Security in Grid Based Robot Navigation using REST

Raja Mukhopadhyay [1]

[1]Jadavpur University

October 30, 2023

American Journal of Software Engineering, 2016, Vol. 4, No. 1, 1-5
Available online at http://pubs.sciepub.com/ajse/4/1/1
©Science and Education Publishing
DOI:10.12691/ajse-4-1-1

Security in Grid Based Robot Navigation using REST

Raja Mukhopadhyay*, I Mukhopadhyay

Department of Information Technology, Institute of Engineering & Management, Kolkata, India
*Corresponding author: rjmkhrj@gmail.com

Abstract The essentiality of having a secure system is indispensable more importantly it is essential to keep that system protected from outside intrusion. The implementation of one technology along with the other is useful to strengthen the functioning of the former or latter. The severity of a breach in a system can compromise the system's integrity preventing users from its intended use. The unison of Networking with the Internet of Things brings about the efficient working of smart systems and for a system that implements this technology a good build up of security system is essential. This paper discusses about a security mechanism implemented within Representational State Transfer (REST) architecture for a robot navigation system.

Keywords: *REST, internet-of-things, HTTP, XML*

Cite This Article: Raja Mukhopadhyay, and I Mukhopadhyay, "Security in Grid Based Robot Navigation using REST." *American Journal of Software Engineering*, vol. 4, no. 1 (2016): 1-5. doi: 10.12691/ajse-4-1-1.

1. Introduction

The Internet of Things (IoT) involves connecting appliances to make them smarter or performance wise better. In the domain of IoT a constant connection to a network or a server is required for the large sets of information to be stored into. The fusion of IoT and cloud epitomizes the concept of a smart internet driven system. To implement networking, the Representational State Transfer (REST) architecture has been implemented. Basic authentication and token based authentication has been implemented in the paper.

In designing Grid Based Robot navigation an interface is required for the control of the bot inside the room. REST was found to be the most appropriate since it provides a user-friendly environment. As a prototype, we have made use of the Jersey framework. Since REST is inherently associated with HTTP methods, we have used REST methods to perform suitable operations in our paper [1]. Representational State Transfer makes use of HTTP methods to send, alter, and erase data. The return type after a method call is in the form of hypertext (texts that contain links to other texts). REST involves the presence of resources each of which is identified by an URI (Uniform resource Identifier). On making a request the response received is either in JSON (JavaScript object notation) or in XML format [3].

2. Background Study

Data collected from the client requires a storage space that is commonly present in the servers. A server is a computer hosted in one of a data centre in a different location than that of the client machine that in some way or the other serves the client. Cloud servers are generally preferable for better resource allocation. In case of a cloud server the storage space is typically spread over several machines [2]. The concept of virtualizing a resource comes into the picture in which a single physical machine is divided into several virtual machines and each virtual machine shares the resources of the physical machine.

The fusion of IoT and cloud computing has pervaded all the areas in a modern system. The cloud serves as a storage layer for the large amount of data generated from the sensors connected to the cloud environment. Apart from the cloud being a storage space it also benefits the components of an IoT application through the processes of data retrieval and modification. Resource optimization is one significant benefit of the cloud. In an open source cloud the storage space is shared by several users who together consult a cloud vendor. In our paper, we have used IaaS (Infrastructure as a Service) since the cloud is being used only as a storage medium. A private cloud environment has several advantages compared to a public cloud or hybrid cloud. Private cloud provides a layer of security since it is present behind a Firewall or DMZ. A private cloud is dedicated for a single user because of which the user can modify the environment per his needs without interference from third parties. We have specifically made use of own-cloud along with a storage space of 4 gigabytes. The direct connection between the cloud server and the client side API is done using the Raspberry Pi that serves as a BOT collecting data in a room [5]. The BOT functions to collect information regarding obstacles in the room and transfers the information to the cloud. Cloud storage security issues regarding storage have been mentioned in this paper. Using Advanced Encryption Algorithm (AES) for encryption, hashing techniques are implemented [4].

The Internet of Things (IoT) may generate data that can be different depending on the mode of generation [5]. Data can be restricted to simple logic operations or can be encrypted to protect the identity of the data and preserve integrity. Cloud IoT refers to the unison of cloud computing and Internet of Things for a smarter system encompassing all the characteristic of a closed system in which intervention from foreign parties is strictly restricted. For securing data several security implementation strategies can be used, SSL (Secure Socket Layer), user based authentication and token based authentication. This paper discusses in the upcoming sections how securing data generated from a Grid Mapping system can be done, the paper also discusses the mechanism of token based authentication using REST and user authentication.

3. Functioning of Cloud and IoT as Unit

Figure 1 shows that data generated from IoT enabled devices vary in their type [7]. Data may be transferred from databases, data collected from sensors like humidity, temperature and weather conditions as well. Any data requires a secure network for its transfer, we generally encrypt data with popular encryption algorithms like AES, DES etc.

Data security as well as storage security are interrelated to each other- the encrypted data may represent user identity that is stored in the system to be verified later during access. In case of token based authentication, the stored tokens may be transferred to the user in encrypted format. The preconceived notion that there may not be any intruders in the network is false for which Intrusion Detection System (IDS) and Intrusion Prevention System (IPS) must be installed in every system.

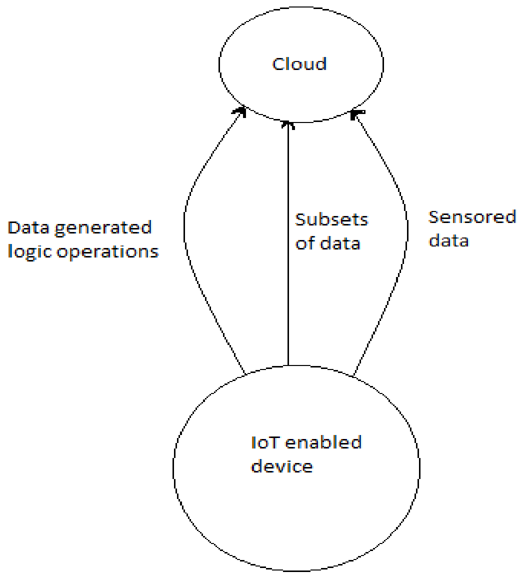


Figure 1. Functioning of cloud and IoT as a unit

3.1. HTTP Methods in REST

In this paper, we have made use of HTTP methods that are inherently associated with REST.

GET Method:

The GET method is used to fetch data from the server. The main characteristic of the GET method is it always returns the same result, if used on the same resource URI. However, there are some exceptions, which we have shown in the paper.

POST Method:

Another method is POST which is used for update of a resource or for creation of new resource. In this paper POST, has been used to create resources. POST may return same or different results depending on its use.

DELETE Method:

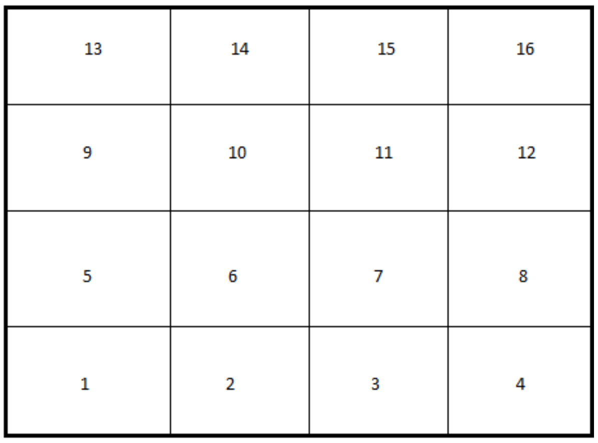
The DELETE Method is used to erase a resource using its UID. The DELETE method should be used cautiously as subsequent use of the DELETE operations may hamper the entire system of resources. If the DELETE method is used on a resource using a specific ID and then used again on the same resource the results would vary since the resource had already been deleted in the first call of DELETE.

PUT Method:

The PUT method is used to modify an existing resource. It is used when the user deals with a specific resource unlike POST in which the user is unknown of the new resource's location.

3.1.1. Mechanism and Working

For this paper lets decide that the room is divided into sixteen discrete regions each identified by a single ID. The value of the ID starts from 1 and continues till 16. 1 denotes the first grid identified by the coordinates (0, 0) and 16 refers to the last grid (4, 4). The Figure 2 gives a graphical picture of the organization of the room.



13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

Figure 2. Grid organization in a room

The BOT may start from any position in the room and accordingly frame its path depending on the presence of nearby obstacles or entities present in adjacent grid positions. The data collected by the BOT is automatically transferred to the cloud server and the client side programming ensures that the BOT functions as instructed by the user [10]. There may be the presence of two or more obstacles in a single grid depending on the functionality of the user.

From the perspective of client side interface using RESTFUL web API, the retrieval process of the information on grids has been implemented by the GET request. We provide a resource API that helps to identify the resource to be fetched, also mentioning the type of format in which the aim is to get back the request. XML format has been preferred over JSON because of several reasons. Firstly, parsing in XML is quicker as compared to JSON, XML provides support for namespaces and XML unites well with HTML. We provide a graphical representation of the data retrieval process for GET request. The status of the HTTP response indicates the reply from the server. A *200 OK* response indicates success from the part of the server, whereas the status code *400* represents that the request sent to the server was erroneous and could not be understood.

Apart from the above basic application of GET request the primary obstacle detection process has been brought about by fetch information about individual blocks we extend the resource ID with the individual resource ID of a block. The coordinates (0,0) refers to the first grid position.

Apart from the fetching of the information and coordinates for a block using the above method it also mentions whether a obstacle is present on that grid or not. The GET request functions as an information retrieval system as well as an obstacle detection query. The following image shows the individual fetching of a grid via the GET request. The content length is 84 and the server is Apache-Coyote/1.1, content type is application/xml.

The user has the capability to post obstacles or introduce obstacles in the room using the POST request. For the POST request, it is important to mention the grid in which the obstacle will be placed. The server responds through the status report 200 indicating that the obstacle or entity has been placed at the requested position.

The PUT is used to change the position of an obstacle i.e. to place it from one grid to the other. We have assumed that the all the entities in the room are dynamic in nature, constantly changing positions with time.

4. Results and Analysis

The results of the simulation have been derived using Jersey framework in Java web based API. The JAX-RS used in Java web based API supported by Maven provided the platform to set up the client side simulation. Individual server requests were created by mapping the specific resource URIs with the corresponding java methods. For creating the database, a mock stub was created by using hash maps.

Postman simulation tool was used to send the requests for GET, POST, PUT after which the server replied with the appropriate message. The *200 OK* response meant that the server could successfully process the request. The time taken for a query to be processed along with server reply depended on the HTTP method used and the number of resources to be fetched. Thus, the user request to fetch the entire set of grids *localhost:8080/robot/webapi/grids* would take much more time compared to the request sent to fetch information about one grid *localhost:8080/robot/webapi/grids/.* Table 1 below shows the various retrieval of grid information using HTTP methods.

To detect the position of obstacles we determine the grids where the obstacles are present. We make use of time intervals to indicate that all the entities in the room are dynamic in nature. During the first user request at T=2 the server responds through a series of numbers that indicate the total number of obstacles present in the room along with their positions. For example, *1 12 9 12 12* indicates that in the 12th grid there are three entities present. If at the next time interval i.e. at time T=3 we add one particular entity into the room through the POST request POST: *localhost:8080/robot/webapi/entities* <a>8</a> and send the GET request for fetching the positions of the entities at T=4 we would find the server would respond by the output of the form (a_1, a_2, a_3, a_4, a_5, a_6) where a_n denotes the grid position of the nth entity.

XML is used for the purpose of RPC (Remote Procedural Calls) supported by XML [11]. Parsing time for is high in XML when interconnection of client machines RPC was deemed to be fitter. Any two machines being present in different networks with different ip addresses may be interconnected with each other in order to act as the client interface. Thus the POST and DELETE requests that change the overall arrangement of entities in the room may be password protected and may be accessible from a different machine. In Table 2 we show the server responses at various tie intervals.

Table 1. Retrieval of Grid Information using HTTP Methods

HTTP Method	User Request	Server Response	Status report	Time
GET	/grids/1	<x>0</x> <y>0</y>	200 OK	139ms
GET	/entities	<a>4</a> <a>5</a> <a>6</a>	200 OK	99ms
GET	/entities/2	<a>15</a>	200 OK	48ms
POST	/entities	<a>8</a>	200 OK	101ms
POST	/entities	<a>8</a>	200 OK	101ms
POST	/entities	<a>9</a>	200 OK	106ms
PUT	/entities/2	<a>10</a>	200 OK	57ms
PUT	/entities/3	<a>10</a>	200 OK	58ms

Table 2. Retrieval of Grids with Obstacles

Time of Response (T)	User Request	Server Response
T=2	/webapi/obs	1 12 9 12 12
T=4	/webapi/obs	12 10 0 3 5
T=6	/webapi/obs	8 14 11 12 10

4.1. Cloud Server Simulation

We used Raspberry Pi 2, a card-based computer to create own cloud finally used to implement the cloud simulation. The host IP 192.168.0.101 is provided in with conjunction with output port 22. X11 forwarding was enabled in putty and the Pi was logged in and connection of Pi with the screen was created via tightvncserver in Pi and VNC Viewer in client machine. The VNC Viewer projected it on the laptop screen of the Raspberry Pi.

The Pi was configured for the cloud server by turning the overclock settings to medium and splitting the memory of the Pi to 16m. The SSH certificate was created using the command *sudo openssl req $@ -new -x509 -days 730 -nodes -out /etc/nginx/cert.pem -keyout /etc/nginx/cert.key*. The nginx web server was set up for the purpose of handling HTTP requests by using the command *sudo apt-get install nginx openssl* [10]. The web server configuration was altered using the command *sudo nano /etc/nginx/sites-available/default.* The Pi's ip was input into the server file of the pi and the pi was rebooted. After the reboot the Owncloud server was installed using the following commands:

sudo mkdir -p /var/www/owncloud

sudo wget https://download.owncloud.org/community/ ownc loud -9.1.0.tar.bz2

sudo tar xvf owncloud-9.1.0.tar.bz2

sudo mv owncloud/ /var/www/

sudo chown -R www-data:www-data /var/www

rm -rf owncloud owncloud-9.1.0.tar.bz2.

The drive used for storing the data was set mounted to the Pi in ntfs format. The hard disk drive was directly connected to the pi following which the drive was booted in the fstab file by the following command *sudo nano /etc/fstab*. The owncloud server was then logged into from the browser.

5. Implementing Security Mechanism

Basic authentication approach for securing web services involves sending user credentials over a secure socket layer (SSL) connection. The user credentials are stored on a server and when the user wants to log in he uses those credentials to identify himself and then continue the operation. The server gives back a hypertext transfer protocol (HTTP) 401 response if the credentials match [8]. The paper involves the examination of basic authentication in the client side of Robot based grid navigation through REST. Token based authentication is also used where the user puts his credentials before using the application. The server checks user credential provided with a token.

A basic authentication system has been implemented to retrieve information about the entities present in the room. The third party is prevented from using this feature because in that case it would spoil the integrity and confidentiality of the data. Before fetching information about the grids where the obstacles are present the user puts his username and password as configured in the web.xml file. If the credentials used by the user do not match the information stored in the server, the server returns a 401-error message signifying that the user is not authorized. The username and password has been allocated in the tomcat-users.xml file.

Once the user enters the valid credentials the server doesn't ask for further authentication in future transactions as the user is already authorized. The basic authentication has been implemented for the POST and PUT request. As for the GET request while determining which grids contain the obstacles we do not make use of basic authentication as the third-party user has every right to use this feature.

In token based authentication the user makes use of a token to send requests. Initially the credentials are matched with the information stored in the server after which the server issues a token that the user uses for further transaction.

5.1. Working of Token Based Authentication

In the system of robot based grid navigation system we make use of two passwords namely "R" and "S" to send to the server. Once the passwords are sent the server checks the password entered by the user to the passwords stored. If the passwords match the server returns a token that the user uses for further operations otherwise if there is a mismatch of passwords the server automatically closes. Each token has a lifetime of 10 seconds after which the token expires and no other transaction is possible.

5.1.1. Results of Token Based Authentication

The token based authentication security built for the grid navigation system made use of an array-list that stored the two passwords R and S. Individual password objects were created and then added to the array-list. The individual tokens that are provided by the server once the user logs in with the correct password are mentioned by default in a different file called *sampler.java* stored in a different location. Thus, if the password mentioned by the user does not match the password stored in the server the server automatically closes after a brief interval of 5 seconds, whereas if the token supplied by the server does not match the token input by the user access is denied for that instance.

The GET request to fetch the individual passwords are protected by the basic authentication system This is to prevent third party intervention and to preserve the confidentiality of the log in function. The basic authentication system that is set up in the *tomcatusers.xml* file makes use of the following piece of xml coding:

```
<role rolename="tomcat"/>
 <role rolename="role1"/>
<user username="tomcat" password="hello"
roles="tomcat"/>
<user username="both" password="hello"
roles="tomcat,role1"/>
<user username="role1" password="hello"
roles="role1"/>
```

Therefore, for the individual user to log in he must have a valid role as well as know the password to the system. With respect to basic authentication token based authentication provides a higher level of security.

Table 3. Access and Retrieval using token Based Authentication

Time of Query	Password Input	Token	Output
T=0	R	token	Access Granted
T=2	-	token	Access Granted
T=11	-	token	Access Denied
T=12	S	key	Access Granted
T=23	-	key	Access Denied
T=26	S	token	Access Granted

The above Table 3 tells us that at time T=0 the password inputted by the user was “R” and the server returned the token as “token”. This token remained valid till T=10 after which it became invalid, therefore when the user makes use of this at T=11 access is not granted. The user inputs the password “S” at T=12 and the server returns the token “key”. The same time constraint applies for this token as well.

Table 4. Results of wrong password input

Time of Query	Password Input	Token	Output
T=0	S	key	Access Granted
T=2	-	token	Access Denied
T=11	-	key	Access Denied
T=12	W	-	-

According to the above table when the user inputs a wrong password at T= 12 the server does not respond with a token but instead closes down after a brief interval of 5 seconds. Before T=12 the server functions properly as the user inputs the authentic password initially at T=0. At T=11 seconds when the lifetime of the token has expired the access is denied by the server.

The table below indicates a third case:

Table 5. Results depicting a third case

Time of Query	Password Input	Token	Output
T=0	S	key	Access Granted
T=2	-	token	Access Denied
T=7	R	token	Access Granted
T=13	-	token	Access Granted
T=17	-	token	Access Granted
T=25	W	-	Access Denied
T=26	R	key	Access Granted

The table above depicts a third case of token- based authentication approach in which the user initially inputs the correct password S to receive the ‘key” token from the server. The user then inputs a wrong token at T=2 where the access is denied to the client interface. At T=7 the user inputs a password to the server after which the server returns a new token ‘token’ and the access is granted to the user till T=17. At T=25 a wrong password is inputted by the user and the access is once again denied by the server.

6. Conclusions

This paper does not discuss the drawbacks of the token based authentication system implemented for the grid navigation system. No matter what password is inputted into the system the two tokens returned are “token” and “key”. A third party can easily identify the pattern or the two possibilities and disrupt the entire system. Therefore, a digital signature is required to maintain the confidentiality of the token so that only the authorized user can make use of it. To bolster the security issues, we can make use of Snort systems and Intrusion Detection Systems in Cloud Server Virtualization Monitoring. In the upcoming improvements, we hope to implement TOR networks and Snort IDS system in Cloud monitoring systems.

Acknowledgements

This paper was based on a project on Robot Based Grid Navigation System using Eclipse IDE and Jersey Framework.

References

[1] Snehal Mumbaikar, Puja Padiya “Web Services Based on SOAP and REST Principles”, International Journal of Scientific and Research Publications, Volume 3, Issue 5, May 2013, ISSN 2250-3153.

[2] Anil Dudhe, S.S Sherekar “Performance Analysis of SOAP and RESTful Mobile Web Services in Cloud Environment”, International Journal of Computer Applications (0975-8887), Second National Conference on Recent Trends in Information Security, GHRCE, Nagpur, India, Jan-2014. http://research.ijcaonline.org/rtinfosec/number1/rtinfosec1401.pdf.

[3] Tobias Fertig, Peter Braun “Model Driven testing of RESTful APIs” May 18-22, 2015, Florence Italy. http://www2015.wwwconference.org/documents/proceedings/companion/p1497.pdf.

[4] Announcing the Advanced Encryption Standard (AES). Federal Information Processing Standards Publication 197. United States National Institute of Standards and Technology (NIST). November 26, 2001. Retrieved October 2, 2012.

[5] Dominique Guinard, Iulia Ion, Simon Mayer. “In Search of an Internet of Things Service Architecture: REST or WS-*? A Developers’ Perspective, Institute of Pervasive Computing, ETH Zurich Switzerland.

[6] Carsten BUSCHMANN, Florian MULLER, Stefan FISCHER “Grid Based Navigation for Autonomous, Mobile Robots”, Institute of Operating Systems and Networks, Technical University of Braunschweig.

[7] Prithumit Deb, Nitin Singh, Saket Kumar, Nitish Rai, Dr. P. A. S Naidu, N.Ch.Sriman Narayana Iyengar “Offline Navigation System for Mobile Devices”, International Journal of Software Engineering and Applications(IJSEA), Vol1, No.2, April 2010. http://www.airccse.org/journal/ijsea/papers/0410ijsea3.pdf.

[8] Mohamed Ibrahim B, Mohamed Shanavas A R “Applying Security for RESTful Web Services- Limitations and Delimitations”, International Journal of Emerging Technology and Advanced Engineering, Volume 4, Issue 9, September 2014 ISSN: 2250-2459.

[9] Bhushan Jain, Mirza Basim Baig, Dongli Zhang, Donald E. Porter and Radu Sion, Stony Brook University “SoK: Introspections on Trust and Semantic Gaps” IEEE Security and Privacy Magazine 2015.

[10] Raja Mukhopadhyay, I Mukhopadhyay. “Home Automation and Grid Mapping Technology Using IoT”, The 7th IEEE Annual Information Technology. Electronics and Mobile Communication Conference 13th – 15th October 2016, Pages 1-5.

[11] Nurzhan Nurseitov, Michael Paulson, Randall Reynolds, Clemente Izurieta “Comparison of JSON and XML Data Interchange Formats: A Case Study”, Department of Computer Science, Montana State University, 2009.

[12] Raja Mukhopadhyay, I Mukhopadhyay “Data Encryption in Virtual Machine Transfer Over Cloud Network”, The 7th IEEE Annual Information Technology. Electronics and Mobile Communication Conference 13th – 15th October 2016.

[13] Ganesh Z Bhade, Vikrant Chole, “Review on Self-Destructive System for Data Privacy on Web Services”, International journal of Computer Science and Mobile Computing, Volume 4, Issue 3, March 2015. http://www.ijcsmc.com/docs/papers/March2015/V4I3201599b.pdf.

[14] Raja Mukhopadhyay, I Mukhopadhyay et.al., “Study On Secure Virtualization In The Cloud” published in IJACSCC Volume 4, Issue 1, May 2016, PP 13-17.