TAIC PART 2010

Linguistic Security Testing for Textual Protocols

Authors

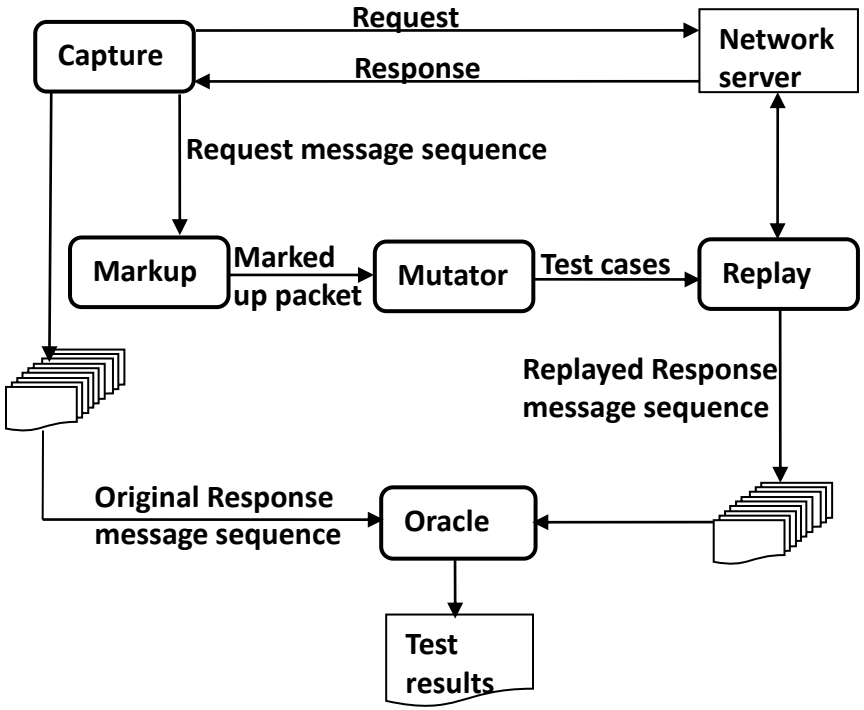
Ben Kam, Tom Dean

Queen's University, Canada

Goals and History

1. Protocol Tester
    - Protocol Tester Group – Queen's University and RMC
    - Testing – binary-based network communication protocols (OSPF)
    - Protocols represented by context free grammar
    - Using a test planner to insert a different set of XML markup tags into the captured message sequences to guide the mutation
    - The set of mutation tags was hard coded
2. Extend our previous versions
    - Testing – text-based network communication protocols
    - Using protocol description file to insert XML markup tags
    - Some of mutation tags are generated automatically by using a program to analyze the grammar
    - Handling complex mutations
    - Protocol independent (HTTP, FTP, iCal …)

Syntax-based Security Testing (SST) framework



Protocol Specification and Markup

```
% partial HTTP grammar
define program
    [request-message]
end define
define request-message
    [request-line][repeat headers_message]
    [CRLF][opt message_body]
end define
define request-line
    [method][space][request-uri][space]
    [http-version][CRLF]
end define
```

The partial low level HTTP protocol specification

```
Include "http.grm"
redefine entity_header
        ...
    |  [SOAPAction]
end redefine
define SOAPAction
    [soap_uri][soap_message]
end define
define soap_message
    [xml_declaration][open_soap_envelope]
    [soap_header] [soap_body][close_soap_envelope]
end define
```

The middle level XML SOAP protocol specification

Example of HTTP request packet

```
POST /return.asp HTTP/1.1
Host: 192.168.1.105
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.1.11) Gecko/20071220 BonEcho/2.0.0.11
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
……..
```

Protocol Specification and Markup

```
% partial HTTP grammar
define program
    [request-message]
end define
define request-message
    [request-line][repeat headers_message]
    [CRLF][opt message_body]
end define
define request-line
    [method][space][request-uri][space]
    [http-version][CRLF]
end define
```

The partial low level HTTP protocol specification

```
define request_line
    <enumeratedLiteral>[method]</enumeratedLiteral>
    [space][request_uri][space] [http_version] [CRLF]
end define
```

Nested Markup tag example

```
<enumeratedLiteral><caseSensitive>[method]</caseSensitive></enumeratedLiteral>
```

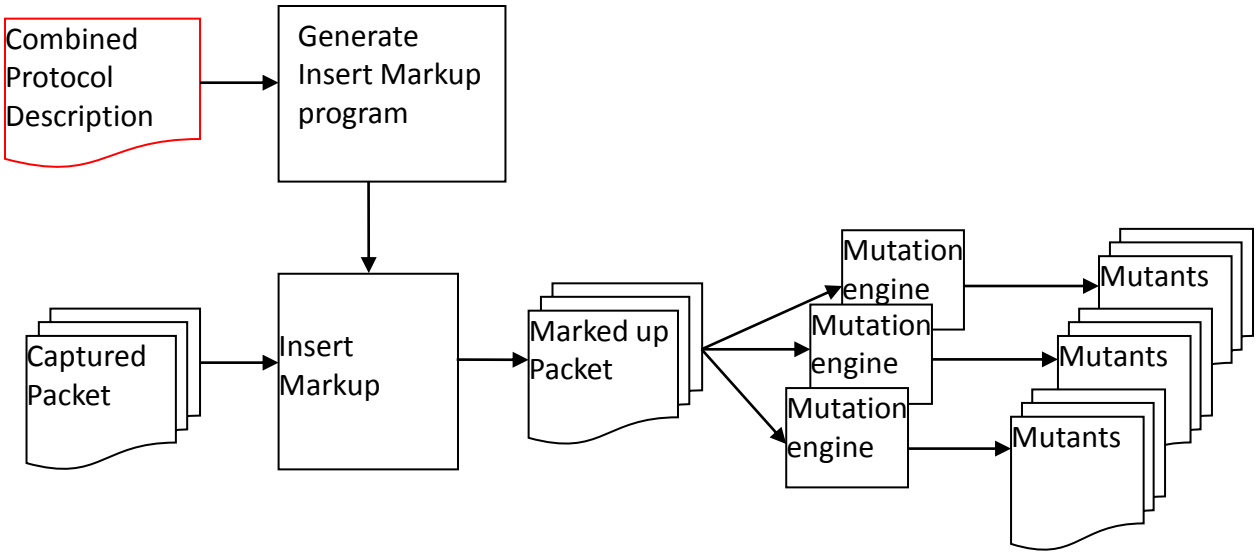
Relation tag example

```
define Content_Length
   'Content-Length : [space] <length id="%" root="request_message" role="length">
   [number]</length>
end define
define message_body
   <length id="%" root="request_message" role="value“>[repeat token_or_key]</length>
end define
```

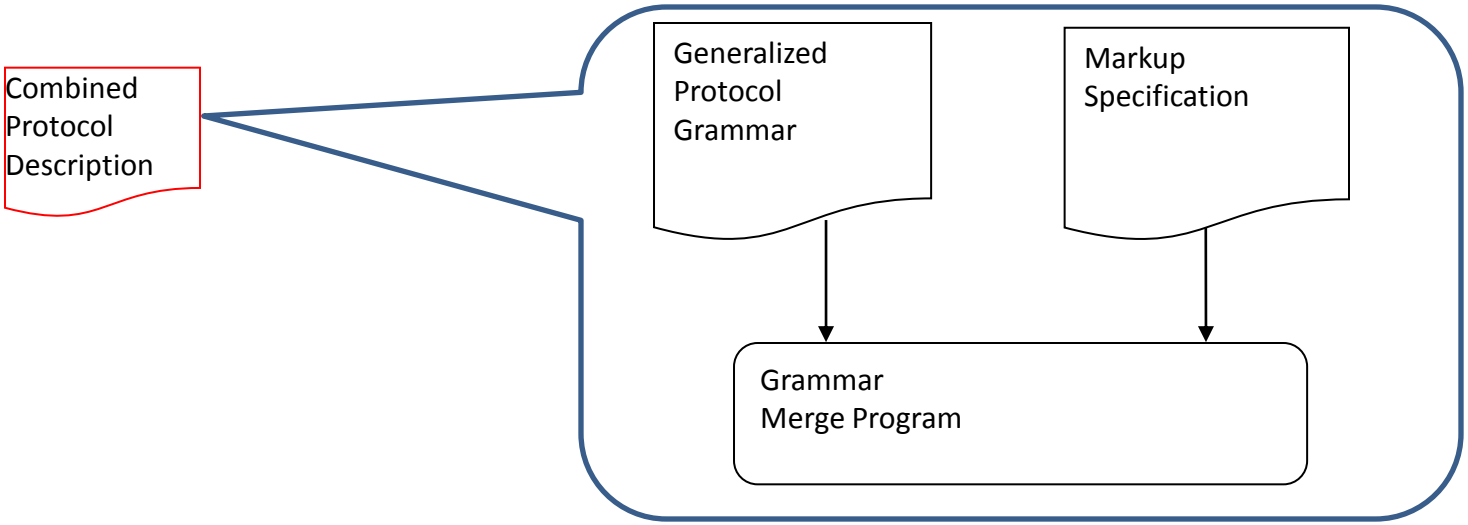
Categorization of markup tags

Types	Tags	Purpose
Syntactic	enumeratedLiteral	Change to another terminal provided from grammar to alter the original semantics
Lexical	caseSensitive	Change the terminal letters from upper case to lower case or vice
	charSpecific	Change the terminal character
	dateSpecific	Change the terminal date format
	syntaxSpecific	Alter the terminal characters
	valueLimitation	Change the terminal value to common boundary values
	stringSpecific	Replace a string values with common alternate strings
Relational	length	Indicates that the number marked by the length role gives the number of characters in the value role.
Custom	jpeg	The content identified by the tag is an embedded jpeg image (e.g. file upload).

Markup and mutate process



Using Agile parsing techniques



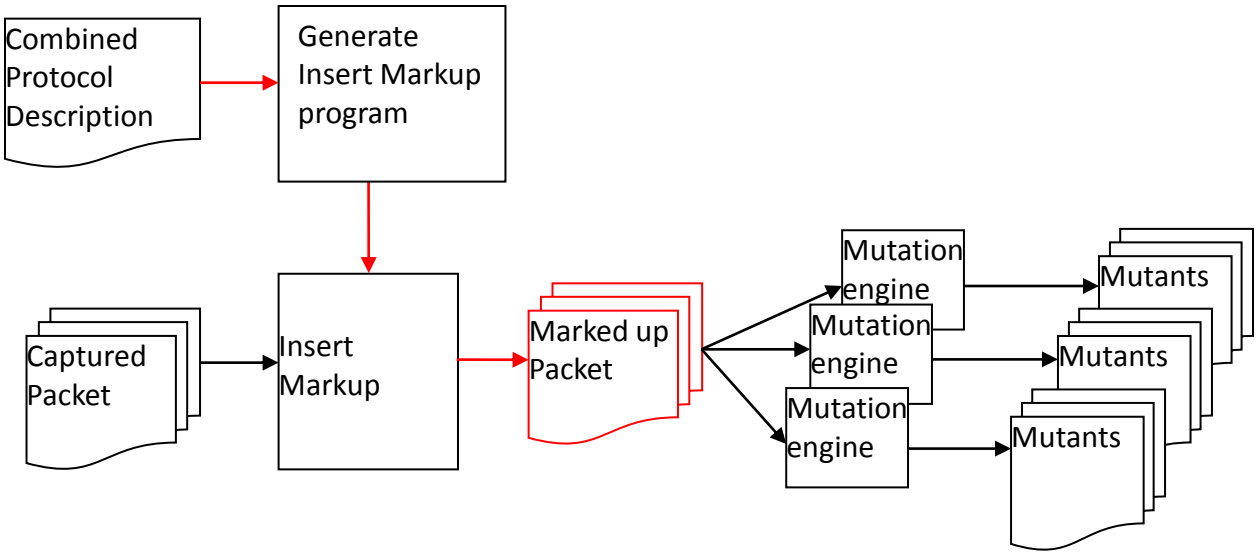
Generalized Protocol Grammar

```
define http_version
    HTTP / [number]
end define
```

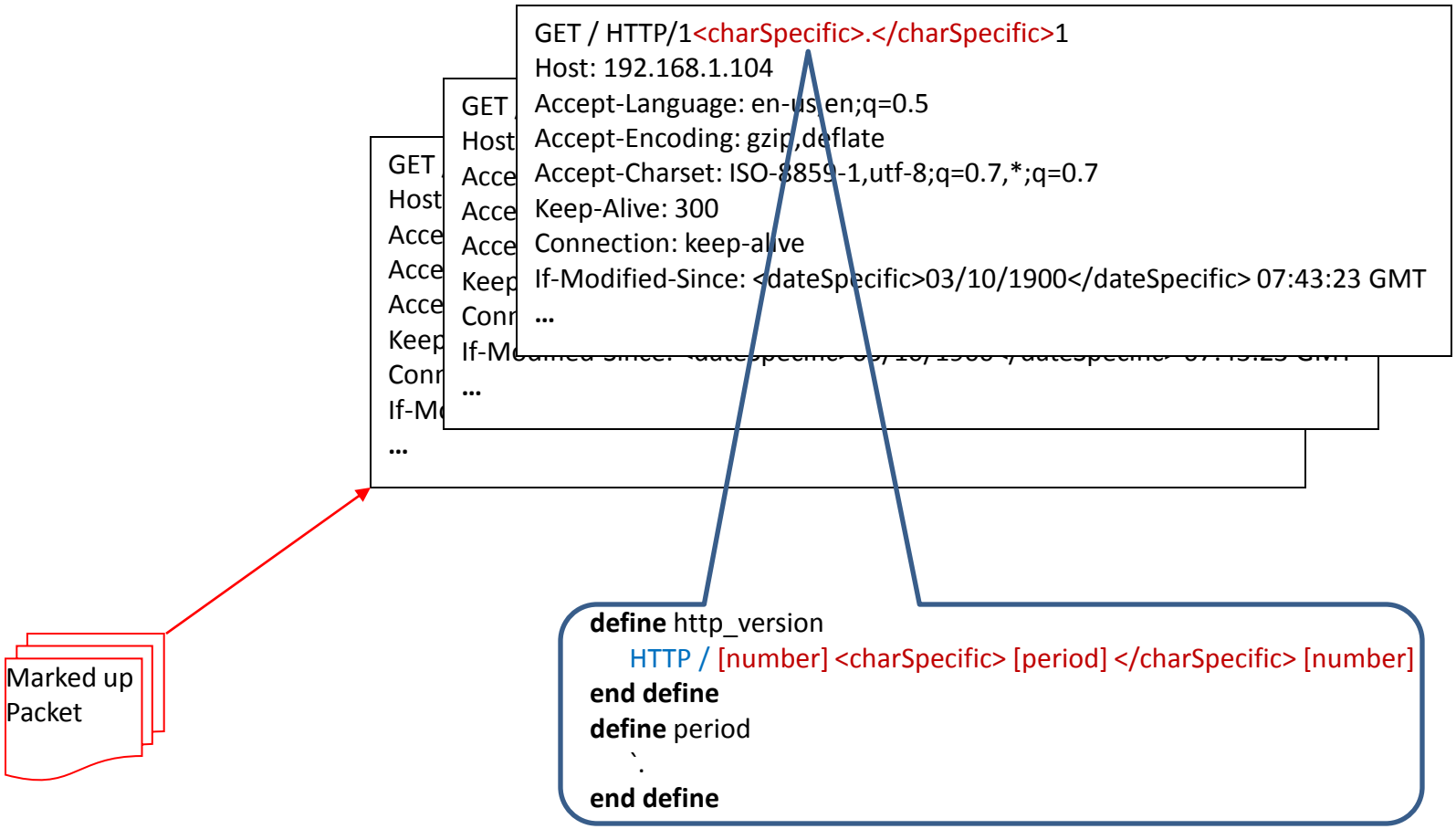
Markup Specification

```
define http_version
    HTTP / [number] <charSpecific> [period] </charSpecific> [number]
end define
define period
    `.
end define
```

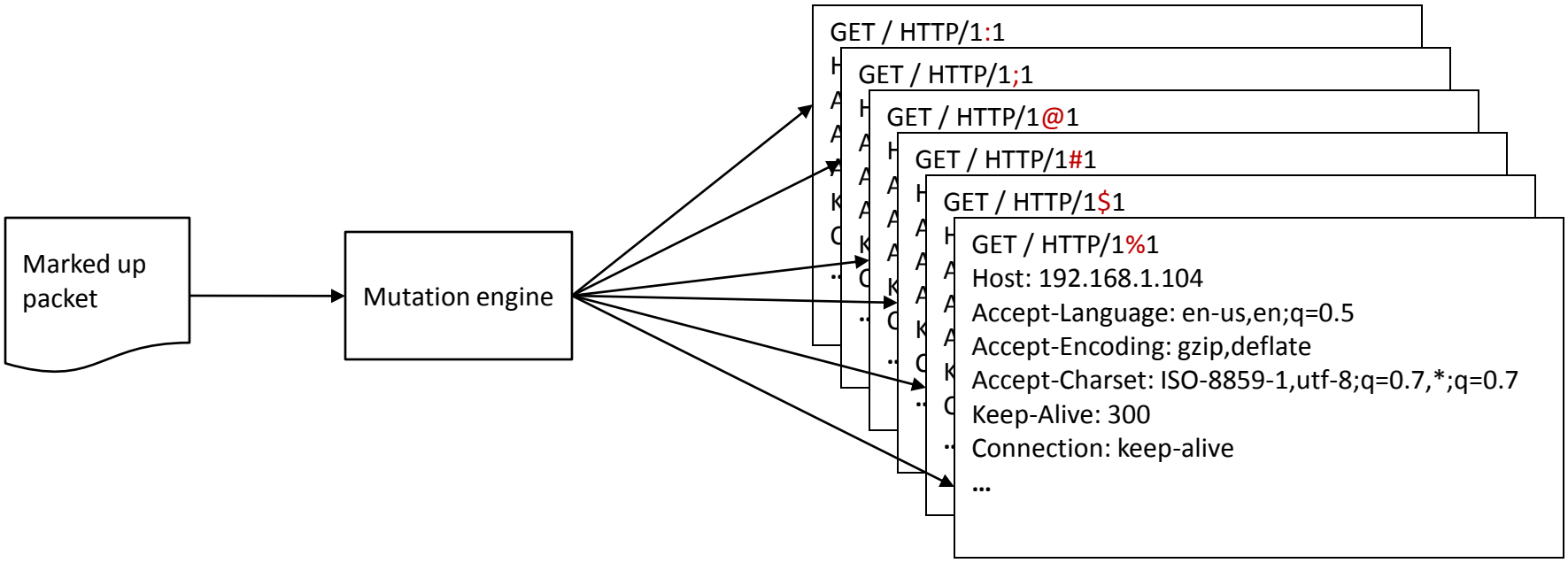
Markup and mutate process



Marked up packets



Mutants examples



Experiments

1. Toy web applications

- to validate the functionality of the framework
- Interesting result – IIS accepted the undefined request method message
  Apache2 rejected the undefined request method message (method POST was changed to post)

2. kOrganizer

- mutated iCal files – caseSensitive, charSpecific, dateSpecific, syntaxSpecific, valueLimitation, and stringSpecific
- Xmacroplay instructs the kOrganizer to open and close the mutated iCal files
- A total of 1026 test cases generated from a single iCalendar file
- kOrganizer crashed by a 16Mb string (SIGSEGV - a segmentation violation)
- The total running time was 244188 seconds (67.83 hours)

SST vs other black box security testing tools

Methodology	Protocol independent	Vulnerability reveal								Reference
		SQL injection	Cross-site scripting	Session Fixation	Directory traversal	Command Injection	Buffer overflows	Denial of service	Bypass restriction	
User Session	No								√	[1,2,6,7,8]
WAVES	No	√	√							[3]
Bypass	No	√	√						√	[5]
SecuBat	No	√	√							[4]
SST	Yes	√	√	√	√	√	√	√	√	-

Conclusion and Future Work

The contribution of this research

- Creation of a light weight testing framework usable for most text-based communication protocols
- Extension mechanism for easily adding new markup tags and mutators
- Integration with external mutators for embedded binary data
- We have demonstrated the framework with attacks on HTTP and iCalendar application
- Protocol independent testing framework
- Extended SST to handle higher level application protocols e.g. shopping cart protocol

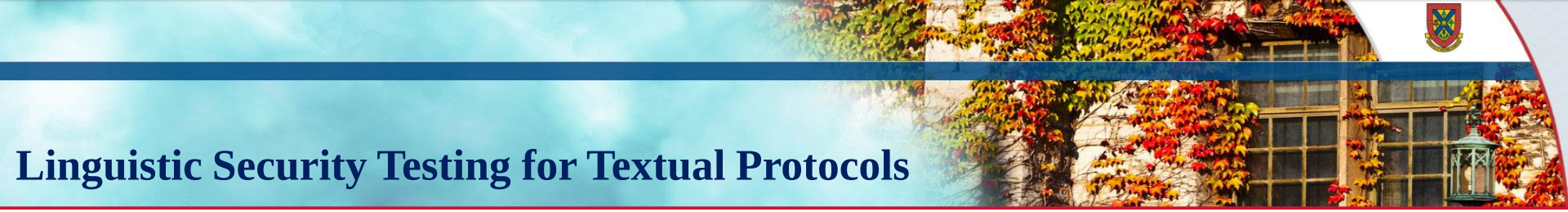
Conclusion and Future Work

Exploring the markup tags

- to create more markup tags to break the syntax and semantic of the grammar
- to break particular programming language constraints e.g. C language
- to generate multiple markup tags per packet
- Modified all the tags that can have attributes sent to the mutation engines – only relation tags can carry attributes in current version

Fully Automated Testing

- Automated insert markup tags by analyzing the protocol grammar e.g. partially done by an enumeratedLiteral markup tag



Thank you

&

Questions?

References

1. Elbaum S., Karre S., Rothermel G., “Improving web application testing with user session data”, Proceedings of the 25th International Conference on Software Engineering, pp. 49 – 59, 2003.
2. Elbaum S., Rothermel G., Karre S., Fisher II M., “Leveraging user-session data to support Web application testing”, IEEE Transactions on Software Engineering, Volume 31, Issue 3, pp. 187 – 202, 2005.
3. Huang Y. W., Huang S. K., Lin T. P., Tsai C. H., “Web application security assessment by fault injection and behaviour monitoring”, Proceedings of the 12th International Conference on World Wild Web, pp. 148 – 159, 2003.
4. Kals S., Kirda E., Kruegel C., Jovanovic N., “SecuBat: a web vulnerability scanner”, Proceedings of the 15th international conference on World Wide Web, pp. 247 – 256, 2006.
5. Offutt J., Wu Y., Du X., Huang H., “Web Application Bypass Testing”, Proceedings of the 28th Annual International on Computer Software and Applications Conference, pp. 106 – 109 vol., 2, 2004.
6. Sprenkle S., Gibson E., Sampath S., Pollock L., “Automated Relay and Failure Detection for Web Applications”, Proceedings of the 20th IEEE/ACM International Conference on Automated software engineering, pp.253 – 262, 2005.
7. Sprenkle S., Gibson E., Sampath S., Pollock L., “A case study of automatically creating test suites from web application field data”, Proceedings of the 2006 workshop on Testing, analysis, and verification of web services and applications, pp.1 – 9, 2006.
8. Sprenkle S., Sampath S., Gibson E., Pollock L., A. Souter, “An empirical comparison of test suite reduction techniques for user-session-based testing of Web applications”, Proceedings of the 21st IEEE International Conference on Software Maintenance, pp.587 – 596, 2005.