MOSQUITTO SECURITY ADVANCEMENT USING THE CHECKSUM FEATURE

B. V. S. Bharghava
Computer Science and engineering, KalasalingamUniversity, Krishnankoil, Tamil nadu, India.
Email: bhargavaraja12@gmail.com

B.Haritha
Computer Science and engineering, KalasalingamUniversity, Krishnankoil, Tamil nadu, India.
Email: harithabattu25@gmail.com

Mrs.R.Kanniga Devi
Assistant Professor, Computer Science and Engineering,
KalsalingamUniversity, Krishnankoil, Tamil nadu, India.
Email: rkannigadevi@gmail.com

***Abstract*— This paper will give a method of increasing the security feature of the MQTT by using the checksum instead of using the MAC's and digital signatures which will consume more data. This checksum can easily identify any data modification with high accuracy.**

Keywords: checksum, MQTT-security, MQTT, Data Integrity;

I. INTRODUCTION

Mosquitto is open source software, which uses the MQTT (Message Query Telemetry Transport) light weight protocol and enables the communication between the machines. It uses Publisher Subscriber model. MQTT properties which make it light weight are low bandwidth, high latency, data limits and fragile connections and also the headers are maintained as small as possible to keep MQTT light weight. The next updated version of MQTT is MQTTS where S stands for security; MQTTS provides a securable connection between publisher and subscriber. In MQTT unauthorized encrypted support provided by using SSL/TLS certificateascatfile/capfile (certfile and keyfile) and pre shared keys. MQTT also provides username/password authentication but while using this we need to ensure that network encryption is on. The MQTT is using the MACs and the digital signatures instead of the checksum. If we use the checksum this will calculate stamp which is typically added to the payload. The receiver of the packet can verify the data integrity recalculating and validating the stamp. If we are able to use the checksum efficiently, we can avoid the use of the MACs and Digital Signatures which use the large size of the data when compared to the checksum.

MQTT supports unauthorized communication where there is no need of usernames and passwords, they will use the certificate based encryptions. These are mainly used for the public communications as there is no need of very high level of security. This certificate security is provided so that whatever the publisher and the subscriber are sharing the information and this will not be vulnerable for a hacker. The other method of the security used in the MQTT is the username/password authentication method wherepassword_file is used to define username and passwords. While using this type of encryption, our network should be encrypted. In certificate based encryption there will be an option called require_certificate which will be set to either true or false. When require_certificate is false then there is no need of the client authentication. But when require_certificate is set to true the user have to provide valid username and password in order to proceed with common name from the client. If the authentication is not valid then the client is allowed as MQTT client. Pre-shared-key concept is also the same. The encryption using psk_hint,this option enables psk support for the listeners and also act as an identifier for the listeners this hint is sent to clients and may be used locally to aid authentication. The hint is free from the string that doesn't have much meaning in itself. The other option is to create a psk_file. If done so, we have to create a security plugin to handle them.

MQTT provides 3 qualities of service levels

- At least once:

The MQTT will give the surety that the packet will be sent to the subscriber at least. But there is a case that the subscriber may receive duplicate packets.

- At most once:

The subscriber will send the message to the publisher once there will not be an acknowledgement sent by the publisher or it won't be stored or redelivered. This is also called as "fire and forget". In this case the publisher may or may not receive the data sent by the subscriber.

- Exactly once:

The MQTT will ensure that the message is received by the subscriber exactly once that is it will make sure that no duplicated massage is transmitted to the subscriber. This is mainly used in the accounts as they will not allow a duplicate data.

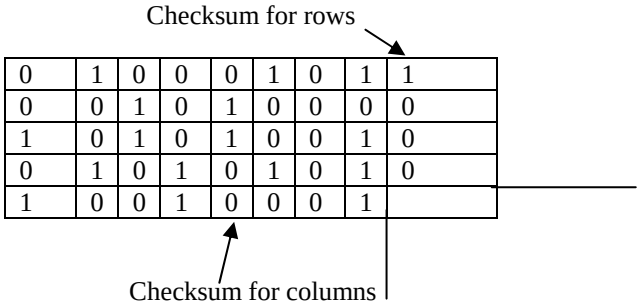
The main encryption is usually done to the MQTT publisher packets. If there is a necessary we can write a custom broker plug-in which can decrypt the encrypted data on broker side they could be

- Publish topics
- Connect the username and password
- Subscribe topics
- Unsubscribe topics

In order to achieve this we can use the encryption methods like END-TO-END encryption,CLIENT-TO-BROKER encryption, Symmetric encryption, Asymmetric encryption.

II. PROPOSED WORK

Using checksum in MQTT protocol is very easy but this will be easily attacked by the attacker. If we use the other mechanisms like MAC and digital signatures the header size will be increased. But in TCP/IP anyhow we have a field for checksum; we can use that field in an effective manner. So we are proposing a new mechanism for handling this in a very effective manner and data can't be revealed. The former mechanism of checksum is given below



0	1	0	0	0	1	0	1	1
0	0	1	0	1	0	0	0	0
1	0	1	0	1	0	0	1	0
0	1	0	1	0	1	0	1	0
1	0	0	1	0	0	0	1	

Here the chance of decoding the message is very high as we need to know just what number is represented for odd and even (i.e. in the above example we are calculating based upon number of one's means for odd number of ones we have represented with 1 and for even number of one's it is zero). The only possible ways to represent checksum is the above case or considering one and zero in reverse order so that we can easily identify the checksum mechanism. If the third party has captured our data and has been modified, they can generate the correct check sum in most cases as there are only two possibilities. So we are proposing a new mechanism for using the checksum with the key.

The sender first transmits the key to the receiver with encryption. Now sender and receiver both are having the key. At the sender side the checksum will be calculated based upon the key.

Consider the key maybe 1116.

Table for generating the checksum

Remainder for checksum	Odd	Even
0	1	0
1	0	1
2	0	1
3	1	0

Based upon the remainder the checksum will be calculated. It is not uniform as in the above case. We will see an example of this, The first line is

1 1 0 00 1 0 1

Initially our key is 1116 so (1116%4=0). So from the table generating the checksum 0(remainder) correspondent values are 1 if odd 0 otherwise. In the above example 0 1 0 00 1 0 1 as it is having odd number of ones we mark checksum as ‘’1’’. After that the key value will be decremented by 2(i.e. 1116/2=558). Now the key will become 558.

For the second line 0 0 1 0 1 0 000. Now the key is 558, so 558%4=2. From the table validating the checksum two correspondent values is 0 for odd 1 otherwise. We have even number of one’s so we mark the checksum as “1”. In both the cases that for even number and odd number of ones we got one as checksum so it is hard to identify the checksum mechanism used. By using this we can easily know that if the data has been modified or not.

III. CONCLUSION:

Our proposedmethod of encrypting the checksum effectively uses the available checksum feature instead of the MAC and the digital signature in a most securable manner. Using this method the probability of finding the checksum mechanism will go down than the existing mechanism.

In the future, we can extend this mechanism by increasing the divisor and also we can change the orders in “table for checksum” so that it is even difficult to predict the order that we are using so, that the modified data will be identified easily at high rate. Depending on the security requirement we can increase the divisor for high security and small divisors for the generalsecurity.

REFERENCES

[1] Meena Singh, M.A.Rajan “Secure MQTT for Internet of Things (IoT)” CSNT 2015 fifth international conference, October 1, 2015.
[2] UrsHunkeler, Hong Ling Truong “MQTT-S – A Publisher/Subscriber protocol for wireless sensor networks” COMMSWARE 2008 Third international conference, June 27, 2008.
[3] http://www.hivemq.com/blog/mqtt-security-fundamentals-payload-encryption
[4] https://eclipse.org/mosquitto/man/mosquitto-conf5.php