Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2017) 25: 2956 – 2967

doi:10.3906/elk-1608-6

Token-based authentication method for M2M platforms

Hüseyin POLAT*, Saadin OYUCU
Department of Computer Engineering, Faculty of Technology, Gazi University, Ankara, Turkey

Received: 01.08.2016 • **Accepted/Published Online:** 11.11.2016 • **Final Version:** 30.07.2017

Abstract: Nowadays the fields in which machine-to-machine (M2M) applications are used and the numbers of M2M devices and users are increasing gradually. In an M2M application, M2M platforms are used in order to follow and analyze the data presented by M2M devices. The communication of multiple users and devices via an M2M platform causes some problems in terms of security. In this study, an M2M platform has been developed by using RestFul web services and NoSQL database. On this platform a token-based authentication method was used for multiple users and devices. In this method, an authorized request approach was adopted for authorized users and an unauthorized request approach was adopted for unauthorized users. In the token-based authentication method on the M2M platform no session information is kept. Thanks to the adopted method, in multiple processes carried out on the platform, not only was data traffic density decreased, but also security level was increased for both user and device authentication.

Key words: Authentication, token-based authentication, M2M platforms, M2M authentication

1. Introduction

M2M refers to the technology of machine-to-machine or machine to human data transmission via mobile or other terminals. M2M platforms are the platforms developed on the basis of M2M technology in order to make the management of applications developed for various industries easier [1]. All management necessities such as remote control systems, measurement analyses, and reports are presented to users with the help of M2M platforms. By means of these platforms, the functionality of M2M applications increases.

An M2M platform is necessary for M2M applications. The M2M platform should be designed in such a way that it can provide service to different users and devices simultaneously and safely. It is quite difficult to maintain security in an M2M platform that enables multiple users to add multiple M2M devices. Authentications that are carried out using equipment such as a smart card, smartphone, or security token are neither recommended nor useful for M2M platforms.

In M2M platforms, not only people but also devices play an active role. Since M2M devices are also included in M2M platforms, using authentication applications such as fingerprint, retina reading, and vessel tracking is not possible. Thus, the authentication of users and devices demanding access to M2M platforms should be conducted by a completely different method. Therefore, token-based authentication is recommended for user and device authentication in M2M platforms.

M2M platforms conducted for specific applications and a few authentication studies applied in these platforms are available in the literature. In their study, Ren et al. mentioned four potential attack types for M2M systems and recommended group-based authentication [2]. Lai et al. studied the group Authentication

*Correspondence: polath@gazi.edu.tr

and Key Agreement (AKA) problem. They tried to decrease the signal blockage in authentication and lower the ID load. As a result, they suggested a lightweight authentication scheme for M2M in 3GPP networks [3]. Chen et al. suggested a Group Authentication and Key Agreement (G-AKA) for a home network. At this point, the protocols cannot ensure sufficient safety but in group communication authentication performance needs to be optimized. In their study, it was revealed that authentication for all devices at the same time is not possible [4]. Bae conducted a P2P service security protocol verification study in an M2M environment. He explained the security threats and requirements in M2M communication issues. A mutual authentication process was carried out using SessionKey and PublicKey on the security protocol recommended by the author [5].

Soliman et al. carried out a smart-home study by integrating web services and cloud computing technology into the Internet of items. In their study, during the process of data transmission from devices, ID information is continuously sent to M2M platforms [6]. Zhang et al. studied a group-based authentication protocol for machine-type communication conducted by using mobile networks. They produced tokens for devices belonging to the same group and shared these tokens for authentication [7]. Aubry et al. aimed to improve system security by disconnecting the central authentication server from the system in their authentication study carried out by using a central authentication service [8]. Needham and Schroeder suggested using encryption for authentication in mainframe networks. They conducted encryption and decryption studies on authentication servers [9]. Hwang and Li tried to improve authentication security by using a smart card in a system based on a password scheme. A user who wanted to log into the system had to use a smart card [10]. In a study of security and confidence survey for M2M systems, authentication mechanisms and encryption algorithms were mentioned. In that study IMEI (International Mobile Equipment Identity) number and IMSI (International Mobile Subscriber Identity) terminals were suggested for devices to be used in the background processes of M2M systems and also M2M platforms were suggested to incorporate sending update keys [11].

In the study by OneM2M about security applications on M2M, various methods from group authentication to mutual authentication were investigated. Mutual authentication was used in both symmetrical and asymmetrical systems. In this approach, a user who sends his identity logs into the system with his own ID info, and when confirmed users contact with the system they continuously and mutually send session data [12].

Regarding security issues, the technical documents published by ETSI (European Telecommunications Standards Institute) and OneM2M organizations can be used as a source. However, many studies were conducted on the structures that OpenMTC (Open Machine Type Communications) present their own sources and support different processes. On the other hand, many studies were carried out on cloud computing. We cannot claim that the cloud computing environment is an exactly safe environment to collect server data. When considered from this point of view, the cloud computing environment should be revised with a view to providing protection of privacy properties [13].

When the literature is investigated it can be seen that many authentications approaches developed for M2M are available [2–5]. Studies were conducted especially for authentication of multiple devices at the same time. In such authentication transactions that included groups of devices, blockage was observed. This problem can be eliminated through authentication of devices and users independently. Therefore, in this study by taking the OneM2M system architecture as a reference an M2M platform was developed and token-based authentication was conducted. In the M2M platform RestFuL web services supporting JSON structure and a nonrelational database were used. On this platform, a token-based authentication method was applied for multiple users and devices. For authentication of the devices unique ID keys were generated. The authentication transaction is carried out by sending these keys to the platform along with other information that describes the device.

Additionally, token-based authentication was set as obligatory for users who want to enter the device. Thus, it is aimed to improve the security level of M2M applications through token-based authentication, which is a different security approach.

2. Materials and methods

2.1. M2M architecture

The basic M2M architecture consists of three layers. In the first layer servers providing data for the M2M system are placed. The second layer is the communication layer, where data transmission occurs. The last layer is the application layer. In this layer, data coming from devices are followed and interpreted. The functional M2M architecture of ETSI is shown in the document OneM2M investigated M2M architectures as shown in Figure 1.

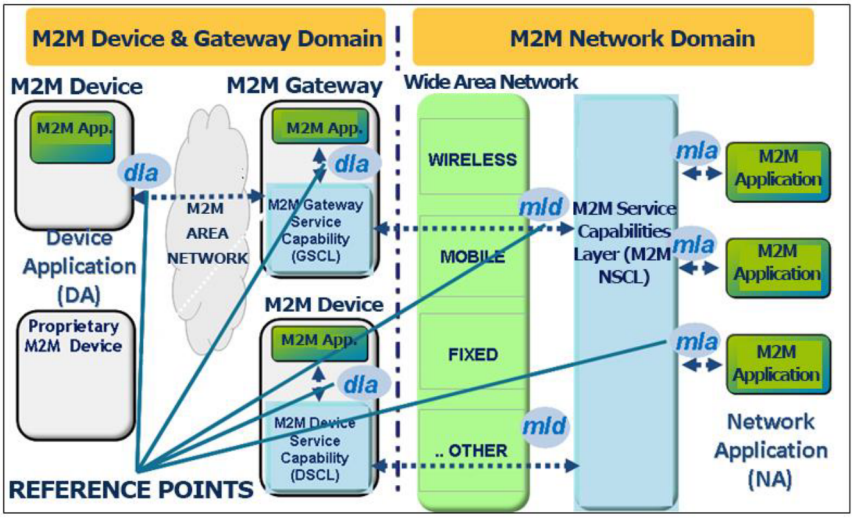

Figure 1. Functional M2M architecture of ETSI [14].

The architecture basically categorized as M2M device & gateway and M2M network domain is divided into different subdomains in each field. Basically, the information gathered from M2M devices is first transferred to the M2M area network. When the M2M devices want to contact the remote server the wide area network comes into operation. This study is based on Internet communication. Unlike basic architecture, in the structure in Figure 1 scalability was increased especially by putting the M2M service ability layer into use. Moreover, this made it easier to develop different devices and different applications. Requests coming from M2M applications and M2M devices are interpreted in the M2M service abilities layer and then the transaction is carried out. The system can easily be improved by adding the functions desired to expand current functions or add new ones to the service layer. This way of usage supported by oneM2M highly increases the scalability.

In this study by taking OneM2M architecture as the reference, an M2M platform architecture was developed. In Figure 2, the developed M2M platform architecture is shown.

As seen in Figure 2, in the field of web service clients a device or a user who can access any data transmission network in any way is available. Each request coming from clients is interpreted and evaluated and if needed transactions as writing or reading on the database is carried out in the web service layer. Web service clients make the requests as JSON type, the web service corresponds the requests as JSON, and typing is carried out as a JSON document. This process considerably increases the speed. The client does not have

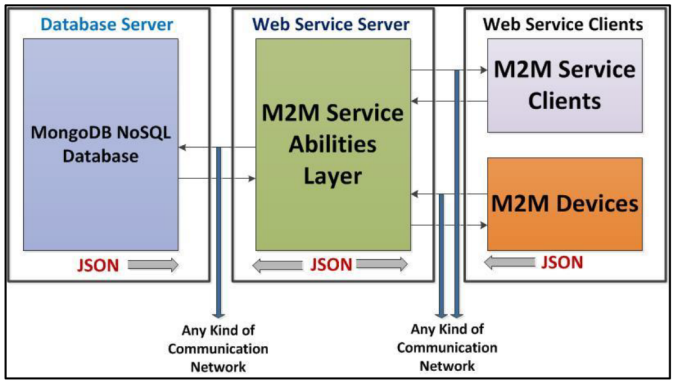

Figure 2. M2M platform architecture.

to know about any sources such as databases. This is a feature of the RestFul structure. The client makes a request and appropriate responses are prepared and sent back to the client. Among security studies each field shown in Figure 2 can be kept in different places and the security level can be increased by studying network security. However, this increases the cost. Hence, improving security processes on M2M service abilities was preferred and the studies are designed in this way.

2.2. M2M platform web service architecture

When M2M architecture is investigated, it is seen that the M2M service abilities layer has importance. A large part of the work will be carried out through service abilities. REST (Representational State Transfer) architecture, a web service architecture, is basically like SOAP (Simple Object Access protocol); however, it has a more understandable structure than SOAP. RestFul structure is not based on any technology. It can support XML, JSON, or BSON messaging structure. In RestFul architecture four main operations of HTTP (Hyper-Text Transfer Protocol) are used. These are GET/POST/PUT and DELETE [15].

In this study SOA (Service Oriented Architecture) as architectural and RestFul web services as a web service approach are chosen. Because JSON structure is supported by the RestFul structure, which is a more favorable structure than XML, the usage of JSON is simple and its size is small. Moreover, in layered architecture one of the six features Fielding [16] stated RestFul with its features as caching, server-client structure, and not storing session information in status info provides an advantage. There are many structures to build up RestFul web services. Among these, Spring is the most important. Spring provides a comprehensive configuration model to improve Java-oriented programming and applications. Spring was improved by Johnson, and it was published under the license of Apache in 2003 for the first time [17]. Spring 3.0 supports Rest architecture [18].

Spring basically adopted two different methods to make software developers' work easier and to increase the portability of developed software. Because improving a project with the Maven method is easy, this study was started as a Maven project. The software development environment should be prepared successfully. The software necessities of a project carried out through a Windows operating system are as follows:

- Eclipse Kepler
- Java Standard Edition Version 8
- Apache Tomcat 7.0

Eclipse Kepler version is quite successful in terms of different toolbars and supporting Maven projects. Java version 8 functions with Spring version 3.0 and 4.0, and supports NoSQL connection structures. Apache Tomcat is a server structure that proved itself and is easy to use.

2.3. Identification of services

In RestFul architecture used in M2M service identifications, request and responses are carried out in a specific hierarchy and order. This order not only decreases the complexity of the systems but also enhances the performance. The Spring MVC RestFul architecture used in developing services is shown in Figure 3.

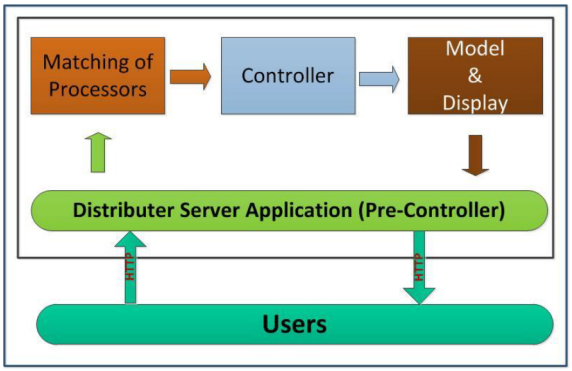

Figure 3. Spring MVC ResfFul web service architecture.

The architecture in Figure 3 meets the requests coming by way of HTTP in the controller and distributes the requests to the necessary points. During the distribution process requests are transferred to appropriate controllers by matching processors. If controllers do not have any error, to compose responses for the requests models and views is arranged. The arranged responses are presented again to the user through distributer server application. Each request and each response are done through HTTP. In Figure 4, the transformation stage of the architecture used into software is shown.

Rest architecture, which works according to the request–response principle, was transferred to the software as seen in Figure 4. According to the figure the requests taken by way of HTTP are interpreted in the pre-controller and processors are matched. Thus appropriate requests are sent to the controllers. The identification of requests and responses is stated in Number 1 software packages seen in Figure 4. The identification of controllers is stated in the controllers' package shown in number 3 of Figure 4. Controllers can make a request about typing and reading to the database while preparing responses. In this case, in order not to face a conflict, work layers were developed and a layered structure was formed in the software. In the work layers shown in number 2 of Figure 4, the tasks requested by controllers or that need to be done by them are carried out. Database processes are carried out in the packages identified in number 5. Models and views are related to how responses will be prepared and presented. In Figure 4 besides models and views, processes in which data conversion tasks are done are shown in packages 4 and 7. This is because in some cases data are converted to specific format and sent to the user. Moreover, package number 6 is the package in which error identifications are done, and when faced with an error, in this package there are error messages and status messages to be presented to the user. According to Spring, status messages are as follows [19]:

- 1XX - Informing
- 2XX - Success

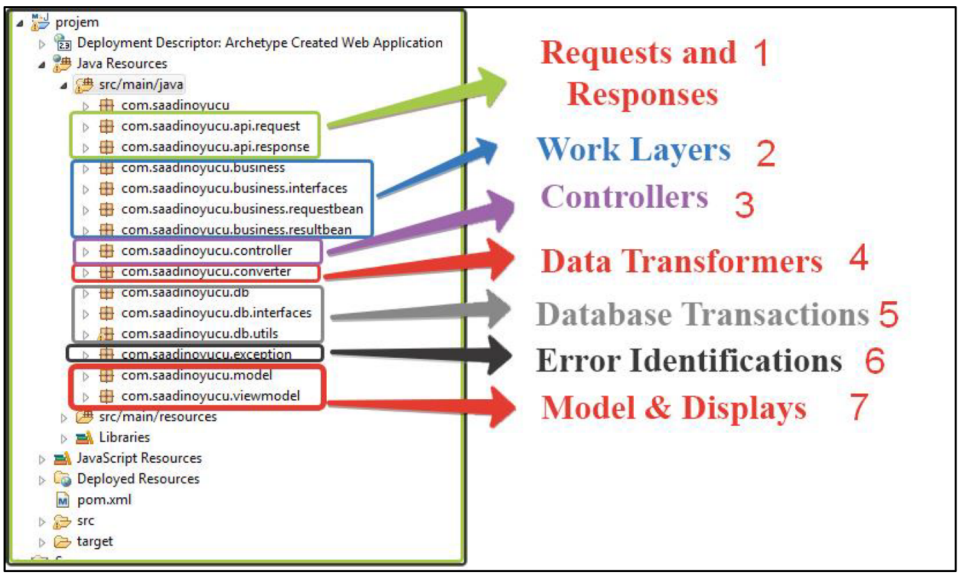

Figure 4. The transformation of RestFul web service architecture into software.

- 3XX - Guidance
- 4XX - Client Error
- 5XX - Server Error

Requests can be received from clients through GET or POST. However, in terms of reliability and continuity, using GET causes some congestion in the system. The reason is that the client continuously sends data through URL and after a short time this is prevented by the security wall and seen as a virus. GET provides accessibility in the small-scale places and the places where data flow is not so much. Since there is a continuous data flow from the clients to our system, POST is used in all processes. When each POSTed request gets a 200 OK status message, it is understood that the system is running successfully. All requests and responses are done in JSON type.

3. ID authentication

The most important problem in web-based applications is ID authentication. There are different solutions for this problem. In a classical web-based application session information can be stored by web server. For example, in a classical PHP-based application the stages of ID check and session control carried out in the web server are as follows (Figure 5).

Firstly, during the process of session control, the message of Authentication is Necessary is sent to the client web browser by the server, and then the username and password screen is opened. When username and password are entered, URL including PHP script is called again through username, password and predefined variables assigning the authentication type. These predefined variables are checked in any processes and information is stored in the server. To improve the security further, in order to get over SQL injection attacks, processes are conducted by checking login form structure [20]. However, in some cases this process is inadequate on its own. Session information stored in the server also falls short in a structure with multiple users. Thus, session information stored in a web server is not an appropriate choice for M2M applications.

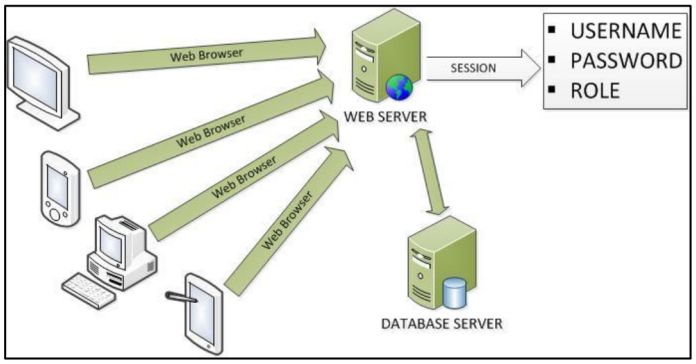

Figure 5. PHP server session control.

In the M2M platform developed within the scope of this study, session control and ID authentication are conducted through a token-based method. Therefore, when each registered user has a login request with their username and ID, the M2M service abilities layer checks the database and if the user registration is confirmed, it gives a unique and valid-for-one-hour token and user ID info as an answer. If the user wants to make any transaction, he has to send his user ID and token info to the M2M service abilities layer. The web service layer examines user ID and token info for each request and if confirmed the request is evaluated. If the token is expired the user is requested to re-log in. On the other hand, if the token or user ID is incorrect the user is notified of the incorrect transaction. Figure 6 shows this situation.

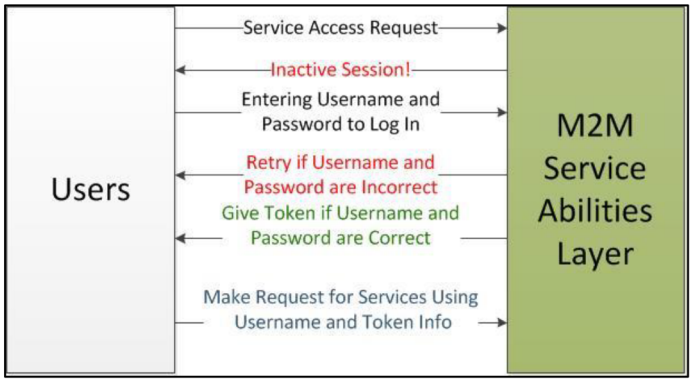

Figure 6. Token-based ID authentication.

As shown in Figure 6 in each M2M service abilities request tokens are generated for the client. On the other hand, in Figure 7 the generation of tokens and how user ID authentication is done are shown.

As is clear in Figure 7 service M2M abilities layer that receives a request to log in searches data from the database and if correct data are found confirmation is sent to the service ability layer. For the next step at a different collection of the database token data are generated and the validity period of the token is reported to the system. User-specific and unique tokens are given to the user. As a result, the token that will be used in any request for the M2M service abilities layer is generated. If the token info is incorrect or expired requests are rejected and the user is requested to re-log in.

Not only users but also M2M devices have access to M2M platforms. In this case, the data that will be continuously POSTed to the M2M service abilities layer can make a request sending username and password at

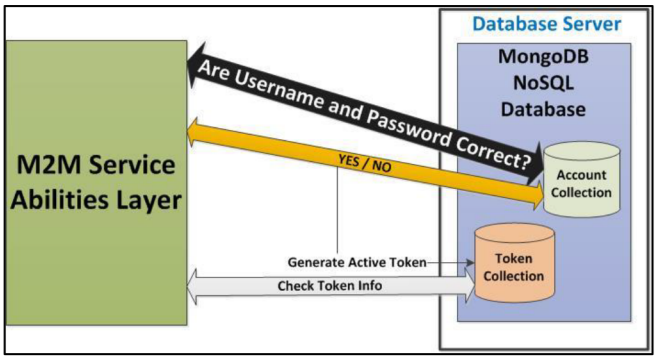

Figure 7. Token generation and check.

the same time. However, in such a situation it is inevitable that the username and password will be captured by third parties. The user who logs into the M2M platform can be permitted to add a device over the platform and the data that need to be added in code snippet can be given. In this case, the data that need to be given to the user are unique and user-specific ID info; user- and device-specific device ID info, unique username; and token data that are generated according to password and device info. In an M2M device this information is added to the data coming from receptors and POSTed to the M2M service abilities layer. The M2M service abilities layer first checks user ID and token info of the requests coming from the M2M device. If the token info and ID info match the database the writing transaction is done. The writing transaction is done according to the user ID and device ID. This is because more than one device can be identified in one user ID. When it comes to the queries the information is listed for the user according to the user ID and device.

Application of such a security check and the successful result of this check decrease the vulnerability of HTTP transactions. This is because the session data are not stored in any place, and requests are realized only through valid tokens. Additionally, doing database transactions through web service layers increases database security, because in Rest architecture data of the sources are not stored; as a result of this, database information is not shared with the user. The code snippet developed for generating and checking tokens can be seen in Table 1 among the software codes.

The tokens that are generated randomly and unique for each user who logs in are developed through Class UUID, which comes with Java version 7. This system type generates 128-bit series composed of number, letter, and characters. For each generated token a validity period is assigned and all of the offline users are discarded from the platform. It is visible in Table 1 that a token-for-one-hour is formed through expire.add (Calendar.HOUR, 1) code. In checkAuth function the token data are checked and true or false data are generated. If the information is correct the below code snippet starts operating. Thus the ID check is done in the M2M platform; the requests are separated as authorized and unauthorized requests. This check should be done for each service that will be added to M2M service abilities. In this way, ID authentication will be done for all requests.

4. Results and discussion

In the developed M2M platform there is a need to apply a security check for each of the web services in the service capability layer. The services can be tested with the help of developing a new client or by means of available clients. For testing, Advanced Rest Client, which is a Google Chrome plug-in, is used. Table 2 shows what a user sees when trying to log into the M2M platform.

Table 1. Token generation and check.

Token Generation

String userName = requestBean.getUserName();
String password = requestBean.getPassword();
Account account = accountDBService.login(userName, password);
if (account == null) {
validAccount.setIsAuth(false);
return validAccount;
}
Token objectToSave=new Token();
objectToSave.setCreatedDate(new Date());
Calendar expire= Calendar.getInstance();
expire.add(Calendar.HOUR, 1);
objectToSave.setExpireDate(expire.getTime());
objectToSave.setToken(UUID.randomUUID().toString());
objectToSave.setUserID(account.getId());
tokenDBService.save(objectToSave);
validAccount.setIsAuth(true);
validAccount.setToken(objectToSave.getToken());
validAccount.setUserId(objectToSave.getUserID());
return validAccount;
}

Token Check

Boolean checkAuth = tokenDBService.checkAuth(apiRequest.getToken(), apiRequest.getUserId());
if (checkAuth == false) {
apiResponse.setExceptionCode(101);
apiResponse.setHasException(true);
apiResponse.setValidatormessage("Auth Error");
return apiResponse;
}
….
return apiResponse;
}

As is obvious in Table 2, userId and token are given as a response to the user that sends a correct request. The user is obliged to use this information in his request for any service. This is shown in Table 3 in detail in the new device adding service.

As shown in Table 3, it is impossible to request login through incorrect or expired tokens. When Table 3 is carefully examined it can be seen that the user is warned through exception codes, exceptional case, and a confirmatory message for this exceptional case if any exceptional case occurred in the system. When a user makes an incorrect request, with the help of 101 code and Auth Error message, he understands that the ID information sent was incorrect. The user, who will log in again with username and password, will be able to make a request through newly generated token data.

It is clear that device ID data are also included in the values that will be presented as an answer to the user who makes a request to add a device. The user can make a request from the M2M platform through embedding these data in his M2M device. The incorrect request will not be accepted as usual. In order for one M2M device to contact with the M2M platform the user settings need to be installed in the device. At the

Table 2. Login process ID check.

Service Address: http://localhost:8080/saadin/service/device/CihazAdd			
Correct Request	Response to Correct Request	Incorrect Request	Response to Incorrect Request
{ "deviceName" : "m2m device 1", "token" : "f7868999-e580-446a-ae79-88e985685b7f", "userId" : "5548b5cb88e5c9a523940a88" }	Status 200 OK { exceptionCode: 0 hasException: false deviceID: "c9a38c16-5c99-4569-8a25-4ceaa98a2a2c" deviceName: "m2m device 1" userId: "5548b5cb88e5c9a523940a88" validatormessage: null }	{ " deviceName" : "m2m device 1", "token" : "f7868999-e580-446a-ae79-88e985-FALSE", "userId" : "5548b5cb88e5c9a523940a88" }	Status 200 OK { exceptionCode: 101 hasException: true device ID: null deviceName: null userId: null validatormessage: "Auth Error" }

Table 3. Authorized and unauthorized requests for services.

Service Address: http://localhost:8080/saadin/service/device/CihazAdd			
Correct Request	Response to Correct Request	Incorrect Request	Response to Incorrect Request
{ "deviceName" : "m2m device 1", "token" : "f7868999-e580-446a-ae79-88e985685b7f", "userId" : "5548b5cb88e5c9a523940a88" }	Status 200 OK { exceptionCode: 0 hasException: false deviceID: "c9a38c16-5c99-4569-8a25-4ceaa98a2a2c" deviceName: "m2m device 1" userId: "5548b5cb88e5c9a523940a88" validatormessage: null }	{ " deviceName" : "m2m device 1", "token" : "f7868999-e580-446a-ae79-88e985-FALSE", "userId" : "5548b5cb88e5c9a523940a88" }	Status 200 OK { exceptionCode: 101 hasException: true device ID: null deviceName: null userId: null validatormessage: "Auth Error" }

end of correct transactions, the devices will start communicating with the M2M platform, which is shown in Figure 8.

As is clear in Figure 8 the M2M device that can make the POST transaction will properly communicate at the end of the user settings that will be done on the M2M device. An Arduino Uno smart card is used as an example. The important point is the validity period of token data that will be valid for M2M devices. This criterion is left to the developers since different system developers may develop different scenarios.

In the present study basically a NoSQL database that can store information as a JSON document was used. At the same time an M2M platform is developed by using RestFul services that support JSON messaging structure. The M2M platform developed in this study has the capacity to support both multiple users and multiple M2M devices transactions at the same time. On the M2M platform a token-based ID control system based on the request–response principle and can discriminate between authorized/unauthorized requests was

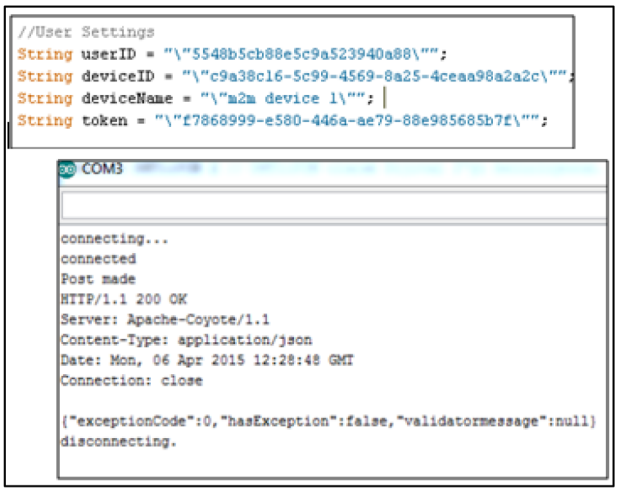

Figure 8. Communication between M2M device and M2M platform.

successfully applied. In this way, no session information was stored on the server. As a result of this in multiple transactions the server load was decreased. Additionally, through incorporating M2M devices, ID control for the devices was also ensured.

Since the RestFul architectures communicate over HTTP, a security gap would be inevitable. However, while web services were being developed, token-based ID control was incorporated so that security level was enhanced to a higher level. Moreover, the use of POST mode, instead of GET mode, is another factor that raised the security level. In the web service tests phase of the study it was observed that the token-based ID control system was functioning properly. In addition, the layered structure used in the development of the services made it easier to fix the possible security gaps. The features such as the layered structure of the general system architecture, and the possibility of storing each layer on a separate server and in different places raise the opportunity to support each layer's security in terms of both hardware and software.

5. Conclusion

Nowadays M2M applications have become a part of modern life and occupy a fairly widespread area of use. The M2M applications developed without conforming to standards and planning have given birth to new security flaws. Therefore, attacks on M2M servers and devices have been increasing. Most of these attacks are conducted deactivating the weak authentication algorithms and the common authentication transactions. Therefore, in this study the authentication needs for M2M applications were explained. An M2M platform was developed in compliance with the ETSI M2M reference architecture and token-based authentication was applied. Thanks to the token-based authentication each user and device added to the M2M platform was enabled with authentication.

This study discussed the authentication procedures necessary for increasing security in M2M applications. However, authentication on its own is not enough to ensure the security of a system. The system developed in this study shed light on different fields of study in terms of security. Coding the token information through certain algorithms, studying network security for each layer, and development of database systems with different methods are some of those different fields.

References

[1] Yang Y, Ye H, Fei S. Design of communication interface for M2M-based positioning and monitoring system. In: 13th Annual Conference of IEEE Electronics, Communications and Control (ICECC 2011); September 2011; Ningbo, China: IEEE. pp. 2624-2627.

[2] Ren W, Yu L, Ma L, Ren Y. How to authenticate a device? Formal authentication models for M2M communications defending against ghost compromising attack. Int J Distrib Sens N 2013; 9-2.

[3] Lai C, Lu R, Zheng D, Li H, Shen X. S. GLARM: Group-based lightweight authentication scheme for resource-constrained machine to machine communications. Comput Netw 2016; 98: 66-81.

[4] Chen YW, Wang JT, Chi KH, Tseng CC. Group-based authentication and key agreement. Wireless Pers Commun 2012; 62: 965-979.

[5] Bae WS. Designing and verifying a P2P service security protocol in M2M environment. P2P Networking and Applications 2016; 9: 539-545.

[6] Soliman M, Abiodun T, Hamouda T, Zhou J, Lung C. H. Smart home: integrating internet of things with web services and cloud computing. In: 6th Annual Conference of IEEE Cloud Computing Technology and Science (CloudCom 2013): December 2013; Bristol, UK: IEEE. pp. 317-320.

[7] Zhang Y, Chen J, Li H, Zhang W, Cao J, Lai C. Dynamic group based authentication protocol for machine type communications. In: 4th Annual Conference of IEEE Intelligent Networking and Collaborative Systems (INCoS 2012); September 2012; Bucharest, Romania: IEEE. pp. 334-341.

[8] Aubry P, Mathieu V, Marchal J. ESUP-Portail: open source single sign-on with CAS (central authentication service). Actes of EUNIS 2004; 172-178.

[9] Needham RM, Schroeder MD. Using encryption for authentication in large networks of computers. Commun Acm 1978; 21: 993-999.

[10] Hwang MS, Li LH. A new remote user authentication scheme using smart cards. IEEE T Consum Electr 2000; 46: 28-30.

[11] Chen H, Fu Z, Zhang D. Security and trust research in M2M system, In: IEEE 2011 International Conference on Vehicular Electronics and Safety (ICVES 2011); July 2011; Beijing, China: IEEE. pp. 286-290.

[12] OneM2M, Security Solutions. Document Number; TS-M2M-0003v1.0.1, Technical Report 2015.

[13] Suciu G, Vulpe A, Halunga S, Fratu O, Todoran G, Suciu V. Smart cities built on resilient cloud computing and secure internet of things. In: 19th Annual Conference of IEEE Control Systems and Computer Science (CSCS 2013); May 2013; Bucharest, Romania: IEEE. pp. 513-518.

[14] OneM2M, Analysis of architectures proposed for consideration by oneM2M. Document Number; OneM2MTR-0002-Architecture Analysis Part 1-V-0.2.0, Technical Report 2013.

[15] Alpay O, Hatem V, Sunel S. Restful Adaptor for JMX. In: 5th Annual Conference of National Software Engineering Symposium; September 2011; Ankara, Turkey: pp. 300-305.

[16] Fielding RT. Architectural styles and the design of network-based software architectures. PhD, University of California, Irvine, USA, 2000.

[17] Deinum M, Serneels K, Yates C, Ladd S, Vanfleteren C. Pro Spring MVC: With Web Flow. New York, NY, USA: Apress, 2012.

[18] Johnson R, Hoeller J, Donald K, Sampaleanu C, Harrop R, Risberg T, Templier T. The Spring Framework–Reference Documentation Interface. 2012.

[19] Richardson L, Ruby S. RESTful Web Services: Web services for the real world. Sebastopol, CA, USA: O'Reilly, 2007.

[20] Salleh MK, Zaini NM. Web-based adaptive audio-therapy recommender system. In: 5th Annual Conference of IEEE Control and System Graduate Research Colloquium (ICSGRC 2012); July 2012; Shah Alam, Selangor, Malaysia: IEEE. pp. 231-236.