Optimising API Signature Validation using Timestamp Grouping

Akash Dinakar Pathade
Department of Computer Science and Engineering
D. Y. Patil College of Engineering & Technology, Kasaba Bawada, Kolhapur, India

Abstract— API (Application Programming Interface) is the best mechanism used for exchanging information between applications. Because of security issues involved in it there are also various standards available for securing data during transmission like Signature. However even if timestamp based signature provides good security it consumes high amount of resources in server side to validate and authenticate request. So here I am providing solution to optimise use of resources in server.

Keywords: API, Signature

I. Introduction

Application Programming Interface (API) is the communication link between applications developed or running on two different environment or platform. It provides access of information, resources or any other entity served by one computer, mostly called as server to the requesting computer called client. To provide this resources requesting user must have registered with server and get his own API secret key to access resource. This secret key is used by requesting machine with POST or GET request to access that information. POST and GET are methods used by HTTP protocol to reach the server. To provide security from eavesdroppers signature is used. Signature is special hash generated during each request to server. Normally signature is combination of (requesting information + shared secret + timestamp) hashed using any well-known hashing algorithm like md5, sha256 etc. Hash is special function which takes any file or string as argument and generate an alphanumerical string which is not reversible. Shared secret is string known by server and client and is implanted in program during development. Timestamp is UNIX system time in seconds, i.e. seconds passed from 1 January 1970. This ensures that new hash is generated for every request. To verify this signature, server runs the same function as discussed above with same shared secret but with decrementing timestamp, because as we know any request needs some time to reach its destination, so server have to check for some N seconds earlier to validate signature. This requires server to run same function N times, which increases resource uses and reduces server performance.

II. Problem Statement

To optimise time needed to validate a single API request signature and utilise resources and increasing server performance

III. Objectives

1) To reduce calculations or computations needed to validate each API request signature
2) To minimise time needed to validate each request
3) To optimise server resources needed for request validation

IV. System

Timestamp Grouping is technique in which some milliseconds or seconds of time is grouped together. So any time falling in group x will imply same operation or meaning. Example if t1, t2 and t3 are in group x then the independent meaning or operation of t2 will be same as another members, t1 and t3 in that group. Proposed system consist of server, client, variables, equations, functions as discussed below.

A. Variables

– API key[1]: The unique alphanumerical string to identify each API user
– shared secret[2]: Randomly generated string shared by both to sign and validate request
– timestamp: representation of time in UNIX system, it is seconds passed from 1 January 1970
– requesting information: These are parameters or file name in URL for which client is requesting
– x[1,2] : are variable or values decided by API server and shared with client during development.
– Some runtime variables may be generated by server or client according to their need like DataToSign

B. Equations

Following equations or functions used by clients and server to sign request or to validate signature respectively. For optimising I will use ‘mod 5’ as timestamp grouping for signing request i.e. (timestamp mod 5) should return 0, else timestamp will be incremented by (5-reminder value) and hash is calculated.

C. Client

Client is the machine which request for information or resources from server using API key and signature. To generate API signature key following procedure is implemented. I am taking mod 5 as grouping function.

```
//temporary variables
DataToSign = (requesting information + shared secret)
timestamp = Current timestamp of client in milliseconds
//function – timestamp group and sign generate
if (timestamp mod 5 = 0)
signature = md5(DataToSign+timestamp)
else if (timestamp mod 5 = 4)
signature = md5(DataToSign+(timestamp+1))
else if (timestamp mod 5 = 3)
signature = md5(DataToSign+(timestamp+2))
else if (timestamp mod 5 = 2)
signature = md5(DataToSign+(timestamp+3))
else if (timestamp mod 5 = 1)
signature = md5(DataToSign+(timestamp+4))
```

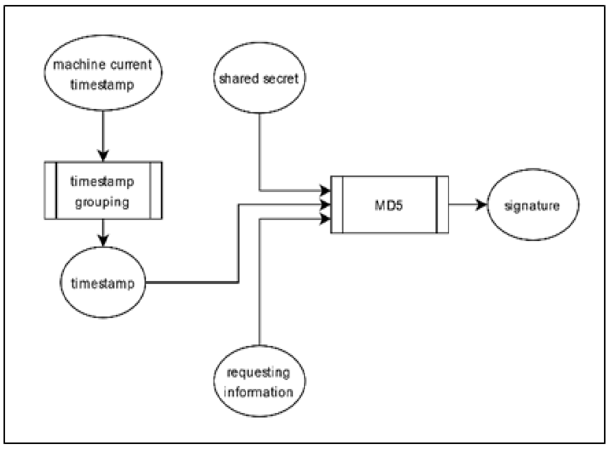

Fig. 1: Signature generation at client side

D. Server

Server is machine which listen for incoming request, validates API signature and allocates information or resources requested by client. To implement grouping mechanism following temporary variables will be needed. Values for that variables are obtained as shown below. To validate a request first server will generate its grouped timestamp (here i). And it will generate hash by keeping gap of 5, as discussed earlier and check if receivedHash matches with its generated hash or not. If hash matches then request is fulfilled else rejected. To implement this, following procedure is used.

```
//temporary variables
receivedHash = signature received from client
API key = API send by client during request
shared secret = check in database for shared secret of API key
timestamp = Current timestamp of server in milliseconds
N = tolerance i.e. seconds for which hashes are to be checked
//function – timestamp grouping
if (timestamp mod 5 = 0)
i = timestamp
else if (timestamp mod 5 = 4)
        i = timestamp+1
else if (timestamp mod 5 = 3)
        i = timestamp+2
else if (timestamp mod 5 = 2)
        i = timestamp+3
else if (timestamp mod 5 = 1)
        i = timestamp+4
//function – sign & validate
 for (counter i less than (timestamp-N))
  {
   calculatedHash = md5(requesting information+ shared secret+ i)
   if(calculatedHash = receivedHash)
        exit loop and grant the resources
  i = i+5
  }
```

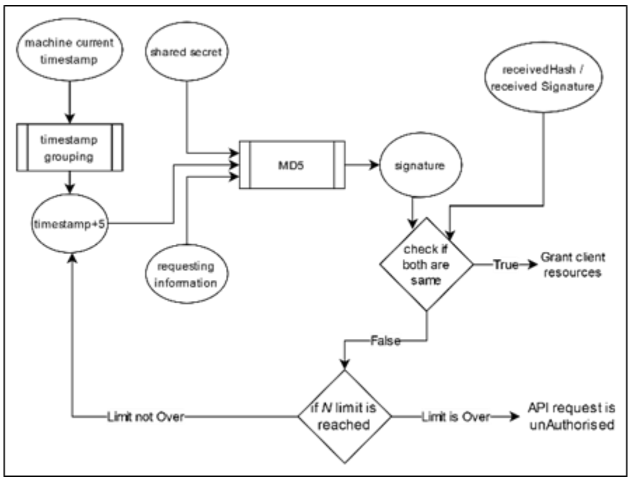

Fig. 2: Signature validation at server

V. Comparison

Below chart is representation of computation needed before using timestamp grouping (earlier) and after using timestamp grouping (now). X axis represents latency, the time gap between request sent from client and request reached to server. Y axis represents computations, number of operations needed to validate one request. From above comparison graph it is clear that using time grouping technique, time required to validate same request is less than that without using it. Example if one request takes 200 computations then the same request is validated in 40 computations using time grouping.

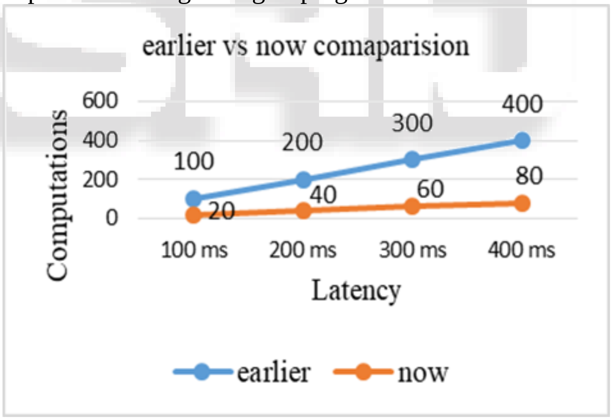

Fig. 3: Computations needed earlier and now after applying timestamp grouping

VI. Conclusion

This mechanism will reduce time required for API request to be fulfil, making way for faster applications and better user experience. Extra calculation workload on server will reduce drastically improving resource utilisation and improved performance. Due to reduced workload cost of server will also decrease as it will use less resources to validate requests.

References

[1] Google Maps, "Get an API Key and Signature | Maps Static API" [Online] Available: https://developers.google.com/maps/documentation/maps-static/get-api-key

[2] Kopecký, Jacek & Fremantle, Paul & Boakes, Rich. (2014). A history and future of Web APIs. it - Information Technology. 56. 10.1515/itit-2013-1035.
[3] Guy Levin, “State of API Security, See what the future of API security could be” [Online] Available: https://dzone.com/articles/state-of-api-security