Applying Security for RESTful Web Services – Limitations and Delimitations

Mohamed Ibrahim B¹, Mohamed Shanavas A R²

¹Software Solution Architect & Research Scholar, India
²Associate Professor, Jamal Mohamed College (Autonomous), Tiruchirappalli, India

Abstract—The Service Oriented Architecture (SOA) becomes an essential element of modern Enterprise Application Integration (EAI). Among the available SOA implementations, Web Services are most preferable choice by the enterprises as they operate on simple Internet protocols. In principle, web services use SOAP protocol as a base for transmitting requests and responses in between service requester and service provider. In recent years, another kind of web services, RESTful web services, are emerging that is even simpler as it uses normal HTTP methods and URIs. Many security solutions for SOAP web services are already exists in market as products and standards; and also many researches are still going on. The RESTful web service is not a standard and it does not contain any security components within it. Research scholars and product based corporations are providing security recommendations for RESTful web services. However, there is no standard security solution defined as of now. And also applying security for RESTful web services is not a straight forward approach; there are few limitations and delimitations that we should consider in developing a security solution for RESTful web services. This paper analyses those limitations and delimitations with the authors’ practical implementation experience of applying security to RESTful web services.

Keywords—RESTful Web Services, Security, Secure Web Services, SOAP, EAI

I. INTRODUCTION

The Enterprise Application Integration (EAI) aims in consolidating and coordinating Software applications that are operated in an enterprise at various levels. The service based integration is the modern era in EAI that offers distributed and loosely coupled application environment. Loosely coupled design allows the applications to run independently of each other where the tightly coupled applications are highly dependent on one another. The main goal of this loosely coupled design is to reduce the risk that an introduced change in one application will result in unanticipated changes in other relevant applications. The SOA is based on loosely coupled design, as service is an independent function that can be consumed by other applications on demand.

Web services made SOA simpler and it is preferred choice by many enterprises. The SOAP is the core protocol which transmits web services data in between service requester and service provider. Roy Fielding [1] introduced a term named Representational State Transfer in his doctoral dissertation in the year 2000; the data and functionalities are considered as resources and they are accessed using Uniform Resource Identifiers (URIs). This kind of service implementation is called as RESTful Web Services. The one which uses SOAP protocol is known as SOAP Web Services.

Representational State Transfer (REST) is an architecture design for the networked applications, where the requests and responses are built around the transfer of representations of resources. Each resource has one or more representation such as XML and JSON. RESTful web service is a web API that is implemented using HTTP and the principles of REST [2].

This paper is critically analyses the limitations and delimitations while applying security for RESTful Web Services as it does not contain any security solutions by itself.

II. SOAP VS REST—APPLYING SECURITY

REST (REpresentational State Transfer) is a Software application architecture modelled after the way data is represented, accessed, and modified on the web. For SOAP-based web services, the standard defines envelopes to transmit requests and responses. In contrast, the REST concepts coined by Roy Fielding in his Ph.D. dissertation simplify access to web services by reusing existing and widespread standards instead of adding new layers to the communication stack [3]. In the REST architecture, data and functionality are considered resources, and these resources are accessed using Uniform Resource Identifiers (URIs), typically links on the web. The resources are acted upon by using a set of simple, well-defined operations. The REST architecture is fundamentally client-server architecture, and is designed to use a stateless communication protocol, typically HTTP. In the REST architecture, clients and servers exchange representations of resources using a standardized interface and protocol.
The REST architectural style enforces the following six constraints [3]; however all of them are not mandatory.

i. Uniform interface for access
ii. Stateless transactions
iii. Cacheable business data
iv. Client-Server implementation
v. Layered system model

In the recent years, one more architectural style is employed for building Web services: RESTful services, as a light-weight simple architecture and an alternate to services based on the WS- standards (also known as “SOAP Web services”).

The following characteristics which encourage RESTful applications to be simple, lightweight, and fast:

- *Resource identification through URI*: The RESTful web services expose a set of resources that identify the targets of the interaction with its clients.
- *Uniform interface*: Resources are manipulated using a fixed set of four create, read, update, delete operations (PUT, GET, POST, and DELETE).
- *Self-descriptive messages*: Resources are decoupled from their representation i.e., the response in multiple acceptable formats such as XML and JSON.

RESTful web services are web applications built upon the REST architecture. They expose resources (data and functionality) through web URIs, and use the four main HTTP methods POST, GET, PUT, and DELETE to create, retrieve, update, and delete resources respectively. RESTful web services typically map these four main HTTP methods to the so-called CRUD actions: create, retrieve, update, and delete.

RESTful web services are focused in accessing the named resources through a single consistent interface. As REST uses standard HTTP, it is much simpler than SOAP web services. Also REST allows many different data formats such as XML and JSON, but SOAP only permits XML format. Compared to SOAP, REST provides better performance and scalability [4]. We can cache the REST read, but SOAP based reads cannot be cached.

“REST APIs must be hypertext-driven” said by Roy Fielding, the architect of REST. According to him, the REST API should follow the following rules.

- It should not be dependent on any single communication protocol.
- It should not contain any changes to the communication protocols (aside from the specifications).

- It must not define fixed resource names or hierarchies; Servers must have the freedom to control their own namespace.
- It should never have “typed” resources that are significant to the client.
- It should be entered with no prior knowledge beyond the initial URI and set of standardized media types that are appropriate for the intended audience i.e., expected to be understood by any client that might use the API.

The Simple Object Access Protocol (SOAP) is a protocol specification that is used for exchanging structured information in web services. It uses XML format for message interchange between client and server. It is independent of the transport protocol [5]. A SOAP message contains two parts: Header and Body. Several security standards are developed so far for SOAP web services. The WS-Security is a standard that aims in providing end-to-end security for the SOAP web services. It incorporates and works with other relevant supporting standards such as WS-Policy.

Pautasso et al. [2] compare RESTful and SOAP Web services on three levels: (i) Architectural principles, (ii) Conceptual decisions, and (iii) Technology decisions. On the level of architectural principles, Pautasso et al. analyze three principles (protocol layering, dealing with heterogeneity, and loose coupling) and note that both styles support these three principles. However, they can identify only one aspect common to both styles – loose coupling to location. Consequently, they conclude that it’s not possible to make a decision at this level and proceed with more detailed analysis. At the level of conceptual decisions, they compare nine different decisions and find that RESTful services require the designer to make eight of them, vs. only five for SOAP WS [6]. However, SOAP WS have many more alternatives than RESTful services. Finally, in the technology comparison, they identify ten technologies that are relevant to both styles. In this comparison, SOAP WS once again offer many more alternatives than their RESTful counterparts. Based on these results, the authors recommend using REST only for ad-hoc integration and using SOAP WS for enterprise-level application integration where transactions, reliability, and message-level security are critical [8].

SOAP uses Web Services Description Language (WSDL) to describe the web services and uses Universal Description, Discovery and Integration (UDDI) to register the services [7].
It is possible for SOAP web services to provide end-to-end security even the SOAP messages pass-through the intermediaries as required by the underlying network. But with RESTful web services, only message security is provided by transport protocol (HTTPS) which offers only point-to-point security. Also it is not possible to reseed the messages in case of any communication failure.

One of the major benefits of RESTful web services is the flexibility for data representation which transfers among servers and clients. And RESTful web services are lightweight that they do not have any extra XML mark-up as SOAP messages have.

While using SOAP message, it has a contract between the server and the client; but you are forced to do such kind of contract at code level in the case of using REST. Considering the performance of data transmission between the service requester and service provider in the SOAP vs REST design, we can use the modern data compression and parsing solutions such as gzip and Fast Infoset (FI) which efficiently works for a specific design; and the difference in the parsing speed is negligible.

We have experience in designing a banking application using web services which sends huge amount of data as request for a web method. It works well with SOAP web services that the SOAP architecture supports to send even huge amount of data in XML as SOAP request. Contrary with RESTful web service, the REST expects the request data should be embedded as part of the request URL. So that it is required to split the request data and send them through multiple requests to the server. It is not desirable method by enterprises. And other way is to send the whole request data by embedding it in the body of a POST request, but certainly it is not really a true design of RESTful web services [8]. RESTful web service is good and possibly the optimal choice for most scenarios but there are certainly cases where SOAP web service is a better fit.

III. SECURITY CONSTRAINTS FOR RESTFUL WEB SERVICES

Few web servers support HTTP GET only for web service operations without complex parameters. Some web clients limit the size of the HTTP GET URL, typically to 2000 characters or less; so it is required to shorten the parameter names and/or limit the number of parameters. A RESTful web service sends responses in a standard data format such as XML and JSON, and it cannot send messages with attachments.

The RESTful web services do not have the management features such as security and reliability as these features are usually carried out by SOAP headers [9,18]. As REST is a pattern and not a standard, there is no concrete security solution available.

A typical security solution to RESTful web service should include the following:

- Scan the content of HTTP headers as well as HTTP query string (in the URL) of every REST request.
- Apply restrictions on calling HTTP methods such as GET, PUT, DELETE, and POST in specific to web service context.
- Examine the request payloads and request repetition.

Thus REST is a modern architectural design for connecting networked applications. The core idea is to utilize the simple HTTP protocol instead of complex mechanism such as RPC, CORBA, and even SOAP [10,17]. However, it is very important to observe that REST is not a standard.

In general, the WWW which is based on HTTP can be viewed as REST-base architecture. These RESTful applications use the HTTP requests for Create/Read/Update/Delete (CRUD) operations [11]. Generally the four primary HTTP methods are utilized as follows:

- GET – Read a specific/collection of resources.
- PUT – Update a specific/collection of resources.
- DELETE – Remove a specific resource.
- POST – Create a new resource.

RESTful web applications typically have the same attack vectors as standard web applications, including injection attacks, XSS cross-site scripting, broken authentication and cross-site request forgery (CSRF) [12].

The RESTful web service uses session based authentication either by establishing a session token or using a cookie. However, the usernames/passwords and session tokens should not appear in the URL query string, as this can be captured in web server logs and makes them intrinsically valuable [13]. To overcome replay attacks, REST service providers are encouraged to use time limited encryption key, encrypted session token and incoming IP address. Protection against using HTTP methods is also mandatory as the user can access POST (create) / PUT (update) methods also using the user’s privilege where the service provider is expecting the user to access only GET (read).
Thus the protection and auditing is required for privileged actions and sensitive resource collections, not every user has a right to every web service [14].

The following are some of the security implementation for RESTful web services [15,16].

- Input validation
- Secure session tokens
- Privileged HTTP method access
- Secure direct object references
- Secure parsing of input requests
- Strong typing of data
- Validate incoming content types (MIME)
- Validate responses
- Secure HTTP headers
- XML encoding and JSON encoding
- Encrypt messages
- Apply/use TLS (HTTPS) +, HTTP Digest, Two/Three-legged OAuth

IV. CONCLUSION

Apart from the standard SSL security for RESTful web services, any encryption standards and authentication algorithms such as OAuth are introduced in the industry. There are also message based end-to-end security mechanisms, like S/MIME, are practiced now a days. The TLS secure sessions are user specific and keys are generated on the fly, and the content will be encrypted again and again in the secure tunnel. The data cannot be cached as the web caches are not allowed to access the data inside the secure tunnel.

RESTful web services threats include session attacks, parser attacks, SQL/code/LDAP injections, Deniel-of-service (DoS), malware, buffer overflow and replay attacks. This paper discussed the important limitations and delimitations of RESTful web services. Even though REST is emerging and preferred choice by many enterprises, it is not defined as a standard supported by any consortium. This is also a factor that affects developing concrete security solution for RESTful web services.

REFERENCES


