

# XML Injection Mutation for Web Services Vulnerability Testing Based on SOAP Messages

Bimal Varghese   Simi Stephen

Department of Computer Science and Engineering  
Federal Institute of Science and Technology  
Mookkannoor

March 5, 2015

- 1 Introduction
  - Web Service
  - Vulnerability Testing of Web Service
- 2 Base Paper
  - Mutation Operators
  - Worst Input Mutation
- 3 Problem Definition
  - XML Injection
- 4 Current Work
  - Testing Tool
  - GUI Screens
- 5 Pending works

# Introduction

## 1 Introduction

- Web Service
- Vulnerability Testing of Web Service

## 2 Base Paper

- Mutation Operators
- Worst Input Mutation

## 3 Problem Definition

- XML Injection

## 4 Current Work

- Testing Tool
- GUI Screens

## 5 Pending works

- Common way of implementing SOA.
- Widely used in the Internet.
- Quality and Reliability of Web Service must be heavily tested.

# Introduction

## 1 Introduction

- Web Service
- Vulnerability Testing of Web Service

## 2 Base Paper

- Mutation Operators
- Worst Input Mutation

## 3 Problem Definition

- XML Injection

## 4 Current Work

- Testing Tool
- GUI Screens

## 5 Pending works

# Vulnerabilities in WS

- Multiple Components in WS

# Vulnerabilities in WS

- Multiple Components in WS
  - WSDL

# Vulnerabilities in WS

- Multiple Components in WS
  - WSDL
  - SOAP



# Vulnerabilities in WS

- Multiple Components in WS
  - WSDL
  - SOAP
  - XML

# Vulnerabilities in WS

- Multiple Components in WS
  - WSDL
  - SOAP
  - XML
  - UDDI

# Vulnerabilities in WS

- Multiple Components in WS
  - WSDL
  - SOAP
  - XML
  - UDDI

# Vulnerabilities in WS

- Multiple Components in WS
  - WSDL
  - SOAP
  - XML
  - UDDI
- Vulnerability refers to flaws in the service that threaten the security of the computer system

# Vulnerabilities in WS

- Multiple Components in WS
  - WSDL
  - SOAP
  - XML
  - UDDI
- Vulnerability refers to flaws in the service that threaten the security of the computer system
- Some types of Web service vulnerability faults might not be effectively revealed by traditional testing approaches

# Introduction

- 1 Introduction
  - Web Service
  - Vulnerability Testing of Web Service
- 2 Base Paper
  - Mutation Operators
  - Worst Input Mutation
- 3 Problem Definition
  - XML Injection
- 4 Current Work
  - Testing Tool
  - GUI Screens
- 5 Pending works

# Mutation Operators

- Testing based on SOAP message.
- An extended Regular Tree Grammar(RTG) model
  - $\langle E; N; DT; P; A; n_s \rangle$
  - $E$  :finite set of elements.
  - $N$  :finite set of non Terminals.
  - $DT$  :finite set of Data types.
  - $P$  :finite set of production rules.
  - $A$  :2 tuple  $\langle n, type \rangle$ 
    - ①  $n$  :number of parameters.
    - ②  $type$ : parameter type.
- A mutation operator is  $r = f(n_1, n_2, \dots, n_i)$  where  $f$  is a function,  $i \geq 1$ , each  $n_1, n_2, \dots, n_i \in N$  and has an arbitrary data type and  $r$  outputs mutated  $n_1, n_2, \dots, n_i$  with the same data type.

# Introduction

- 1 Introduction
  - Web Service
  - Vulnerability Testing of Web Service
- 2 Base Paper
  - Mutation Operators
  - Worst Input Mutation
- 3 Problem Definition
  - XML Injection
- 4 Current Work
  - Testing Tool
  - GUI Screens
- 5 Pending works



# Mutation Operators

- Regular Mutation
  - Small modification of the legitimate input

# Mutation Operators

- Regular Mutation
  - Small modification of the legitimate input
- Worst Input Mutation
  - Use Farthest neighbor sequence from the legitimate input.

- Many program faults result in failures manifesting in contiguous areas of the input domain.
- if previously executed test cases have not revealed a failure, new test cases should be as far from the already executed non-failure test cases as possible
- For Testing, a Web Service Vulnerability Testing System (WSVTS) is created

### Advantages

- Have the greatest possible test coverage.
- Typical representation for triggering faults.
- Low redundancy.

# Introduction

- 1 Introduction
  - Web Service
  - Vulnerability Testing of Web Service
- 2 Base Paper
  - Mutation Operators
  - Worst Input Mutation
- 3 Problem Definition
  - XML Injection
- 4 Current Work
  - Testing Tool
  - GUI Screens
- 5 Pending works

# XML Injection

- XML is the default way by which web service communicates.
- Also used in storing data with dynamic tag values.
- Tampering an XML will compromise the security of the web service.
- One of the most serious xml vulnerability is the XML injection.
- Can compromise data and even the security of the entire system itself.

# Introduction

- 1 Introduction
  - Web Service
  - Vulnerability Testing of Web Service
- 2 Base Paper
  - Mutation Operators
  - Worst Input Mutation
- 3 Problem Definition
  - XML Injection
- 4 Current Work
  - Testing Tool
  - GUI Screens
- 5 Pending works

# Work Completed

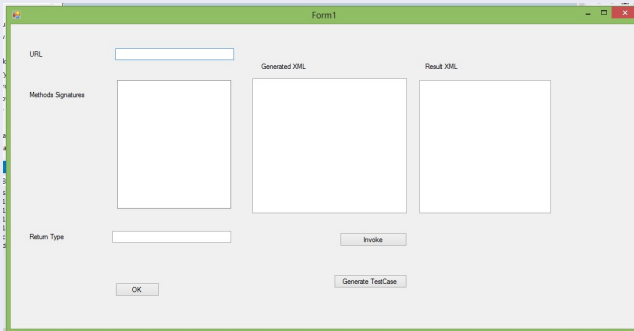
- Web service Testing Tool is Created.
- Tool is able to identify the available Method names, their parameters, types, and their return values from the wsdl file.
- It is able to generate SOAP message for available Web methods dynamically.
- It can also invoke the web method and display the result in xml format.
- Can inject *xml injection vulnerability* code into SOAP message.

# Introduction

- 1 Introduction
  - Web Service
  - Vulnerability Testing of Web Service
- 2 Base Paper
  - Mutation Operators
  - Worst Input Mutation
- 3 Problem Definition
  - XML Injection
- 4 Current Work
  - Testing Tool
  - GUI Screens
- 5 Pending works



## Main Form



The screenshot shows a Windows-style application window titled "Form1". The window has a light gray background and a green title bar. It contains several input fields and buttons:

- URL:** A text input field at the top left.
- Methods Signatures:** A large text area below the URL field.
- Return Type:** A text input field at the bottom left.
- Generated XML:** A large text area in the center-right.
- Result XML:** A large text area on the far right.
- Buttons:** There are four buttons: "Invoke" (located below the Return Type field), "Generate TestCase" (located below the Generated XML field), "OK" (located at the bottom left), and a button with a question mark icon (located at the bottom right).

Figure: Main Form

## Main Form

The screenshot shows a Windows-style application window titled "Form1". It contains several sections for configuring and executing a SOAP test:

- URL:** A text box containing "http://localhost:5728/Service1.asmx?WS".
- Methods Signatures:** A tree view showing a list of methods: "Hello World", "Login", "AddResult", "Test", and "GetBookDetails".
- Return Type:** An empty text box.
- Generated XML:** A text area displaying the SOAP request XML. It includes headers for XML version, encoding, SOAP envelope, and namespaces, followed by a body containing an "AddResult" operation with two string arguments.
- Result XML:** A text area displaying the SOAP response XML. It includes headers for XML version, encoding, SOAP envelope, and namespaces, followed by a body containing an "AddResultResponse" operation with two string arguments.
- Buttons:** There are three buttons: "Invoke" (to execute the test), "Generate TestCase" (to generate a test case), and "OK" (to close the dialog).

Figure: Main Form

## Test Cases



Figure: Main Form

## Results

Form1

URL:

Methods Signatures:

- [-] HelloWorld
- [+] Login
- [+] AddResult
- [+] Test
- [+] GetBookDetails

Return Type:

Generated XML:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
  xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body><GetBookDetails
    xmlns="http://localhost:6728/Service1.smx?WS"
    </GetBookDetails>
  </soap:Body>
</soap:Envelope>
```

Result XML:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
  xmlns:xs="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:soap="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://localhost:6728/Service1.smx?WS"
  xmlns:tns1="http://schemas.xmlsoap.org/soap/envelope/">
  <tns:AddResultResponse
    xmlns:tns="http://localhost:6728/Service1.smx?WS"
    </tns:AddResultResponse>
  </soap:Body>
</soap:Envelope>
```

Number of time Invoked : 5  
Errors obtained : 1

OK

Generate Test Case

Figure: Result Form

# To be completed

- Proper analysis of output.
- Proper formatting of XML for display.
- Display output for test case invocation.
- Fix display issues is showing the results.
- Clean the code.

# Demo

Thank You