; *)

MYSQL error
at index 12

(% or “= ; *)

MYSQL error
at index 78

Syntactically valid SQL injection

(‘ or admin=‘1 ; *)

Hi admin.
Time 11:23

(‘ join … admin… ; *)

Login	*Password*
admin	admin123

(‘ join … bidule …; *)

Login	*Password*
bidule	bidule123

?

Valid data

(truc ; truc123)

Hi truc.
Time 07:53

(admin ; admin123)

Hi admin.
Time 08:13

(bidule ; bidule123)

Hi bidule.
Time 21:28

Proposed algorithm

- ❑ Find SQL injections that:
  - *would not* generate *error pages* (MYSQL error, authentification error)
  - *would generate* successful execution pages (successful authentication, access to another page)

- ❑ Principle
  - Build a reference model of error pages returned by the web site
    - Generate (randomly) requests with invalid authentication data AND Requests with syntactically invalid SQL injections
    - Analyse responses ⇨ reference model for *error pages*
  - Generate syntactically valid SQL injections
  - Identify the responses that are distant from the reference

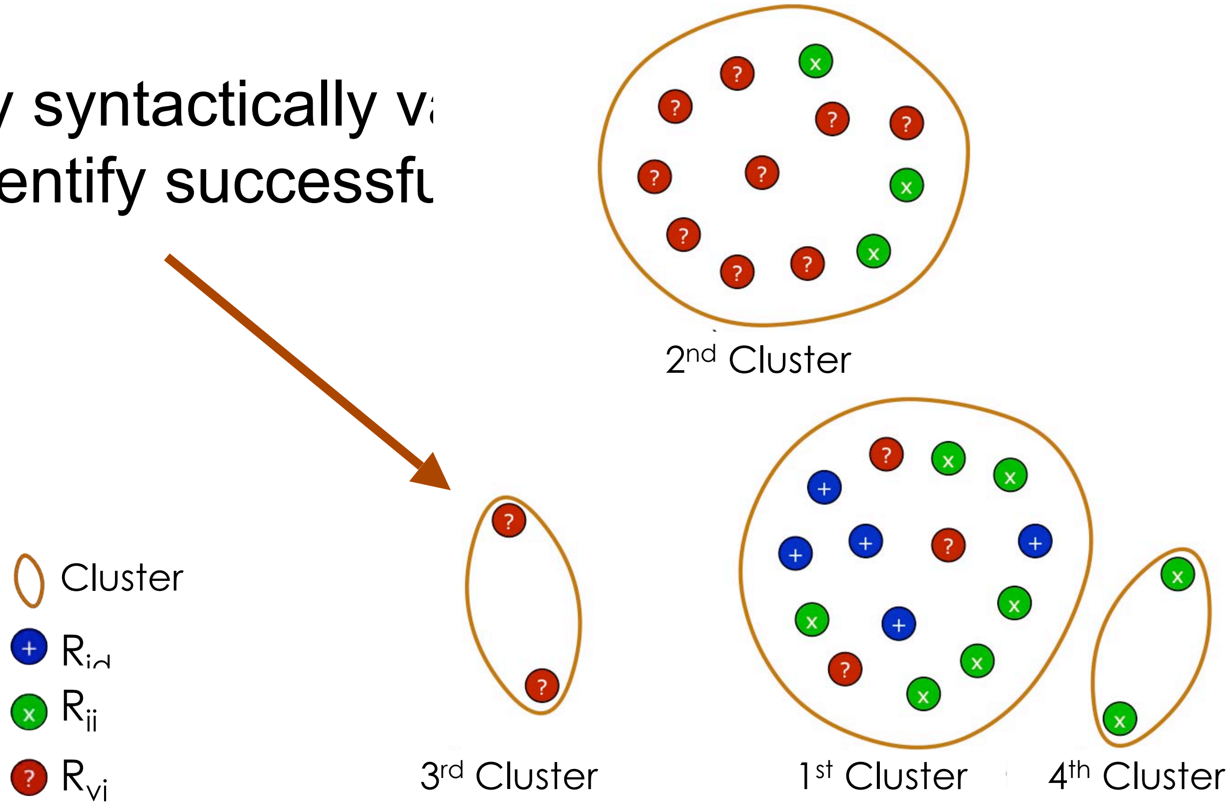
    ⇨ These responses are likely to be associated to valid SQL injections

- ❑ Hierarchical clustering algorithm
  - Paired similarity analysis of response pages based on *diff*

Illustration

- ❑ Requests sent to the server through an injection point
  - – R_{id} : requests with invalid data
  - – R_{ii} : syntactically invalid SQL injections
  - – R_{vi} : syntactically valid SQL injections -> generate execution pages

- ❑ Clusters with only syntactically va SQL injections identify successfu injections



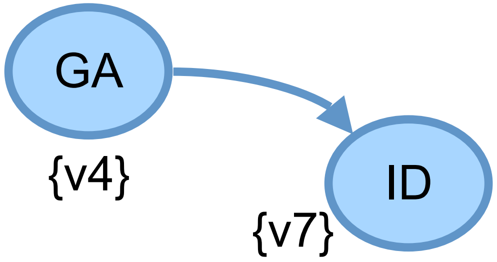
Preliminary results

		Vulnerability scanners			
		Skipfish	W3AF	Wapiti	Our tool
Vulnerabilities	v1 phpBB3	✗	✗	✓	✓
	v2 SecurePages	✗	✗	✓	✓
	v3 HardwareStore	✓	✓	✓	✓
	v4 HardwareStore	✓	✓	✗	✓
	v5 HardwareStore	✓	✗	✗	✓
	v6 HardwareStore	✗	✗	✗	✓
	v7 HardwareStore	—	—	—	✓
	v8 Kereval	✓	✓	✗	✓
	v9 DVWA	✓	✓	—	✓
Number of detections		*5*	*4*	*3*	*9*

✗: non detected ✓ : detected — : non tested injection point

❑ Observation

- error pattern matching approach not efficient enough
- Skipfish approach needs to be improved



Perspectives

❑ Algorithm

- a more through experiment is need to validate the preliminary conclusions including also other vulnerability scanners
- Investigate applicability to other types of vulnerabilities

❑ Goal-driven attack strategy

- Formalisation and implementation of the approach to generate automatically attack scenario and different possible instantiations of these scenarios

❑ Experimental assessment of the two intrusion detection techniques developed in the project