Central Audit Logging Mechanism in Personal Data Web Services

Abdulsamet HAŞILOĞLU
Department of Computer Engineering, Atatürk University
Erzurum, Turkey
asameth@gmail.com

Abdulkadir BALI
Ataturk University, Institute of Science and Technology,
Computer Engineering
Erzurum,Turkey
abdulkadir.bali@tuik.gov.tr

Abstract— **Personal data have been compiled and harnessed by a great number of establishments to execute their legal activities. Establishments are legally bound to maintain the confidentiality and security of personal data. Hence it is a requirement to provide access logs for the personal information. Depending on the needs and capacity, personal data can be opened to the users via platforms such as file system, database and web service. Web service platform is a popular alternative since it is autonomous and can isolate the data source from the user. In this paper, the way to log personal data accessed via web service method has been discussed. As an alternative to classical method in which logs were recorded and saved by client applications, a different mechanism of forming a central audit log with API manager has been investigated. By forging a model policy to exemplify central logging method, its advantages and disadvantages have been explored. It has been concluded in the end that this model could be employed in centrally recording audit logs.**

Keywords— Web Service; API; API Policy; Audit Logging; Personal Data.

I. INTRODUCTION

Establishments and enterprises are legally bound to maintain the confidentiality and security of personal data collected either from the individuals themselves or relevant establishments and enterprises to execute their current activities [1]. In the case of utilizing as a legal evidence in any given administrative or legal investigation and to maintain accountability, it is an emerging requirement to provide access logs for the information on personal records. These logs can be either central or application-based records. In this paper the way to keep this access log in a centered and effective manner and in the event of sharing personal data with web service, standard mechanism of forming a central audit log with API manager have been explored. Minimum required data to be saved in this log has also been analyzed in this paper. Lastly a logging method based on API manager policies has been suggested.

sIn any web service request, access information and inquiry parameters are transmitted to the server application within the same message [2]. In a general view Web service can be accessed by users, databases, applications and computers. To distinguish these objects that can provide access to a Web service, it is essential to symbolize them with unique identifiers. Directory services are the kind of solutions generated to meet such needs in organizational level [3]. To manage the control of access authorities, API manager can consult to a directory service that is either a component of the manager or organization in policies involving an analysis of requests.

II. WEB SERVICE ACCESS POLICIES

In the case of accessing personal data via web service it is expected that the request will be executed by a client application that has been activated by a user or a scheduler. In accordance with that, web service access policies are divided into two categories as policies that cover both users and applications or only the applications.

In the policy where users can achieve access via an application, the procurement of results by client application alone is not adequate on its own; it is also mandated to check user information. Computers that this application is operated on and computer that the user logged in can also be integrated to computer access policy.

Client applications operating under the control of scheduler are used to automatically transfer data received from web service to a certain database or a location. Since such applications are not dependent on a user, their access policies do not cover users. Here too, computers that operate this application can be integrated to the access policy.

The primary task of client applications could be more comprehensive than just consuming a unique web service. On the other hand, in API manager policies all applications that consume a web service are treated as a client of this service.

Web service is based on well-known protocols and standards such as SOAP, WSDL and HTTP [4]. Depending on being a SOAP or REST type of Web service, access information can be sent to the server in Custom SOAP header or HTTP header. In some sources, it is recommended that the

978-1-5386-3449-3/18/$31.00 ©2018 IEEE

HTTP header be used for information purposes [5]. But, that does not mean that the opposite is totally useless. In this approach a new layer is added in which information for identity confirmation can be transmitted without making any changes in description files or in the unique composition of messages.

Attribution of a universal unique ID of each transmitted request in API manager policy would ease the process of tracking logs. To save in their own logs this ID can be transformed into a client application by integrating to HTTP header of the Response.

III. INFORMATION TO BE LOGGED

Rather than logging everything, it is suggested to choose minimum essential information for the logging stage based on performance, confidentiality and legal conformity level. Logs should be saved as hash stamp and log access should be bound to an administrative and legal procedure.

In the event that a request on personal data query is submitted to API policy, this information can be logged. Logging such information could be for the benefit of users and programmers. If final users can access the web service through the menus of client application, the aim of query can be coded within the application itself.

For a web service in which personal data were inquired, classifying the information to be logged could assist in policy formation. A simple classification can be such: user, client application, computer, request, routing (routing), response and time information. Minimum quantity of components to log from the information in these parts are as explained below.

User log component: For the User, it should be a unique identifier that is valid outside the organization and it has to be directly used in official dealings as well. Some examples of such identifiers are user's identity number, passport number or credit card number.

Client application log component: User is expected to access the web service via an application. There are also certain client applications that can work automatically without a user. Another variable to log is which client application has been used. As seen in users, applications are also described in directory service and it is enabled to receive an access control in API policies.

Computer log component: Information such as the name of the computer owned by the User or client application, data such as IP and MAC addresses belong to this category. Application could be working in the user's computer or a server. To transfer this information on the network to API manager, it is required to install proxy settings.

Request log component: Except password, names of the variables such as URL where the request was sent from and web service name, operation name and operation parameters and attributed values to such variables are such components. Since the names and numbers of parameters are likely to be changed depending on the type of operation, if they are logged as keyname-value pairs it becomes more feasible to genericize API Policies. Universal IDs submitted to the requests within the policy can be recognized as request information.

Routing log component: It is the unique URL address of the operation of the web service server application where the request will be routed. This is the information unknown to User and client application but present in API policy.

Response log component: Since logging is a duplication process in essence it should be taken into account that logging response parts that bear personal data could lead to certain administrative and legal issues. It may be essential to learn if a response from sent address of the Request has been taken or not. For the routed request, HTTP status code information of the sent response could be utilized for that purpose. HTTP status codes are extensible [6].

Time log component: It is time related information about receiving of the Request by the API manager from client application, routing to a server application and sending of a response from server application. Since Web service can be called as a multi thread or from a program circuit, if time information is saved in the most precise manner in the type of a time stamp, it becomes much easier to achieve a more accurate chronological order of logs. Any delays in policy and messages can be detected via using this time information.

Policies can be ordered in a way to save an error log that can record logs belonging to faulty requests. Since a myriad of reasons may be attributed as the cause of error it is, in most cases, infeasible to be recorded in a standard log structure. For instance, it is likely that a nonexistent operation may be called or a request outside of web service may be processed. For the faulty requests the policy can transform apply to the client either SOAP Error or HTTP Status information depending on their applicability. It is expected that a good policy design would refrain from making unnecessary routings for faulty requests.

A web services architecture should have its own logging mechanism [7]. Logs can be recorded in an external database or a professional log system. Against the risk of any malfunction or disconnection in the system alternative logging methods such as recording on a local database or file should be specified in the policy. Such logs should be periodically transmitted to the unique log system.

Log records are utilized not only for administrative and legal issues but for the statistic generation and debugging purposes as well. Logging system should be qualified enough

to swiftly process requests such as filtering and listing of the records.

IV. RESULTS AND DISCUSSION

In accordance with the logic of central policy depicted in this paper, in Figure 1, a model API policy flow has been designed. For each new request sent to a web service identified in API manager, this flow is operated independently from one another. In the 1st step a time stamp of minimum in milliseconds level is generated to identify the received time of a new request. With the help of time stamp, requests can be chronologically ordered in logs. In the 2nd step, a universal and unique ID number is attributed to all requests. This ID would function as the fingerprint of request on logs. API managers possess all the tools that can generate time stamp and universal unique ID. In the 3rd step, IP address from where the request was taken is obtained. In the 4th step existence of the application expected to access from that IP address is questioned. If the application is detected it is authenticated in the 6th step. After authenticating the application, then user authentication is actualized in the 8th step. In the steps ranging from 9 to 14, name of the operation, essential and nonessential parameters are distinguished respectively. In the 15th and 16th steps this request is redirected to application server. In the 17th and 18th steps HTTP status code is obtained which respectively shows received date of the response sent from application server by the API Manager and the way the sent parameters are utilized. Those that achieve requesting from the client application are logged in the 19th step and upon confirming the actualization of logging process in the 21st step a positive response is directed to the client application. For the requests that failed in the 5, 7, 9, 11, 13 and 20 th control steps of the policy, an error message is prepared in the 21st step and after logging it in the 23rd step, it is directed to client application.

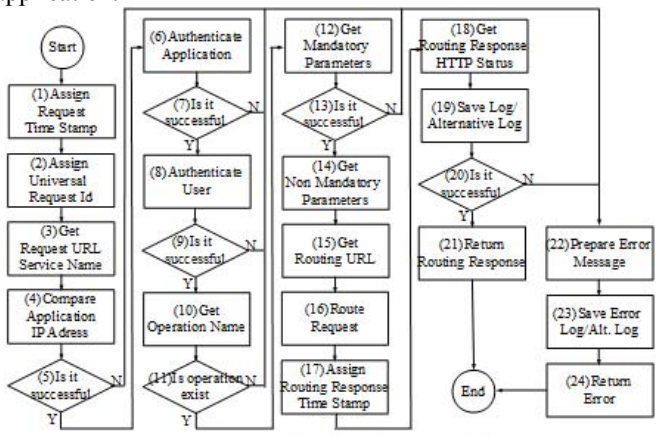

Fig. 1. A sample API policy flow for personal data service

Some variables such as the name of client application or user and the reason of query needed by API policies would have to be obtained from SOAP header or HTTP header. On the other hand integrating such headers to request message may create extra load in client applications. To make this process simpler, it is suggested to devise a programming library

Since in API managers logging of the requests is a centrally executed process, any demands for change like starting to use a different log system or adding new variables to logs can be actualized only by updating the policy identified in API manager. In the method where logs are recorded by client applications it is essential to make updates with all clients. Clients could have been programmed in dissimilar application languages or been programmed by an external service provider. Updating and testing more than one application rises the emerging costs.

Since clients use autonomous (independent) system sources performance would rise if audit logs were recorded by client applications. By boosting system sources of API managers, performance can be elevated. Finally, API policies can be deployed to centrally record audit logs into all web services including central personal data services.

REFERENCES

[1] Regulation, Law on The Protection of Personal Data, Turkey, Regulation No:6698, 2016.

[2] C. Geuer-Pollmann and J. Claessens. Web Services and Web Service Security Standards. Information Security Technical Report, 10(1):15-24, 2005.

[3] T. Howes, M. Smith, G. S. Good, "Understanding and deploying LDAP Directory Services" in , Indianapolis, Indiana:Macmillan Technical Publishing, 1999.

[4] B. Hollunder, "WS-Policy: On conditional and custom assertions", ICWS'09: Proceedings of the 2009 IEEE International Conference on Web Services. IEEE Computer Society, pp. 936-943, 2009.

[5] M. Masse, REST API Design Rulebook, O'Reilly, 2012.

[6] Hypertext Transfer Protocol HTTP/1.1 RFC 2616 Fielding, et al.

[7] A. Chuvakin and G. Peterson. Logging in the age of web services. IEEE Security and Privacy 7(3):82–85, 2009