The Twelfth International Conference on Networks - ICN 2013
January 27 - February 1, 2013 - Seville, Spain

A Validation Model of Data Input for Web Services

Rafael B. Brinhosa, Carla M. Westphall, Carlos B. Westphall, Daniel R. dos Santos, Fábio Grezele

Post Graduation Program in Computer Science
Federal University of Santa Catarina
{ brinhosa,carlamw,westphal,danielrs,fgrezele }@inf.ufsc.br

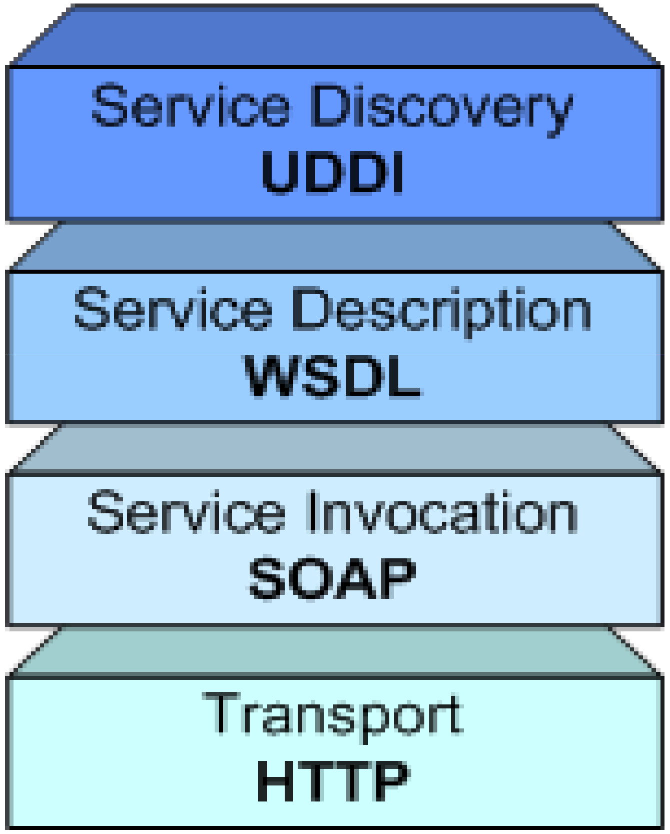
Content at a Glance

Introduction

- SOA is based on web services but there are security related concerns
- The lack of proper input validation is a major cause of data breaches and Web application attacks
- Application attacks: SQL injection and cross-site scripting (XSS)

Web Services Input Validation Model: an XML schema, an XML specification and a module for performing input validation according to the schema

Introduction – Web Services



Find Web services which meet certain requirements
(Universal Description, Discovery and Integration)

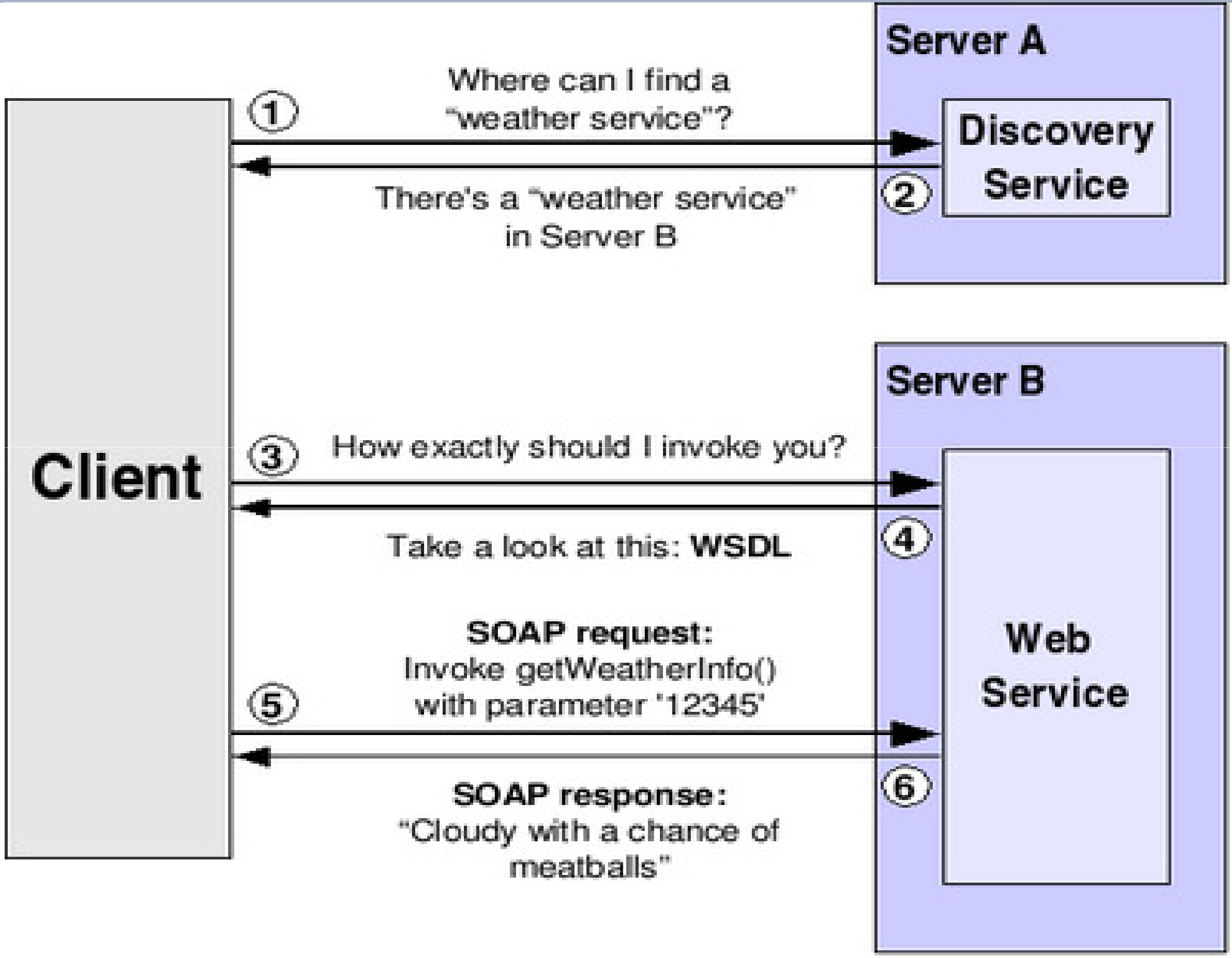
Services describe their own properties and methods
(Web Services Description Language)

Format of requests(client) and responses (server)
(Simple Object Access Protocol)

Message transfer protocol

(Hypertext Transfer Protocol)

Introduction – Web Services



Server A
Discovery Service
Client
1 Where can I find a "weather service"?
2 There's a "weather service" in Server B
Server B
3 How exactly should I invoke you?
4 Take a look at this: **WSDL**
Web Service
5 **SOAP request:** Invoke getWeatherInfo() with parameter '12345'
6 **SOAP response:** "Cloudy with a chance of meatballs"

Figure available from: http://gdp.globus.org/gt4-tutorial

Related Work

- **Lack of input validation is a major cause of Web application attacks**
  - SANS, 2011: The Top Cyber Security Risks
  - OWASP 2010: OWASP top 10 Web application security risks
  - [T. Scholte, D. Balzarotti, E. Kirda, 2011] - “Quo vadis? A study of the evolution of input validation vulnerabilities in Web applications”
- **Few specific mechanisms for Web Services**
- **[N. A. Nordbotten, 2009] [L. Sun and Y. Li, 2008] use XML security technologies (encryption)**

Related Work

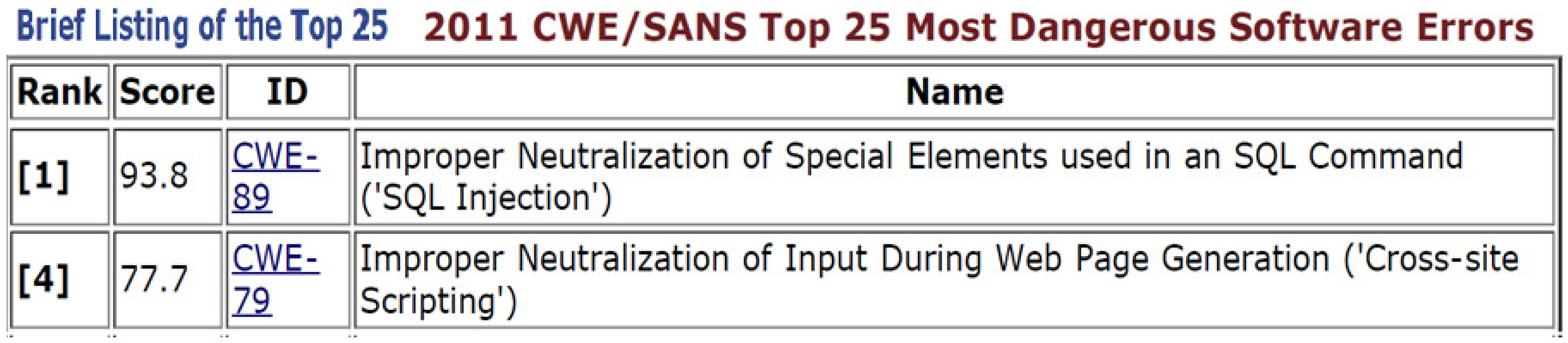
- WS-Security Wrapper: is an intermediate between the Web service and the client; is an adapter program that converts plain XML exchanges to and from SOAP with WS-Security (XML signature and encryption). It does not include features such as validation of predefined data entries
- [J. Lin and J. Chen, 2009]
  - Collects web pages (crawler), identify weak points and test them
  - insert the input validation (meta-programs) on the server side, acting as a web application firewall
  - many false positives with blacklist approach

Related Work

- IAPF (Integrated Application and Protocol Framework) [N. Sidharth and J. Liu, 2007]:
  - Protection in UDDI, WSDL, SOAP (WS-Sec)
- XML firewall:
  - [A. Blyth, 2009] is concerned with validation of the structure of XML content but not the content itself
  - [Y. Loh, W. Yau, C. Wong, and W. Ho, 2006] mentions protection against SQL injection through an XML schema and a precompiled blacklist of SQL commands, an approach which tends to produce many false positives

Security Issues in Web Services

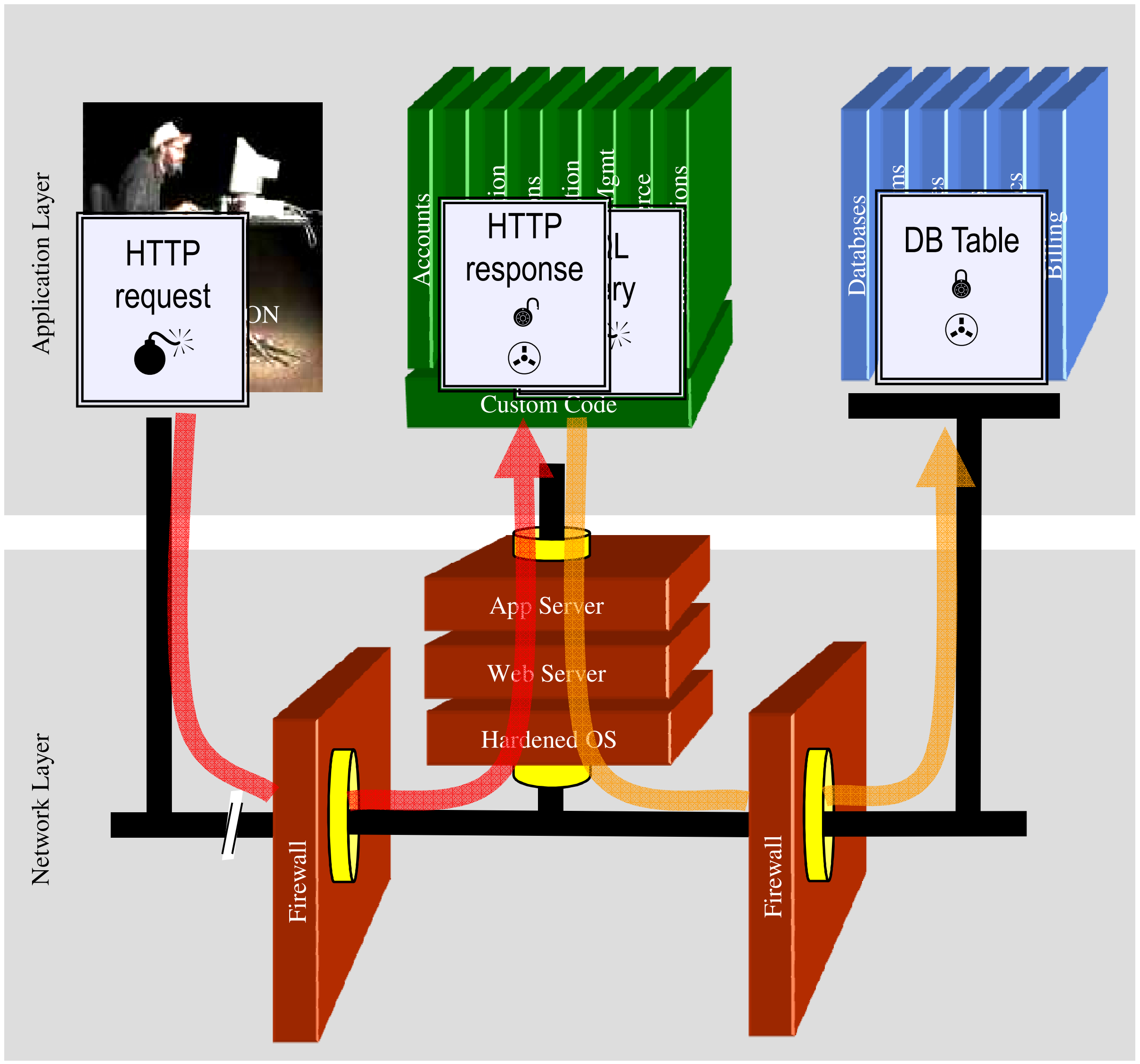
- WSDL scanning: to reveal sensitive information about invocation patterns, underlying technology implementations
- Serious and important data manipulation attacks: SQL injection and XSS
- Normal firewalls, antivirus and using WS-Security standards are not able to protect web services against SQL injection and XSS attacks

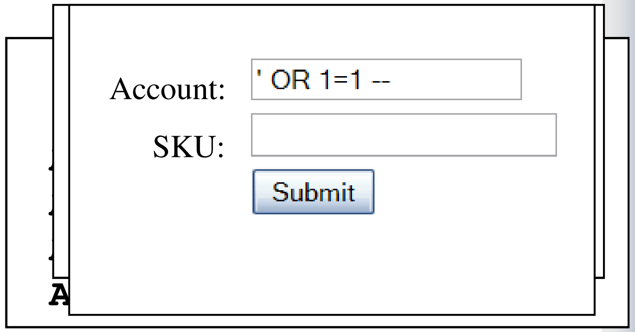


Brief Listing of the Top 25 2011 CWE/SANS Top 25 Most Dangerous Software Errors

Rank	Score	ID	Name
[1]	93.8	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
[4]	77.7	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

SQL Injection – Illustrated





Account:	' OR 1=1 --
SKU:	

1. Application presents a form to the attacker
2. Attacker sends an attack in the form data
3. Application forwards attack to the database in a SQL query
4. Database runs query containing attack and sends encrypted results back to application
5. Application decrypts data as normal and sends results to the user

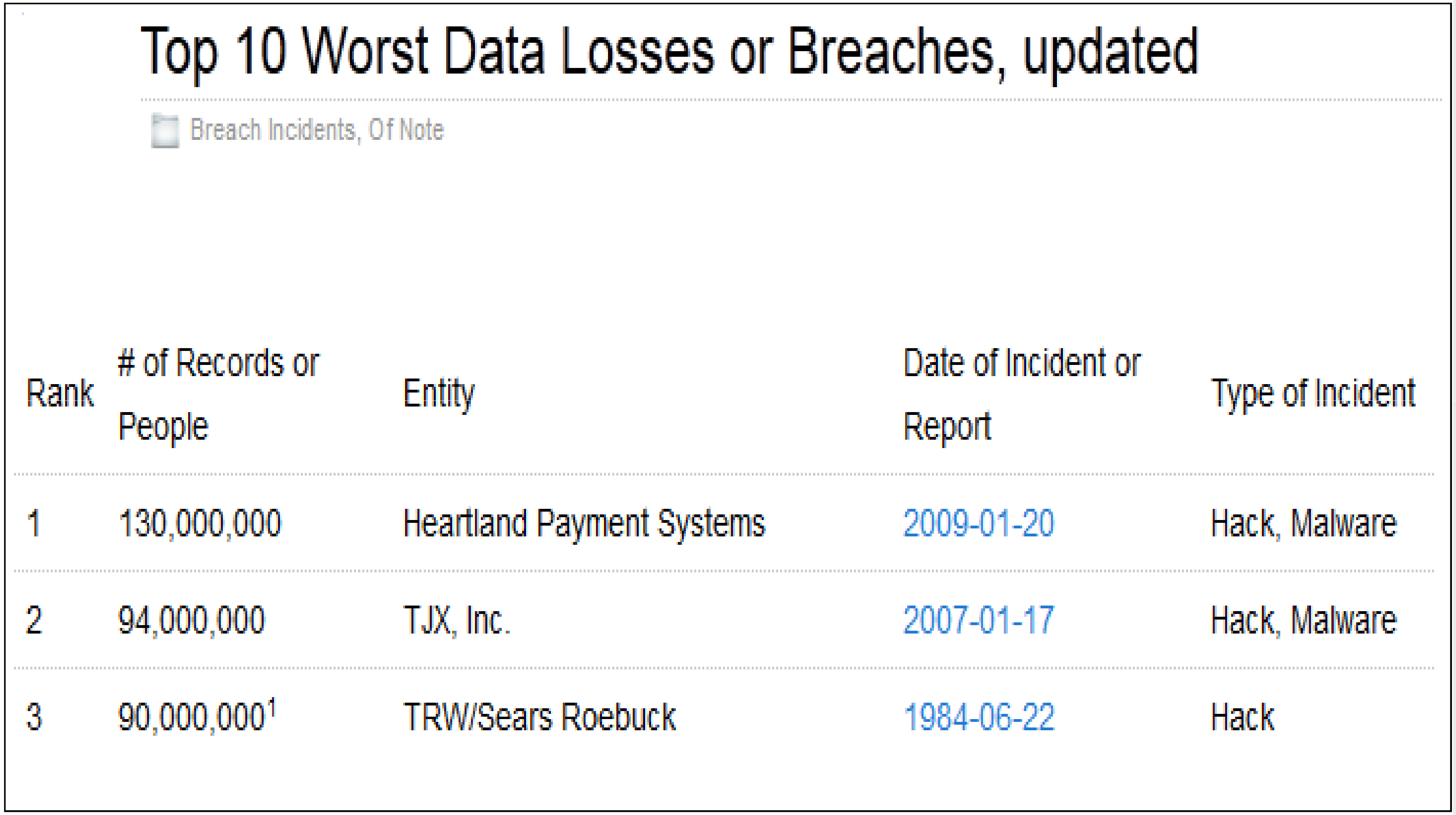
Security Issues in Web Services

- SQL Injection: by sending SOAP requests with properly handled parameters, for example, “'"1=1 -” as a parameter for a particular service

```
ERROR: The query was not accomplished. Description:
1064 - You have an error in your SQL syntax; check the
manual that corresponds to your MySQL server version
for the right syntax to use near '1=1'' at line 1
```

```
Line 11: Incorrect syntax near '')) or  ItemId in (select
ItemId from dbo.GetItemParents('4''. Unclosed
quotation mark before the character string ')) ) ) > 0 ‘
```

Heartland Payment Systems



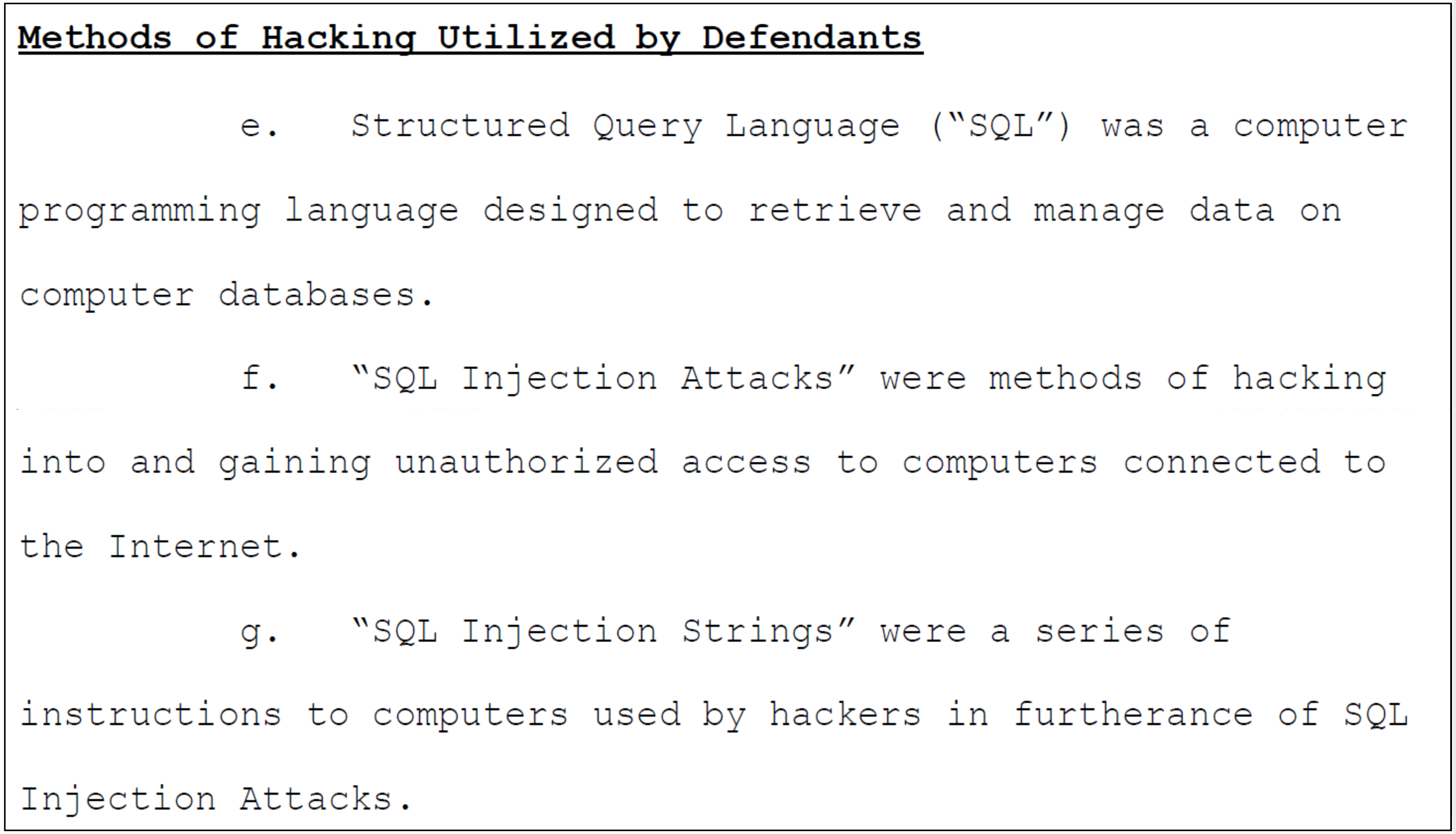
Top 10 Worst Data Losses or Breaches, updated

Breach Incidents, Of Note

Rank	# of Records or People	Entity	Date of Incident or Report	Type of Incident
1	130,000,000	Heartland Payment Systems	2009-01-20	Hack, Malware
2	94,000,000	TJX, Inc.	2007-01-17	Hack, Malware
3	90,000,000[1]	TRW/Sears Roebuck	1984-06-22	Hack

Available from: http://www.databreaches.net/?p=7691

Heartland Payment Systems



Methods of Hacking Utilized by Defendants

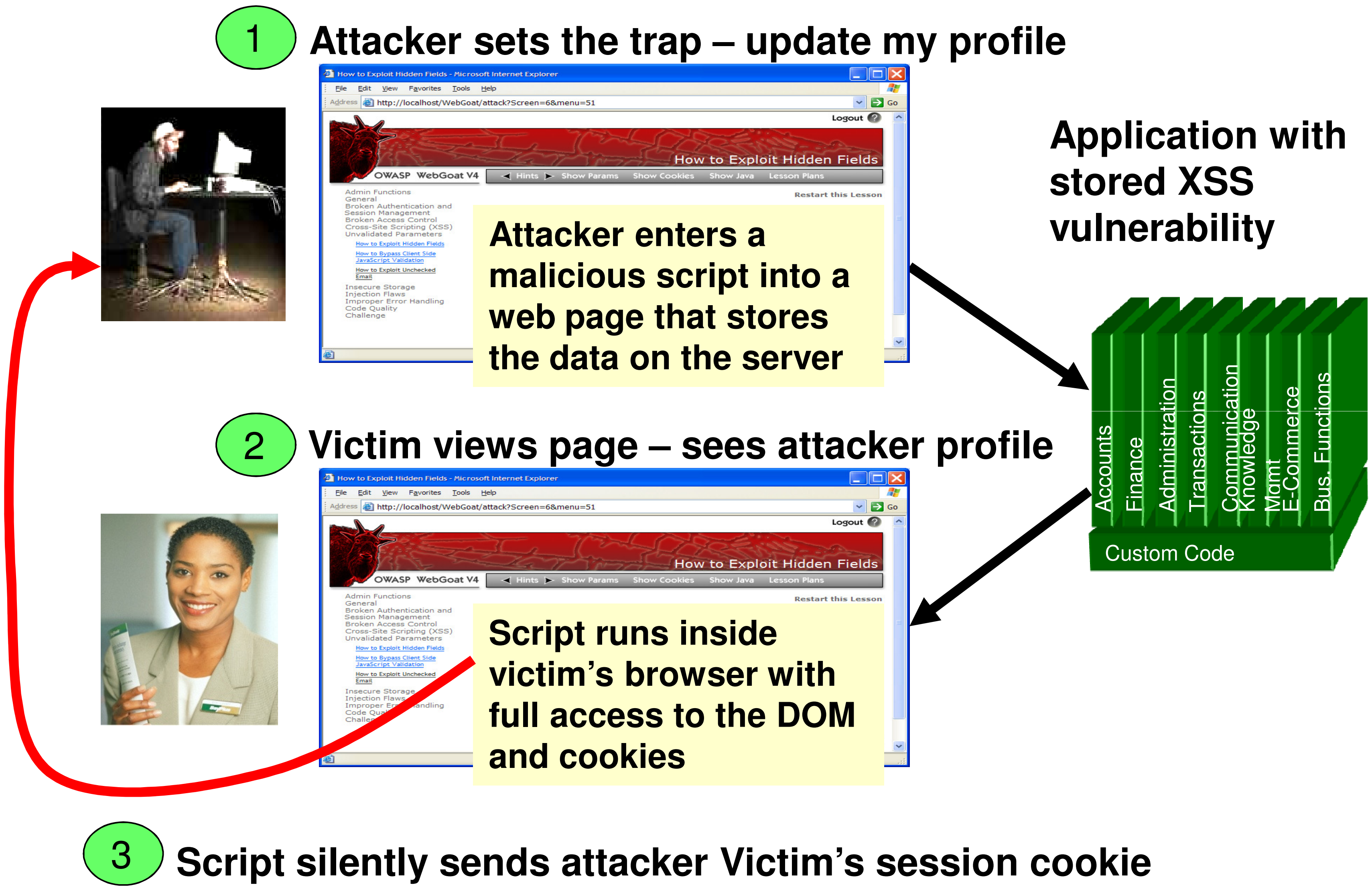
e. Structured Query Language ("SQL") was a computer programming language designed to retrieve and manage data on computer databases.

f. "SQL Injection Attacks" were methods of hacking into and gaining unauthorized access to computers connected to the Internet.

g. "SQL Injection Strings" were a series of instructions to computers used by hackers in furtherance of SQL Injection Attacks.

Available from: http://www.wired.com/images_blogs/threatlevel/2009/08/gonzalez.pdf

Cross-Site Scripting Illustrated



1 Attacker sets the trap – update my profile

Attacker enters a malicious script into a web page that stores the data on the server

Application with stored XSS vulnerability

2 Victim views page – sees attacker profile

Script runs inside victim's browser with full access to the DOM and cookies

3 Script silently sends attacker Victim's session cookie

How to Exploit Hidden Fields - Microsoft Internet Explorer
http://localhost/WebGoat/attack?Screen=6&menu=51
How to Exploit Hidden Fields
OWASP WebGoat V4
Accounts
Finance
Administration
Transactions
Communication
Knowledge
Mgmt
E-Commerce
Bus. Functions
Custom Code

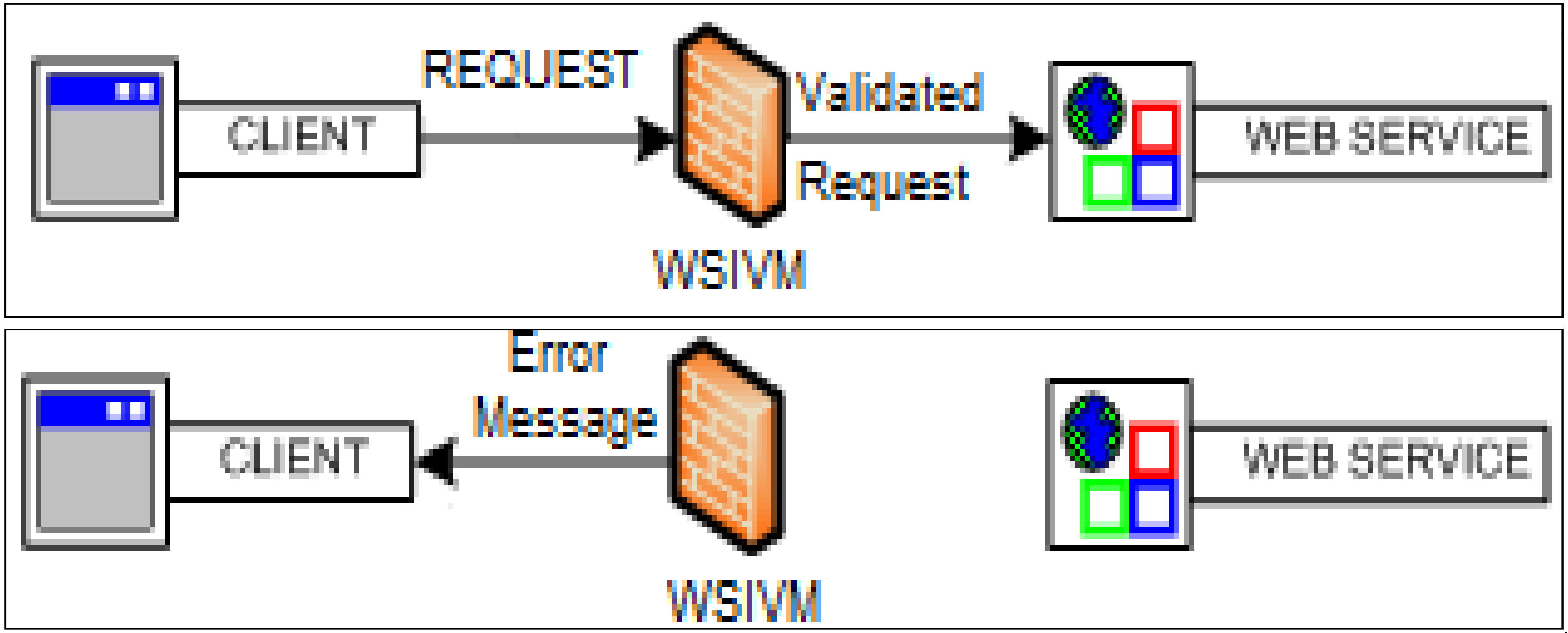
Available from: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

Security Issues in Web Services

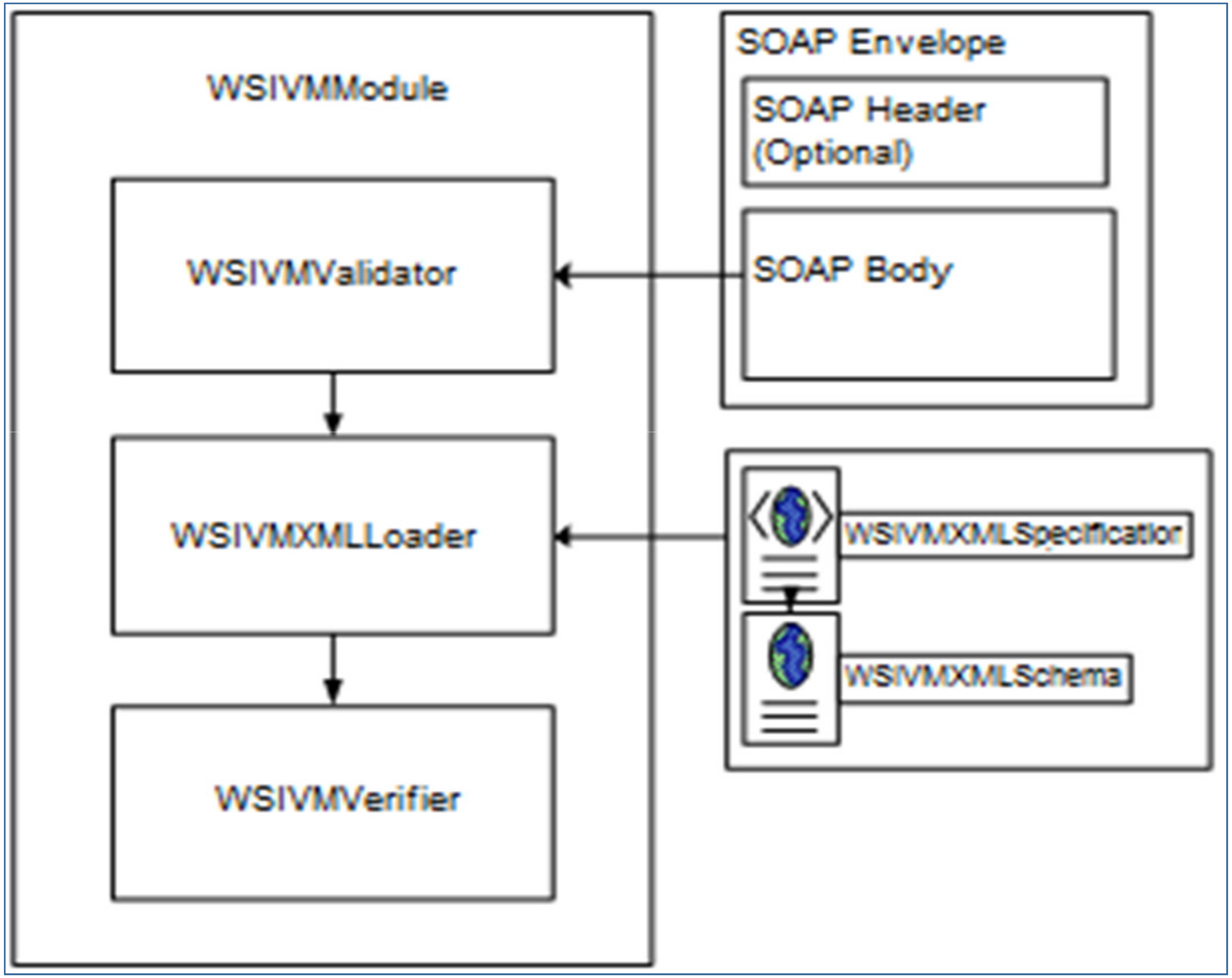
- XSS: by presenting unvalidated data directly to the user, Web services can be attacked. Using, for example, the command *document.write(xmlhttp.responseText)*, if the answer to this AJAX (Asynchronous JavaScript and XML) call made to a Web service contains HTML and JavaScript data, these data will be interpreted and executed, posing a risk to the user

WSIVM – Web Services Input Validation Model

- Validate input data to provide security for Web Services. Controls on the lexical and syntactic aspects, type checking
- Best way to avoid attacks: whitelist input validation (sanitization - change input into an acceptable format)

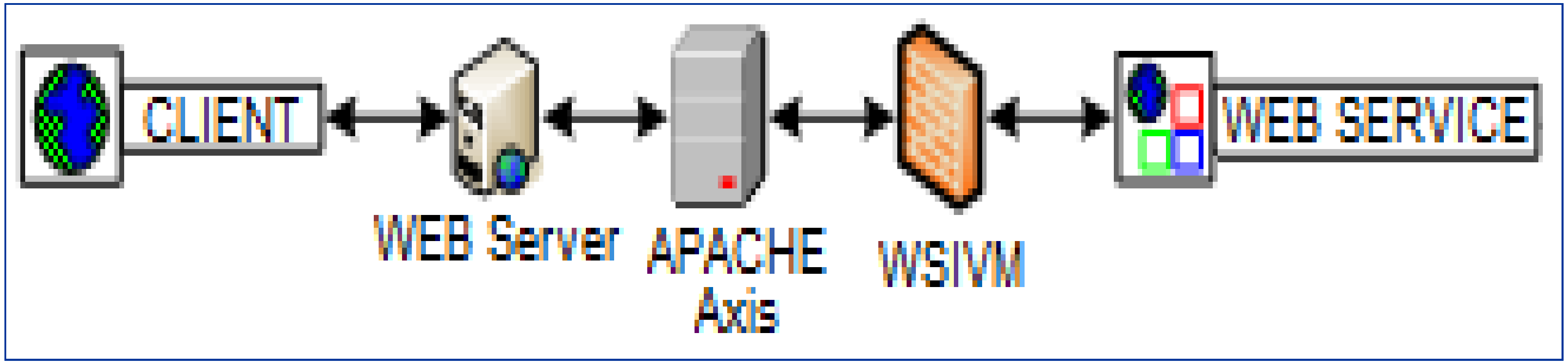


WSIVM – Web Services Input Validation Model



WSIVM – Development

- Apache Tomcat Web server
- Apache Axis2 framework for SOAP messages
- To implement the validation module for Apache Axis2 the Rampart module was used
- It was chosen to intercept the message in the phase *PreDispatch*
- WSIVMXMLSchema, WSIVMXMLSpecification, and WSIVM Rampart module



WSIVM – WSIVMXMLSpecification

OperationName: the name of the operation
SanitizeOperation: defines whether the parameters of this operation can be reformulated if necessary for the removal of characters that are not accepted
ParamName: the name of the parameter or field
Allowed: an allowed field type, which is valid (text, html, html+java-script, email, number, and all)
Length: specifies the exact size of the field
Maxsize: specifies the maximum field size
Minsize: specifies the minimum field size
Nillable: determines whether or not it is possible that the field is null (true or false)
regEx: allows a regular expression to be specified for validation

Case Study

- Case Study: Client Application + Web Service
- UniversityManager web service:
  - searchStudent receives a registration number (ID) and returns the student record containing a *String* with his or her information
  - registerStudent operation receives the information, which must not contain HTML or Javascript code, and registers it on the MySQL database. In the database a student's table is created: ID, name, age, email, comment, site, and birthday

WSIVMXMLSpecification

WSIVMXMLSpecification–UniversityManager

```
<?xml version="1.0" encoding="UTF-8"?>
<valid_inputs_specification mlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
WebServiceID=“UniversityManager"
xsi:noNamespaceSchemaLocation="valid_inputs_specification.xsd">
<operation name="registerStudent ">
<input name="name" type="String" min="5" max="20" accept="text" sanitize="true"/>
<input name="age" type="Integer" min="0" max="150" accept="number"
sanitize="true"/>
<input name="email" type="String" min="0" max="200" accept="email" sanitize="true"/>
<input name="comment" type="String" min="0" max="200" accept="text"
sanitize="true"/>
<input name="site" type="String" min="0" max="300" accept="url" sanitize="true"/>
<input name="data" type="String" min="0" max="200" accept="regex" regexpattern=
"(\\d{4})-(\\d{2})-(\\d{2})" sanitize="true"/>
</operation>
<operation name="searchStudent ">
<input name="id" type="Integer" min="0" max="10000" accept="number"
sanitize="true"/>
</operation>
</valid_inputs_specification>
```

Case Study - Results

- Experiments:
  - With and without WSIVM
  - 150 users
  - The test runs for 300 seconds
  - soapUi: direct calls to the web service

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:univ="http://university.wsivm.example">
<soap:Header/><soap:Body>
  <univ:registerStudent>
  <univ:name>John</univ: name >
  <univ:age>12</univ: age >
  <univ:email>john@hsj.com</univ:email>
  <univ:comment>Passed</univ: comment >
<univ:site>http://www.univ.com</univ:site>
<univ:birthday>1980-09-12</univ: birthday >
</univ: registerStudent >
</soap:Body></soap:Envelope>
```

Case Study - Results

Comparison	Min. Time	Max. Time	Avg. Time	Transferred Bytes	B/s	Insertions In DB
Without WSIVM	35 ms	27848 ms	2494,85 ms	1974195 B	6506 B/s	10078
With WSIVM	64 ms	13346 ms	4541,24 ms	1236330 B	4012 B/s	5134
Total	83 %	- 52 %	82 %	- 37 %	- 38 %	- 49 %

Conclusions

- Reusable and independent mechanism for data entry validation, regardless of the implementation of the web service
- Based on the white list approach (reduction in false positives)
  - More reliable than the blacklist. If a blacklist is created based on the current version of HTML, in the case of new versions, this list may no longer be considered valid
  - The number of false positives or false negatives will depend on the WSIVM XML Specification defined

Conclusions

  - The framework provides the specification to be customized according to the Web Service requirements and needs
- Prevention of data injection attacks in Web services and the waste of server processing with invalid messages
- Reduces the possibility of denial of service using content of messages
- Negative impact on the performance of the developed Web service but reduces the possibility of inserting invalid data
- Solution for legacy applications reducing development costs

Future Work

- Optimization of the implementation to improve the performance of the proposed model
- Development of a semi-automatic generator of security specifications from WSDL
- Verification of SOAP messages and paths in XPath format
- Use of artificial intelligence or an anomaly detection system
- Making a feedback loop filter validation of invalid entries

Some References

1. OWASP Top Ten - https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project
2. T. Scholte, D. Balzarotti, and E. Kirda, “Quo vadis? A study of the evolution of input validation vulnerabilities in Web applications,” in *Proc. Int. Conference on Financial Cryptography and Data Security '11*, St. Lucia, 2011.
3. N. A. Nordbotten, “XML and Web services security standards,” *Communications Surveys & Tutorials, IEEE*, vol. 11, no. 3, pp. 4–21, 2009.
4. L. Sun and Y. Li, “XML and Web services security,” in *Proc. 12th Int. Conf. Computer Supported Cooperative Work in Design*, *CSCWD 2008*, April 16–18, pp. 765–770.
5. N. Sidharth and J. Liu, “A framework for enhancing Web services security,” in *Proc. 31st Ann. Int. Computer Software and Applications Conf., 2007, COMPSAC 2007*, Jul. 24–27, vol. 1, pp. 23–30.
6. WS-Security Wrapper - http://wsswrapper.sourceforge.net/
7. J. Lin and J. Chen, “An automated mechanism for secure input handling,” *Journal of Computers*, vol. 4, no. 9, pp. 837–844, 2009.
8. N. Sidharth and J. Liu, “A framework for enhancing Web services security,” in *Proc. 31st Ann. Int. COMPSAC 2007*, Jul. 24–27, vol. 1, pp. 23–30.
9. A. Blyth, “An architecture for an XML enabled firewall,” *International Journal of Network Security*, vol. 8, no. 1, pp. 31–36, 2009, ISSN 1816–3548.
10. Y. Loh, W. Yau, C. Wong, and W. Ho, “Design and implementation of an XML firewall,” in *Proc. 2006 Int. Conf. Computational Intelligence and Security*, Nov. 3–6, vol. 2, pp. 1147–1150.
11. CWE/SANS TOP 25 Most Dangerous Software Errors - http://cwe.mitre.org/top25/archive/2011/2011_cwe_sans_top25.pdf
12. R. Wu and M. Hisada, “SOA Web Security and Applications”, *Technology*, vol. 9, nº. 2, p. 163-177, 2010.
13. T. Scholte, W. Robertson, D. Balzarotti, E. Kirda, "Preventing Input Validation Vulnerabilities in Web Applications through Automated Type Analysis," in Proc. *2012 IEEE 36th Annual COMPSAC 2012* , pp.233-243, July 2012.

Thank you!

Rafael B. Brinhosa, Carla M. Westphall, Carlos B. Westphall,
Daniel R. dos Santos, Fábio Grezele

{ brinhosa,carlamw,westphal,danielrs,fgrezele }@inf.ufsc.br