XaT-SOAP: XML 기반 공격 허용 SOAP 메시지
(XaT-SOAP: XML-based Attack-Tolerant SOAP Messages)

아지즈 나스리디노브 † 박복홍 †
(Aziz Nasridinov) (Pham Phuoc Hung)

림청 † 변정용 ††
(Lin Qing) (Jeong Yong Byun)

요 약
여러 웹서비스 보안 기술들은 협력 환경에서 협력자 간에 SOAP 메시지의 안전한 교환을 목적으로 이용된다. 하지만 이러한 보안기술들을 이용할지라도 SOAP 메시지는 아직도 XML기반 공격에 취약할 수 있다. 위와 같은 공격유형을 다루기 위하여 우리는 XaT-SOAP (XML 기반 공격 감내 SOAP 메시지)라는 새로운 접근법을 제안한다. 이 접근법에서 우리는 먼저 SOAP 메시지의 인도 구조를 온톨로지를 사용해서 만들고 SOAP 메시지 헤더부에 부착한다. 부착된 온톨로지를 접수단에서 검증하면 XML공격을 탐지할 수 있을 것이다. 또한 본 연구에서 SOAP 메시지에 관한 모든 변경사항은 로그에 쓰여 진다. 그래서 만약 보안 실패가 일어나면 우리는 이 로그를 참조할 수 있고, 보안이 성공한 지점에서부터 복구될 수 있다. 우리는 수행성능 평가를 통해서 효율성 측정과 함께 제안된 접근법의 구현을 제공한다.

키워드: SOAP 메시지, 온톨로지, 웹서비스, XML기반공격

Abstract
Several Web Service Security (WS-Security) technologies are used aiming at securing exchanges of SOAP messages among partners in a collaborative environment. However, although all of these security standards, SOAP message can still be vulnerable to XML-based attacks. In order to deal with these types of attacks, we propose a new approach called XaT-SOAP (XML-based Attacks Tolerant SOAP messages). In this approach, we first build SOAP message elements structure using ontology and then attach it in SOAP message’s header. If we validate this ontology in the receiving end, we will be able to detect XML-based attacks. Also, in our approach, all modifications on SOAP messages are written to a log. So if security failures have occurred, we can check this log and recover from effect of successful execution. We will provide an implementation of our proposed approach along with efficiency measurements through performance evaluation.

Key words: SOAP message, Ontology, Web Service Security, XML-based attack

서 론
Several WS-Security standards are widely utilized aiming at securing exchanges of SOAP messages among partners in a collaborative environment. However, despite all of these WS-Security mechanisms, SOAP messages may still be vulnerable to a class of attacks based on the malicious interception, manipulation, and transmission.

McIntosh and Austel [1] have shown that the content of a SOAP message protected by an XML Digital Signature as specified in WS-Security can be forged without invalidating the signature. This is so called XML Rewriting attack [2], is possible because XML Digital Signature refers to a signed object of an XML Document in a way that does not take care of the location of that object. Moreover, SOAP extensibility model allows a SOAP message to contain a SOAP header element that is not recognized by the receiver and WS-Security allows multiple security headers to exist in the same SOAP message. All of these features along with the weaknesses of XML Digital Signature may work as a weapon to perform XML Rewriting attack on SOAP messages.

In order to deal with these types of attacks, we propose a new approach called XaT-SOAP (XML-based Attacks Tolerant SOAP messages). In this approach, we first build SOAP message elements...
structure using ontology and then attach it in outgoing SOAP message’s header. By validating this ontology in the receiving end, we will be able to detect XML Rewriting attacks early in validating process. Furthermore, in our approach, all modifications on SOAP messages are written to a log before they are applied. So if security failures have occurred, we can check this log and recover from effects of successful execution. We will also provide an implementation of our proposed approach along with efficiency measurements through performance evaluation.

The rest of the paper is proceeds as follows. Chapter 2 presents related works. Chapter 3 describes our proposed approach. Chapter 4 shows our implementation. Chapter 5 describes performance evaluation and Chapter 6 highlights conclusion and future work.

2. Related Studies

There have been much research toward protecting SOAP messages. We compared them with our approach as shown in Table 1.

Authors in [2] propose an advisor for WS-Security policies, a tool that generates a security report by running queries that check for over thirty syntactic conditions corresponding to errors found during security reviews. Similar work with recovery mechanism in addition, called Bit-Stream, has been conducted in [3]. However, it is important to note that a simple variant of XML Rewriting attacks is the deletion of elements. In order to detect this form of attack with these approaches, every element should be declared as mandatory. This reduces the flexibility of the XML technology and decreases the performance in the validation phase.

In [4], the authors proposed an inline approach that takes into account information about the structure of the SOAP message by adding new header element called SOAP account. In [5,6], authors demonstrated that inline approach cannot detect all kinds of attacks and proposed an extension to take into account new characteristics of SOAP message such as the depth information and parent elements of the signed node. In [7] the proposed to use a new header in SOAP message containing the signed elements positions in the message. While these extensions looks an appropriate way to detect XML Rewriting attacks, they still do not have a solution for recovering task.

3. System Design

Taking the shortcomings of all the discussed solutions, in the previous section, into consideration, we will propose an extension to existing message flow. The main purpose of proposed extension is to detect and correct any XML Rewriting Attack attempts. To achieve this, we propose following four new modules: initialization module, diagnostics module, decision making module and recovery modules. The modules and the relationship between them and other modules are shown in Figure 1.

On the sender side, in Figure 1, at first, the Requester will acquire the required security token from the Security Token Service and then the protocol stack generates SOAP envelopes that satisfy its policy. Once the policy is incorporated in SOAP message, it is sent to the initialization module in which the structure information of SOAP message is built. We used ontology to build this SOAP

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Warning Analyzing Ability</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Decision Making Ability</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Recovery Ability</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Partial</td>
<td>Yes</td>
</tr>
<tr>
<td>SOAP Flexibility</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
elements structure information. In the content of our proposed approach ontology is able to let us build hierarchies and thesauri that can be used for expressing how elements within SOAP message relate to one another [8]. So before sending SOAP message we build ontology of SOAP elements structure and attach it in SOAP message header.

After SOAP message is enriched with additional security headers and is ready to being sent, logging mechanism of the initialization module is activated. The logging mechanism has two major functions. One function is to intercept the message and log it in a reliable storage for the future recover process. The other function is to checkpoint critical states periodically to backups.

The diagnostics module resides on the receiver side and monitors the incoming message, detects all the security threats that occur in SOAP message, and generates security report. Any legitimate receiver of the message, as soon as message arrives, builds ontology-based schema of SOAP elements structure information of the received message and compares it with the attached structure information. The differences of these ontology-based schemas are reflected in security report.

By observing this security report the receiver can immediately detect if there has been a rewriting attack on a message. This action is performed in the warning analyzer module. Specifically, the security report, based on comparison, states what sort of security vulnerability may exist and identifies the suitable recovery action for the detected problem.

The recover module applies the appropriate recovery actions set by the warning analyzer module. The module is responsible to retrieve the logs from the log DBMS. If message has been attacked and recovery action is needed, recovery module will compliment the missing parts of SOAP message.

4. Implementation

We will provide an implementation of our

![Figure 1 Proposed extension to SOAP message flow](image)
proposed method. We have chosen Apache Extensible Interaction System (Axis), a SOAP processor engine, for the creation and processing of SOAP messages. The prototype has been developed using the NetBeans 6.9 environment.

Suppose a client wants to make a loan request. To simulate the attacker in this scenario we design a Java class RewritingAttack which represents the malicious host. After receiving the message from the legitimate sender it updates the message in the following way: <MessageID> element is moved into new bogus header element, making this <MessageID> element meaningless for the receiver, and creates its own <MessageID> element. The result of this modification on SOAP message is shown in Figure 2.

From Figure 2, it is easy to recognize that the attacker has bypassed the replay detection checks, and by replaying this message, we may cause the same request to be process several times, making the client pay several times for the same query and forcing the server to do redundant work.

In order to avoid these kinds of attacks, in the initialization module of our model, we build ontology of the SOAP elements structure information. We build ontology in Protégé Tool. Jena Ontology API provides an easy to use APIs for ontological profile development and good reasoning support. For the purpose of the implementation, Jena APIs is used for interrogating, manipulating, inferencing and querying the ontologies. So in our scenario, ontology of outgoing SOAP elements structure is built, attached it in SOAP message header (shown in Figure 3) and validated at the receiving end.

5. Performance Evaluation

We have used Colored Petri Net (CPN) tools to conduct performance evaluation. The simulation of XaT-SOAP in CPN tools is shown in Figure 5.

The model starts working from WS-Requester place. This place holds all initial SOAP requests. As the message flows, the WS-Security transition will intercept the message and enrich it with security headers. As soon as enrichment is done, the request will be logged. In Attack transition, we have simulated several attacking cases including Replay Attack, Redirection Attack and Multiple Header Attacks. Once attacks are simulated, Detection Mechanism transition is activated. In this transition we simulated four methodologies which discussed in Chapter 2. Those are WS-Security, Policy Advisor, SOAP account approach and our approach, XaT-SOAP. The result of the detection-based comparison is reflected in Figure 4. From the graph it is easy to recognize that XaT-SOAP approach bettered the other approaches and therefore, can be effectively utilized in battling against XML Rewriting attacks.

Once detection process is done and threats are identified, we pass security report to Recovery Mechanism transition. If security failures are occurred, we could easily check the log and recover from effects of successful execution.
6. Conclusion

In this paper, we proposed an ontology-based detection and log-based recovery mechanisms to combat with XML Rewriting attacks. Results implementation and performance of this approach showed that it is able to detect and recover from XML rewriting attacks early in the validating process. Thus, we believe that our approach is able to bring a reasonable protection to entire Web Service environment.

Our current method, however, may cause a reduction of the effectiveness when changes in the behaviors of the known attack happen or when new attacks appear. In order to overcome this, in our future works, we are planning to add self-adaptation ability to the changes that occur in the patterns of attack.

Reference


