IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Issue-1, Jan 2019 E-ISSN: 2347-2693

S-REST: A design of Secured Protocol for Implementation of RESTful Webservices

Chatti Subbalakshmi[1*], Rishi Sayal[2], H. S. Saini[3]

[1, 2 & 3] Computer Science & Engineering, Guru Nanak Institutions Technical Campus, Ibrahimpatnam, R. R. Dist., Telangana

Corresponding Author: subbalakshmichatti@gmail.com , Tel.: +91-9032312260

Available online at: www.ijcseonline.org

Accepted: 20/Jan2019, Published: 31/Jan/2019

Abstract—Representational State Transfer (REST) is an architectural style for developing web services and its key constraints are Use of Uniform Interface (UI), client-server based, stateless operations, and Resource caching. It is popular due to its simplicity and builds on the existing systems. Hence, many cloud providers such as Amazon, Google are moving their APIs from Simple Object Access Protocol (SOAP) to REST. Unlike SOAP, RESTful service doesn't provide standard for security while accessing web services. Hence, we considered the security issues in execution of RESTful web services and proposed a design of a secured model (S-Rest) over RESTful web services with 3-level security services at communication, Application and Management. The proposed architecture enhances the performance of RESTful web application.

Keywords— Webservices; RESTful; Security issues;

I. INTRODUCTION

Web Services are the services that are made available from a business's Web server for Web users or other Web-connected programs. Web services range from major services as storage management and Customer Relationship Management (CRM) down to much more limited services such as the furnishing of a stock quote and the checking of bids for an auction item. There are two types of Web Services: SOAP web services and RESTful web services [1].

SOAP (Simple Object Access Protocol) is a messaging protocol that allows programs that run on operating systems to communicate using Hypertext Transfer Protocol (HTTP) and Extensible Markup Language (XML). The merits of SOAP are simplified communications through proxies and firewalls; it has the ability to leverage different transport protocols including HTTP and SMTP, as well as others [2]. Its demerit is typically much slower than other types of middleware standards, including CORBA. There tends to be firewall latency due to the fact that the firewall is analyzing the HTTP transport.

REST (REpresentational State Transfer) is an architectural style for developing web services. REST builds upon existing systems and features of the internet's HTTP in order to achieve its objectives as opposed to creating new standards, frameworks and technologies [3]. The advantages of REST are REST-based interactions happen using constructs that are familiar to anyone who is accustomed to using the internet's Hypertext Transfer Protocol (HTTP). Details such as encryption and data transport integrity are solved not by adding new frameworks or technologies, but instead by relying on well-known Secure Sockets Layer (SSL) encryption and Transport Layer Security (TLS). RESTful web services can be written using any language. So, developers tasked with implementing such services can choose technologies that work best for their situation [4]. The disadvantages of REST are that since HTTP doesn't have any mechanism to send push notifications from the server to the client, it is difficult to implement any type of services where the server updates the client without the use of client-side polling of the server or some other type of web hook.

RESTful API is an application program interface that uses HTTP requests to GET, PUT, POST and DELETES data [3]. A RESTful API also referred to as a RESTful web service is based on representational state transfer (REST) technology, an architectural style and approach to communications often used in web services development. REST leverages less bandwidth, making it more suitable for internet usage [4]. RESTful APIs are used by such sites as Amazon, Google, LinkedIn and Twitter. The presumption is that all calls are stateless; nothing can be retained by the RESTful service between executions. REST is more useful in cloud applications.

In proposed S-REST model, we consider the security principles which can avoid the attacks during the execution of RESTful web services through a 3-level security model. A 3-Level Security services is being modeled for better authentication and authorization access of the RESTful web services. They are Communication level, Application Gateway level and Management level.

Rest of the paper is organized as follows, Section II contain the related work, Section III describes on security issues of Restfull web services, Section IV contain security principal of Restfull web services, Section V shows the architecture and functionality of each proposed model, Section VI contain applicability of proposed model and concludes research work with future directions.

II. Related Work

By comparing RESTful services and WS* services in [5], discuss some of the problems facing the existing RESTful services applied to large enterprise systems. They are: Identification of resources, Manipulation of resources through representations, Self-descriptive messages and Hypermedia as the engine of application state (abbreviated HATEOAS).

These principles describe the architecture of systems and interactions that make up the Web. The building blocks of the Web are called *resources*, which can be named as a target of hypertext. In response to a request for a resource, the client receives a *representation* of that resource, which may have a different format than the resource owned by the server. Resources are manipulated via *messages* that have standard meanings; on the Web, these messages are the HTTP methods. The fourth principle means that the state of any client–server interaction is kept in the *hypermedia* they exchange. Any state information is passed between the client and the server in each message, thus keeping them both stateless. WS* services do not have a single metaphor. Web Services Architecture document fromW3C describes four architectural models of WS*. One of the models is the Resource Oriented Model (implies REST) considered are limited to various standards: SOAP, WSDL, and others. New capabilities are added to WS* in the form of new standards.

The authors in [6] proposed RESTful and WS* services are compared on three levels: (1) architectural principles, (2) conceptual decisions, and (3) technology decisions. On the level of *architectural principles*, analyze three principles (protocol layering, dealing with heterogeneity, and loose coupling) and it's not possible to make a decision at this level. 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 WS*. 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, WS* once again offer many more alternatives than their RESTful counterparts. Based on these results, the authors recommend using REST for ad hoc integration and using WS* for enterprise-level application integration where transactions, reliability, and message-level security are critical. This study illustrates two key difficulties. First, it's difficult to select the most relevant principles to compare. Second, once the principles are selected, it's difficult to identify choices that are shared by the competing ideas.

The principles that are relevant to all systems available on the Web are discussed in [7]. They identify four system properties of RESTful services: (1) uniform interface, (2) addressability, (3) statelessness, and (4) connectedness. In RESTful Web services, these properties are embodied in resources, URIs, representations, and the links between them. Many WS-*services are stateless. Having a uniform interface shared by all services is the only property not supported by WS*. WS* services exhibit three of these four properties. Both styles of Web services possess certain characteristics that guide their design and development, although they are defined in ways that make it difficult to compare them side-by-side.

Proposed data security architecture in [8], includes encryption and verification services both at file and block storage level to satisfy the data protection needs of different cloud service models, especially computing service (IaaS) model. Our solution facilitates cloud consumers to store their sensitive information and application data objects in corresponding storage devices with complete data privacy and security. It also leverages both CSP and cloud vendors for achieving transparency in security processes of cloud.
Proposed approach in [9], the WS-Security User name Token and secondary password are added into the HTTP header. By this way, the approach allows service providers to define their own authentication which makes up for the disadvantages of the current security aspect of REST-style Web services, especially when Basic HTTP Authentication and HTTP Digest Authentication are not applicable.

The authors of [10] proposed approaches an important part of this requirement by introducing a REST-ful CoAP message authentication scheme. The overarching goal of this work is, though, to establish a message-oriented security layer for CoAP. Here, specific challenges are stemming from the architectural style REST and the resource-restrictiveness of IoT networks and devices.

The authors of [11] introduce the REST security protocol to provide secure service communication, together with its performance analysis when compared to equivalent WS-Security configuration.

III. SECURITY APPROACHES

In recent years, Representational State Transfer (REST) represents 70% of public APIs using light-weight data interchange format, i.e., JavaScript Object Notation (JSON). It is software architecture to access web services in much simpler way than possible with Simple Object Access Protocol (SOAP) using HTTP. Compared to SOAP, RESTful services need more security in accessing the web resources.

Security Approach: Security issues and attacks may occur in the RESTful web services: Client Impersonation, Access Tokens, Authorization Codes, Resource Owner Password Credentials, Credentials Guessing Attacks, Phishing Attacks, Cross-Site Request Forgery and Click jacking.

IV. SECURITY PRINCIPALS OF RESTFULL WEB SERVICES

The security attacks can be avoided using the following Security principles of RESTful web services:

HTTPS - REST services must only provide HTTPS endpoints. This protects authentication credentials in transit. It also allows clients to authenticate the service and guarantees integrity of the transmitted data.

Access Controls - Non-public REST services must perform access control at each API endpoint. Web services in monolithic applications implement this by means of user authentication, authorization logic and session management. This has several drawbacks for modern architectures which compose multiple micro services following the RESTful style.

JWT (JSON Web Tokens) - There seems to be a convergence towards using JSON Web Tokens (JWT) as the format for security tokens. JWTs are JSON data structures containing a set of claims that can be used for access control decisions. A cryptographic signature or message authentication code (MAC) can be used to protect the integrity of the JWT.

API Keys - Public REST services without access control run the risk of being framed leading to excessive bills for bandwidth or compute cycles. API keys can be used to mitigate this risk. They are also often used by organization to monetize APIs; instead of blocking high-frequency calls. API keys can reduce the impact of denial-of-service attacks. However, when they are issued to third-party clients, they are relatively easy to compromise.

Input validation – Do not trust input parameters/objects. Validate input by length/range/format and type. Achieve an implicit input validation by using strong types like numbers, Booleans, dates, times or fixed data ranges in API parameters.

Management end points - If management endpoints must be accessible via the Internet, make sure that users must use a strong authentication mechanism. Expose management endpoints via different HTTP ports or hosts preferably on a different NIC and restricted subnet.

Error Handing – Respond with generic error messages - avoid revealing details of the failure unnecessarily. Do not pass technical details to the client.

Audit Logs – Write audit logs before and after security related events. Consider logging token validation errors in order to detect attacks.

Security headers – To make sure the content of a given resources is interpreted correctly by the browser, the server should always send the Content-Type header with the correct Content-Type, and preferably the Content-Type header should include a char set. The server should also send an X-Content-Type-Options:nosniff to make sure the browser does not try to detect a different Content-Type than what is actually sent.

V. METHODOLOGY

Representational State Transfer (REST) is an architectural style for developing web services. The four key constraints in implementing RESTful web services are Use of Uniform Interface (UI), client-server based, stateless operations, and Resource caching. REST is popular due to its simplicity and the fact that it build upon the existing systems and features of internet's HTTP. Because the calls are stateless, RESTful is useful in cloud applications and many cloud providers such as Amazon, Google are moving their APIs from Simple Object Access Protocol (SOAP) to REST. When more and more cloud data is moving through APIs rather than browsers, developers need consistent and secure methods to access and manipulate cloud hosted services. SOAP based APIs have standards like WS-Trust and WS-Security to define the authentication and authorization. But RESTful service doesn't provide standard for security while accessing web services.

The proposed S-REST model provides security services over RESTful web services by introducing 3-level security services.

In this model, we proposed a new concept of dynamic cipher suite which can be more secure HTTP communication over Transport Layer Security (TLS). In the process of connection establishment using 3-way handshake protocol, client and server agree on one of the cipher suites from browser-dependent set of cipher suites for their communication. The cipher suite is selected by the server from the client's cipher suites. To avoid attacks like Man-in-the-middle, Client impersonation, Credentials guessing attacks, Phishing, Cross-site request forgery, click jacking, we need to provide additional security approach. Hence, we introduced dynamic cipher suite which changes the set of cryptographic algorithms over a time by server.

It also provides an Application gateway to handle security issues like Access control, authorization and authentication on RESTful API.

Apart from these security measures for communication and accessing of RESTful API, this project also provides the security services using Machine learning algorithms for monitoring data security and data classification for user behaviour.

In this S-REST proposal, we consider the security principles which can avoid the attacks during the execution of RESTful web services through a 3-level security model.
A 3-Level Security services is being modelled for better authentication and authorization access of the RESTful web services. They are Communication level, Application Gateway level and Management level.

1. **Communication Security service**
   - It handles the security principles HTTPS, security headers, validate content types.
   - Provides security through proposed Dynamic cipher suites over HTTPS.
2. **Application Gateway Security service**
   - It handles the security principles access control, JSON Web Tokens, API keys, Input validation, Error handling.
3. **Management Security service**
   - It handles the security principles Management endpoints, Audit logs using Machine learning.
   - Data Monitoring and Classification to identify business-critical data and analyze access patterns and user behavior
   - Data Security to be proactive with security compliance and achieve preventive security.

Applicability of proposed method:
- S-REST can be adopted in any RESTful web services.
- Additional security can be provided to Social networks such as Face book, Twitter, LinkedIn, Google
- Security for personal information is provided in Financial sectors such as Internet banking, credit card payments, E-commerce item purchases
- Benefits for the users will be protected by securing the data in Government servers. Or Public Online Services.

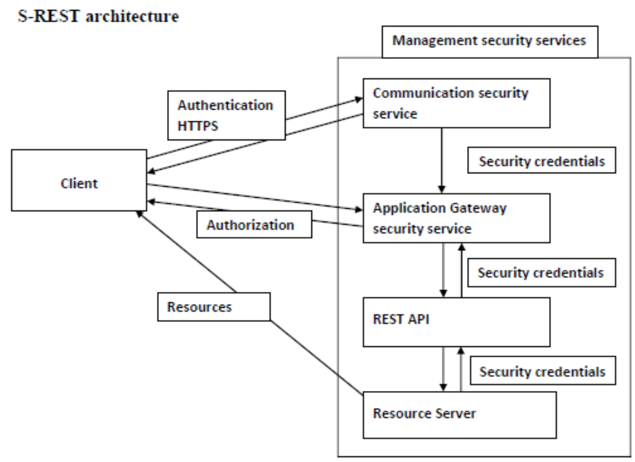

Figure 1: A three module S-REST architecture

VI. CONCLUSION AND FUTURE SCOPE

The security attacks are frequently occurring in the networks for misuse of the information and applications through web services. Unlike the SOAP, a RESTful web service does not provide the standard for security. Hence, developer has to adopt security principals to achieve the security on different levels. In this paper, proposed the security principles for RESTful web services in a 3-level security model. It provides more secured communication using proposed dynamic cipher suites. In second level, it provides a secured way to access RESTful web services through Application API Gateway. In top level, it provides the security services that use machine learning to automatically discover, classify and protective sensitive data.

The proposed S-REST will be implemented on TLS1.2 using Dynamic cipher suites. Further, it can be enhanced with the next version of TLS, i.e., TLS1.3. And also more security principles can be added in the design of Application gateway. Management security service module can be enhancing by adapting upcoming advanced machine learning algorithms.

REFERENCES

[1]. Meiko Jensen, Nils Gruschka, Ralph Herkenhöner, "A survey of attacks on web services", Computer Science Research and Development, Springer, November 2009.
[2]. Hirsch, Frederick; Kemp, John; Ilkka, Jani. "Mobile Web Services: Architecture and Implementation", John Wiley & Sons, 2007.
[3]. Richardson, Leonard; Amundsen, Mike, " RESTful Web APIs", O'Reilly Media, retrieved 15 September 2015.
[4]. "Web Services Architecture". World Wide Web Consortium. 11 February 2004. 3.1.3 Relationship to the World Wide Web and REST Architectures. Retrieved 29 September 2016.
[5]. Fielding, "Architectural Styles and the Design of Network-based Software Architectures", Doctoral dissertation. Technical report, University of California, Irvine, 2000.

[6]. Pautasso, O. Zimmermann, and F. Leymann, "RESTful Web Services vs. "Big" Web Services: Making the Right Architectural Decision", In WWW '08: Proceeding of the 17th international conference on World Wide Web, pages 805–814, New York, NY, USA, 2008. ACM

[7]. Richardson and S. Ruby, "RESTful Web Services", O'Reilly, Oct. 2007

[8]. Dharmendra S. Raghuwanshi, M.R.Rajagopalan, " MS2: Practical data privacy and security framework for data at rest in cloud", Computer Applications and Information Systems (WCCAIS), 2014 World Congress on 17-19 Jan. 2014.

[9]. Dunglu Peng, Chen Li, Huan Huo, "An extended UsernameToken-based approach for REST-style Web Service Security Authentication", Computer Science and Information Technology, 2009. ICCSIT 2009. 2nd IEEE International Conference on 8-11 Aug. 2009.

[10]. Hoai Viet Nguyen, Luigi Lo lacono, "REST-ful CoAP Message Authentication, Secure Internet of Things (SIoT)", 2015 International Workshop on 21-25 Sept. 2015.

[11]. Gabriel Serme, Anderson Santana de Oliveira, Julien Massiera, Yves Roudier, "Enabling Message Security for RESTful Services", Web Services (ICWS), 2012 IEEE 19th International Conference on 24-29 June 2012