A Security Description Assistance in Web Services

Pham Phuoc Hung, Aziz Nasridinov, Jeongyong Byun
Department of Computer Engineering, Dongguk University
e-mail: hung205a2@yahoo.com, aziz_nasridinov@yahoo.com, byunjy@dongguk.ac.kr

웹서비스에서 보안 설정 지원

밤복흥, 아지즈 나스리디노프, 변정용
동국대학교 컴퓨터공학

Abstract

When SOAP message in Web Services has sensitive and important data, it is necessary to protect the message from XML rewriting attacks. These attacks create a foundation for typical faults in SOAP message and make it vulnerable to use in Web Service environment. Currently, Web Services middleware offers limited functions to detect these faults and possibly fix them. In this paper, we propose a Security Description Assistance which identifies and fixes typical faults in SOAP messages. Our system adapts simulation-based approach, which allows system to self-optimize its performance in different conditions and thus improve the reliability of Web Services.

1. Introduction

Web Services are broadly used by many companies to do a business over the Internet. And, they are required to have an appropriate level of security including confidentiality, integrity, reliability, and availability [1]. It is widely recognized that one of the barriers preventing widespread adoption of this technology is a lack of products that support non-functional features of applications, such as security and reliability [2]. Because Web Services use SOAP messages to communicate with each other, the reliable performance of Web services is strongly dependant on the mechanisms of this communication protocol.

All communication protocols are subject to damage and there are several external factors. XML rewriting attacks [3] are among these factors and widely used by attackers. It is commonly known that these attacks on SOAP messages create a foundation for typical faults in SOAP messages and make it vulnerable to use in Web Service environment. Unfortunately, Web Services middleware offers limited functions to detect these faults and possibly fix them. Therefore, one of the challenging problems is to protect SOAP message and make sure it satisfies security requirements of both requester and provider.

In this paper, we tackle these problems by proposing a Security Description Assistance which identifies and fixes typical faults in SOAP messages. Our system adapts simulation-based approach, which allows system to self-optimize its performance in different conditions and improve the reliability of Web Services.

The paper is organized in following way. In Section 2 Related Study are discussed. Section 3 illustrates the Motivating scenario and Section 4 presents System Design. Section 5 discusses our System Implementation. The paper will end with conclusion and future work.

2. Related Studies

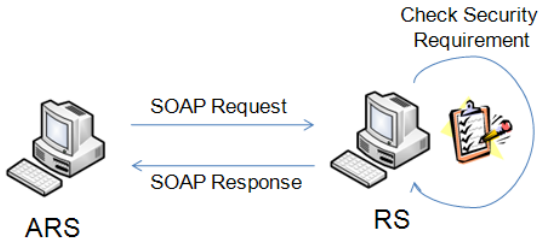
Similar classification and functional relationships were explored in various discovery working groups.

This paper mainly reflects future directions of our previous researches [4, 5]. In previous researches, we developed a technique called Bit-Stream. It works based on the importance of SOAP elements in order to automatically detect the vulnerabilities and risks, while offering advice for higher security. Current research complements previous ones by adding more functions such as fixing and optimizing performance of the system in case if SOAP request has faults.

In [6], authors proposed a system with a set of extensible recovery policies to specify how to handle and recover from faults in Web Service composition. However, their system does not recognize if the SOAP message should be fixed or not. In [7], researchers describe a framework that supports the development of self-optimizing autonomic systems for Web Services architecture. But the disadvantage of this approach is that it only deals with the performance optimizing without mentioning how it fixes faults in a system. In contrast, our system is able to solve problems in above mentioned approaches [6, 7]. It can detect the faults in SOAP message, analyze them, and make a decision if the message should be fixed or not. By that we ensure optimization of the system performance.

3. Motivating Scenario

Consider a scenario where Auto Repair Shop (ARS) connects to Retailer Shop (RS) to buy a car part, shown in Figure 1. In this scenario, we assume that ARS has already passed the RS's authentication control and, ARS and RS have established reliable connection. To order a car part, ARS have to go through following steps:



(Figure 1) Motivating scenario

1. ARS sends SOAP message to RS to buy a car part.
2. After receiving SOAP message from ARS, RS systems will check if SOAP satisfies security requirements.

   - If SOAP message satisfies security requirements, system can process the request and reply result to ARS.
   - If SOAP message does not satisfy security requirement, our system fixes fault elements in SOAP message.

4. Security Description Assistance

In this chapter, we will describe the Security Description Assistance system architecture along with detailed system sequence chart and proposed algorithm.

4.1 System Architecture

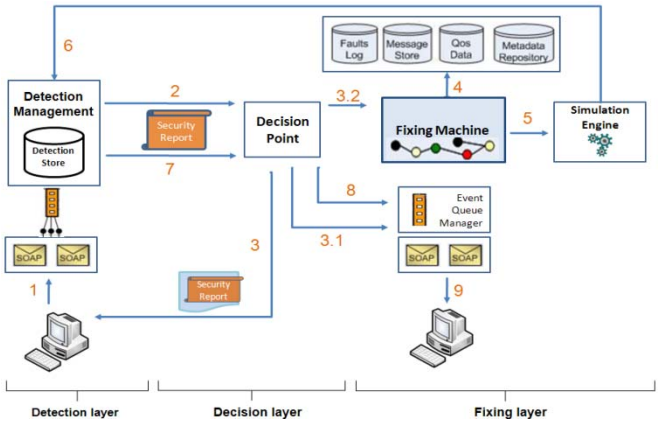
In figure 2 the general design of the proposed system is illustrated. The Security Description Assistance system architecture is structured in following three layers:

Detection layer: The layer where SOAP request is received from Web services client, Detection Management will use Bit-Stream technique to analyze, detect faults and produce Security Report.

Decision layer: The layer where according to the Security Report, Decision Point will make a decision if the SOAP request should be fixed or not. Decision layer also chooses the optimal SOAP message among fixed ones.

Fixing layer: The layer where SOAP request is being fixed. We used simulation engine to generate alternative SOAP messages.

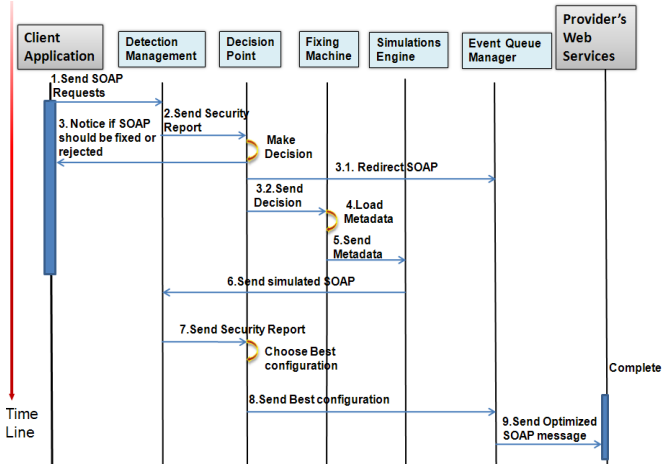
The steps in system architecture will be detailed in system sequence chart in next subsection.



(Figure 2) Security Description Assistance system design

4.2 System Sequence Chart

Security Description Assistance system architecture is described in details through system sequence chart in Figure 3. Web service client sends message to Web services provider to get some information.



(Figure 3) Security Description Assistance system sequence chart

(1) Client request is accepted in Detection Management.

(2) Detection Management analyzes SOAP message and passes security report to Decision Point.

(3) After receiving security report from Detection Management, Decision Point first notifies customer about decision.

(3.1) If SOAP message does not need to be fixed then the system will redirect SOAP message to Event Queue Manager.

(3.2) If SOAP should be fixed then the system sends it to Fixing Machine.

(4) In case if SOAP message should be fixed, Fixing Machine receives decision from Decision Point and loads a set of metadata from database. Metadata has corresponding instructions to fix SOAP fault elements.

(5) Loaded metadata transmitted to Simulation Engine and Simulation Engine generates SOAP messages according to provider's requirements.

(6) Newly generated SOAP messages are sent to Detection Management.

(7) Detection Management receives the SOAP messages, analyzes them and sends the security reports about newly generated SOAP messages to Decision Point.

(8) Based on security reports Decision Point chooses the best result and sends it to the Event Queue Manager.

(9) Event Queue Manager sends the optimized SOAP message to provider.

After finishing above steps, the original SOAP message has been optimized and satisfies the provider's security requirement. Security Description Assistance system has identified and fixed typical faults in SOAP messages. Thus reliability of Web Services is improved.

4.3 Algorithm Design

In this chapter, we explain main algorithms used in our systems for detecting and optimization.

First, we describe a function to create the Security Report.

Algorithm 1: Creating Security Report Phase

Input:
Rs = Soap Request required by customer
Function creatingSecurityReport (*Rs*)
{
Read Rs
Load Policy *Pl* from database
XmlDomTreeRs = Parse(*Rs*)
XmlDomTreePI = Parse(*Pl*)
Report = Content Handler finds Matching (XmlDomTreeRs, XmlDomTreePI)
Return Report
}

Next, we need a function to optimize SOAP message. In this function, we read metadata from database and using these metadata, system makes simulation. Security report is made based on simulations. And the best values are chosen according to Security Report.

Algorithm 2: Optimize SOAP Message Phase

Input:
Rs = Soap Request
Re = Security Report
Function OptimizeMessage(*Rs,Re*)
{
Load Metadata Repository *Meta*
Read *Rs*, Read *Re*
Var *TempRs*,*ResultArr[ ]* // Array of Simulation's results
Foreach(*Meta as Row*)
{//BeginSimulation
TempRs = simulatedRequest(*Rs , Row*)
ReportArr[]=creatingSecurityReport(*TempRs*)
} //EndSimulation
Var *bestConfigValueArray[]* = GetBestValue of *ReportArr*
If(count(*bestConfigValueArray []*) == 1)
return *bestConfigValueArray[0]*
Else{
Var bestConfig = findBestConfiguration (*bestConfigValueArray*)
return bestConfig
}
}

5. System Implementation

The results of the implementation are shown as following. The system receives a SOAP request (figure 4) from client.

```
- <S:Envelope>
  - <S:Header>
    - <To>
        http://localhost:8080/CaculatorApplication/CalculatorWSService
      </To>
      <Action>http://calculator.me.org/CalculatorWS/addRequest</Action>
    - <ReplyTo>
        <Address>http://www.w3.org/2005/08/addressing/anonymous</Address>
      </ReplyTo>
      <MessageID>uuid:181ce3ef-5545-42d4-a466-c8e71860a5af</MessageID>
    - <ns2:Sequence ns7:mustUnderstand="true">
        <ns2:Identifier>uuid:c93c14a4-7395-4189-8405-335b0daf2556</ns2:Identifier>
        <ns2:MessageNumber>1</ns2:MessageNumber>
      </ns2:Sequence>
    - <ns2:AckRequested ns7:mustUnderstand="true">
        <ns2:Identifier>uuid:c93c14a4-7395-4189-8405-335b0daf2556</ns2:Identifier>
      </ns2:AckRequested>
    </S:Header>
  - <S:Body>
    - <ns2:add>
        <i>4</i>
        <j>5</j>
      </ns2:add>
    </S:Body>
  </S:Envelope>
```

(Figure 4) Screenshot of original SOAP request

After detecting that it does not include Fault element, the system generates a security report (figure 5) with Bit-Stream String of **11011** and security level of 4/5.

Report Before Fixing view SOAP

Element	Importance Level	Require	Result
S:Body	15	1	1
Action	10	1	1
Fault	9	1	0
MessageID	5	1	1
To	3	1	1

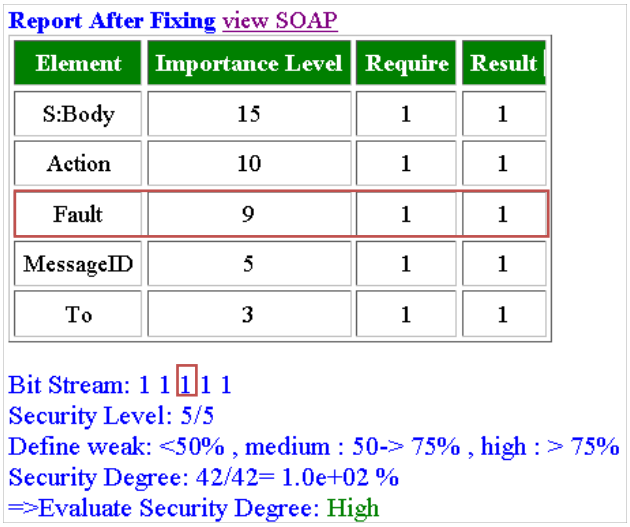
Bit Stream: 1 1 0 1 1
Security Level: 4/5
Define weak: <50% , medium : 50-> 75% , high : > 75%
Security Degree: 33/42= 79 %
=>Evaluate Security Degree: High
Advice: Insert element Fault in SOAP message

(Figure 5) Screenshot of the security report before fixing SOAP request

Through this bit stream, system can quickly recognize which element has fault. In our case it is Fault element. The advantage of Bit-Stream technique is that it can indicate importance of elements. Left bits are more important and get higher security than right bits. So system can choose which element of SOAP message has higher priority. So now, after fixing, Fault element is added to SOAP message (figure 6).

(Figure 6) Screenshot of SOAP request has already fixed

In this Bit-Stream should be called up again to analyze newly generated SOAP message. So after analyzing SOAP message, we have following bits **11111** security level (5/5) which indicates that SOAP message is securable to process (figure 7).



(Figure 7) Screenshot of the security report after fixing SOAP request

6. Conclusion

In this paper, we proposed a Security Description Assistance which identifies and fixes typical faults in SOAP messages. Our system adapted simulation-based methodology, which allows system to be able to self-optimize its performance in different conditions. After implementation of our approach, we have seen that attacked SOAP message are detected and fixed according to Web Service provider's security requirements. Our results have shown that implementation of this method was able to deliver reasonably good performance in order to achieve our goals listed in previous sections.

In future work we are planning to improve and extend our approach in a variety of areas to get a better methodology to choose the best configuration to optimize performance and thus to achieve higher reliability of Web Services.

Reference

[1] Esmiralda Moradian and Anne Håkansson, "Approach to Solving Security Problems Using Meta-Agents in Multi Agent System", KES-AMSTA 2008, pp. 122–131, 2008.

[2] Nirmal K Mukhi, Pierluigi Plebani, Ignacio SilvaLepe, Thomas Mikalsen, "Supporting Policy-driven Behaviors in Web Services: Experiences and Issues", ACM, 2004.

[3] M. A. Rahaman, A. Schaad, and M. Rits. "Towards secure soap message exchange in a soa", In Workshop on Secure Web Services, 2006.

[4] Pham Phuoc Hung, Jeongyong Byun, "An Importance-Based Advisor for Web Services Security", UCWIT, 2010.

[5] Pham Phuoc Hung, Aziz Nasridinov, Jeongyong Byun, "Importance-Based Security Level Verification in Web Services", KIPS Spring Conference, 2010.

[6] Abdelkarim Erradi, "Recovery Policies for Enhancing Web Services Reliability", IEEE International Conference on Web Services , ICWS'06, 2006.

[7] Massimiliano Rak. "Optimizing Secure Web Services with MAWeS: a Case Study", Third International Conference on Security and Privacy in Communication Networks and the Workshops, SecureComm, 2007.