International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 4 Issue: 5 164 - 168

A packet forgery and packet drop attack detection using efficient Encoding and Decoding mechanisms.

Rashmi R H
Student, M Tech., Department of Computer Science Engineering,
CMRIT, Bangalore, India
rashu.holla@gmail.com

Mrs. Sagarika Behera
Associate Professor, Department of Computer Science Engineering,
CMRIT, Bangalore, India

***Abstract*--**A series of systems are connected to form a network. Vast amount of data moves through this network. A user who is not authorized to access the data tries to get forcible entry to the network by adding a malicious node or spoiling the node that already exists. The data in such network is not trustable. There is always a possibility that the information sent is changed or it is dropped in-between. Therefore identifying such behavior and the reason behind it is one of major issue. This is solved by using Provenance that verifies the data originality. The proposed technique is mainly concerned with the safe transmission of packet from sender to receiver through intermediate participants. The mechanism detects if there is any data change attack or data drop attack in the network and identifies the intruder responsible for such attacks with the help of provenance. The system is implemented experimentally and analyzed for its effectiveness

***Keywords*--** *Network, provenance, packet forgery, packet drop, encryption, decryption.*

*****

I. INTRODUCTION

A network consists of devices communicating with each other. These communications involve lot of information exchange. Since thisinformation is communicated through many sources it becomes necessary to validate the data originality. The information received at the receiver end is used for making important decisions; therefore information validity becomes one of the major requirements.

Provenance of information is one of the key elements to prove the validity. It is used to make sure that only proper data is used for processing at the receiver. They consists details on the sender from where data is sent, the path taken by the data and the provenance factor that proves the originality. These details help in tracking the attack in the network.

The studies carried out recently have shown that using wrong data leads to failure which cannot be recovered. The study also showed that the provenance application was limited to databases and workflows and was not effectively implemented in networks. In network using the provenance has many performance related challenges; therefore the scheme using this technique should address these challenges in mind as well.

In the proposed mechanism, the packet is transferred in the network in a secure manner. The path taken by the packet is monitored by the server which is similar to the Base Station in wireless sensor network.The server is responsible for receiving the information and checking for its integrity. The server also gets immediate alert if any packet is dropped and the node where the drop has occurred. The server also maintains details about every packet received and the status of that packet i.e whether it is successfully received/Changed/Dropped.

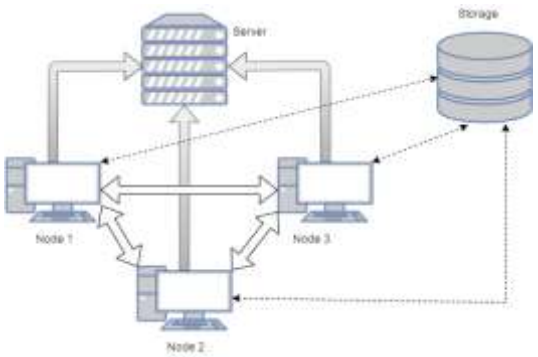

Figure 1: Architecture of the network

II. RELATED WORK

Ramachandran et al.[2]discussed about*Pedigree*that saves provenance information in tags and attaches it to the packet being sent. These tags contain details about the process applied on every packet and also the node history. The only constraint here is the network is in trusted environment. W. Zhou et al.[3]proposed*ExSPAN*where a protocol executed in distributive manner is used for storing the brief information about the state of the network but the drawback is that, the security is not considered and only applicable for only some scenarios. They further introduced"*Secure Network Provenance"*[4]that can be implemented in unreliable environment but resource constraints are ignored here.

N. Vijayakumar et al.[7] introduced an almost real-time provenance system for stream of data, which detects the source of stream long after the process ends.

S. Chong et al.[8]introduced the procedure of using the provenance within the packet. Since the security is not considered here the intrusion is not handled. They first used bloom-Filters for storing provenance within the packet.

S. Sultana et al.[10]they described that many discrete transmission of packets are necessary for securely sending the provenance. Here the notion was that, the provenance does not vary for at least few packets.

S. Sultana et al.[1] , introduced a system where sending of provenance is achieved using in-packet bloom filters. The mechanism mainly concentrated on the secure transferring of provenance, an encoding as well as decoding techniques were designed for this purpose. They also proved that the resources used by this mechanism is very less so it is more suitable for "Wireless Sensor Networks". The proposed idea in this paper is obtained from this research work.

III. SYSTEM MODEL

In this section, introduction about the network, data and provenance model used is described.

A. *Network Model*

A Local Area Network is used to implement the scheme. The network consists of one server and three nodes N1, N2 and N3 (restricted to 3 for implementation convenience) connected to each other. IP Addressand Port number are used for linking nodes. A single node is designated as sender. Server has the responsibility of generating the secret key k_i, collect the packets from dispatcher node and check for alteration and packet drop.

B. *Data Model*

The text document is fragmented into packets and is sent to the server from the sender node.
The path of the data with D hops is shown as $<n_l,n_1,n_2..n_d>$ where n_l is the source and n_i is *i*steps from source. The packet is convertedto cipher-data at every node it travels using key sent from the server to the nodes.Multiple encryptions increase security of data.

C. *Provenance Model*

We consider Provenance at node-level[1] that is nodes areprogrammed at each stage of handling the data. Given a packet *p*, its provenance is represented in the form of graph G(V,L) where vertex$v \in V$is attributed to specific node NODE(v)=n and provenance record for that node is represented. Vertex ID generated by cryptographic functions is used to uniquely identify each vertex of the provenance graph.

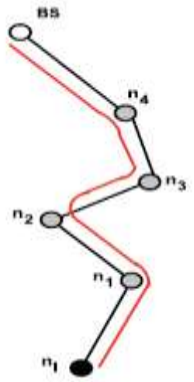

Fig 2: provenance graph

In Fig 1, the source node n_l generates data packet p, at intermediate nodes the data is encrypted with the node's key and then forwarded to the Server.

D. *Threat Model*

The assumption is made that the Server is trustworthy but the other nodes in the network can be malicious. An intruder can monitor the ongoing traffic in the network and also install a malicious node or compromise the existing nodes of the network with the intention of disturbing the network.

IV. IMPLEMENTATION

A. *Server login page*

A user logs in to the server which is connected to many nodes. The user enters his userID and passcode to obtain a successful entry. Wrong credentials blocks the customer from logging in to the system thus maintaining the secrecy. The authorizations entered by customer is crosschecked with the ones stored in the Database(DB) table, therefore the valid IDs and passcodes must be stored in DB. The following condition is checked:

/*'name' and 'pass' are user input, 's1' and 's2'- obtained from DB. */
"if((name.equals(s1))&&(pass.equals(s2)){}"

B. *Generation & allocation of key*

The proposed mechanism includes an proficient encoding that requires a key(k_j) which is kept secret.The k_j is actually generated automatically by using Class called "Random", which produces a unique key using "48-bit seed", which is changed using a formula that is "congruential linear formula". This makes sure that different keys are generated for different nodes.

"/*Random key generator code snippet*/
Random rk=new Random();
int rndky=rk.nextInt();"

C. *Path generation module*

All the possibe paths that a packet can take is stored node-wise. A table is created with each column representing each node. The different paths from that node to the server is listed and is distributed across every nodes.
"t_nodip=(TreeMap)ios.readObject();//NodeIP is added to tree"
"t_nodprt=(TreeMap)ios.readObject();// node port number is added to tree"

D. *Module for generating packets*

The information to be sent is in the format of text, which cannot be sent through the network. For successful sending of information these text documents are split into smaller fragments called packets.These fragments are indexed and accelerated to maintain the series.

"while(sr.hasMoreTokens())//sr: string
//tokenizer object
{

```
pkets[idex]=sr.nextToken();//idex:index
//of packets
 idex++;
 }"
```

E. Encryption and decryption provenance module.

The Provenance encrypting mechanism occurs at all the nodes whereas the decoding mechanism only occurs at the server only after successful reception of scrambled data. The packet is known using its 'index number'.

Provenance Ciphering

The nodes of the network are called vertices.Each time a packet reaches a particular node a vertexPacket(vp) is generated. They are generated by encoding the received packet with the help of the key given to the node by the server.this generation happens at all the nodes that receive and forward the packet. The vp is created using the below equation:

$vp_k = EcryK_i(\ pket[idex]\ );$

Where Ecry is an encoding method *idex*is the index of every packet.

Algorithm 1 Packet Encryption

Input: The packets(*pket*) received by the node and the index(*[idex]*) of the packet, key of that particular node (K_{idex}) and the Route $R_p=<n_l,\ldots,n_1,\ldots n_d>$

P←NULL//Empty provenance string
foreach n_{idex} Є R_p **do**
$vp_{idex} = EcryK_{idex}(\ pket[idex]\ )$//Encrypt the data
endfor
P←vp_{idex}
return P

The algorithm 1 shows the steps involved in encoding process.

Provenance Deciphering

The server on receiving the packet which is encoded at each node it visited, performs the decrypting technique using that particular node's key stored in the server.The number of times the decoding is performed depends on the number of nodes the packet has visited. That is if the ($pket_{idex}$) has visited m nodes before reaching the end node, then it must be decrypted m times.

The intruder intentionally changes the content of the packet when it reaches the malicious node before encoding and forwarding it, this can be checked by comparing the data in the packet obtained with the original data sent.The following equation is used for decrypting:

$$dcryptedData_{idex} = DcryK_{idex}(EcryK_{idex}(pket[idex$$
$$= DcryK_{idex}(vp_{idex})$$

Where Dcry is the decoding method that is used to obtain the data of the packet with index number*idex*.

Algorithm 2 Packet Decryption & Detecting data modification

Input: The encrypted packets($pket_{idex}$) i.e $EcryK_{i\ dex}(\ pket[idex]\ )$, key of that particular node (K_{idex}), the Route $R_p=<n_l,\ldots,n_1,\ldots n_d>$ and *OriginlData*

$dcryptedData_{idex}$ ← NULL
foreach n_{idex} Є R_p **do**
$DP_{idex} = DcryK_{idex}(EcryK_{idex}(\ pket[idex]\ )) = DcryK_{idex}(vp_{idex})$
endfor
$dcryptedData_{idex}$ ← DP_{idex}
if ($dcryptedData_{idex=}OrignlData$) **then**
return true// Provenance is verified
endif
return false

The algorithm2 is used for obtaining the data and detecting changes made to the packets.

F. Module for detection of droppin of packets

The packet sent by the sender sometimes does not reach the server at all, this is because the particular packet is been dropped at some intermediary node. The attacker controlling one of the mid-way node intentionally drops the data resulting in an attack. Such attack should be identified fast and the attacking node should be removed. The proposed mechanism is extended to detect such attacks.

In the route R, the source is s_l and the connection between the two nodes d_i and d_{i+1} is c_i. If the packet drop occurs at the link c_i, then d_{i+1} is responsible for the drop.

The below snippet runs the thread used for finding the dropper.

```
/*Thread running to detect dropping*/
public detectDropper()
{
 thr=new Thread(this);
 thr.start();
}
```

V. RESULTS ANALYSIS

After successful implementation the next step is to document the results save the snapshots and analyze the performance effectiveness of the proposed technique.

Consider a scenario, a packet no. 8 is sent to the server from the node no. 1 along the route 2-0. The attacker deviates the packet from node no. 2 to node no. 3 and intentionally drops the data.In fig 3a, the dropping is shown and in fig 3b the server is alerted about the attack.

In another scenario, the packet is sent from the source node no. 1 and the data travels in the route 2-3-0. This route contains two mid-way nodes so double encoding is done. When the data reaches node no. 3 the content of the packet is changed, which is detected by the server once the data reaches the server.The fig 4a shows the modifying node no. 3 and fig 4b shows the server detecting it.

Fig 5 shows the tables maintained by the server which contains details about every packet sent by the sender.

The proposed method was executed in two ways, one way was using a single system. Another way was using separate systems for server and nodes.

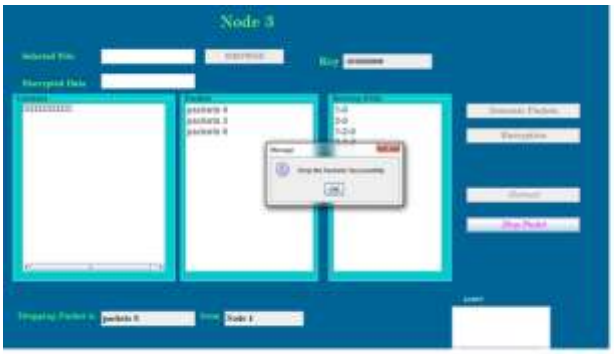

Fig3a: Packet is been dropped intentionally Fig 3b: Server is notified about the attack

Fig4a: Unauthorized changing of data Fig 4b: data change is notified to the server

Fig5: Tables at server with packet information

VI. CONCLUSIONS

The method implemented in this paper, solved the problem of detecting two major attacks of the network that is the data forgery attack and the dropping of packet attack. The encoding and decoding methods were used in an efficient manner to improve the performance of the system. The technique helps in maintaining the originality and secrecy of the data throughout the network. Through the experimental results, the proposed technique proved that it provided all the functionalities necessary for successful installation and working of the detection scheme. In future enhancement, the detection must be more accurate and effective. This methodology should be utilized in real-time sensor networks.

REFERENCES

[1] Salmin Sultana, Gabriel Ghinita, Elisa Bertino"A Lightweight Secure scheme for detecting provenance forgery and packet drop attack in Wirelesss Sensor Network" published in IEEE Transactions On

Dependable And Secure Computing, Vol. 12, No. 3, May/June 2015

[2] Ramachandran, K. Bhandankar, M. Tariq, and N. Feamster, "Packets with Provenance," Technical Report GT-CS-08-02, Georgia Tech, 2008.

[3] W. Zhou, M. Sherr, T. Tao, X. Li, B. Loo, and Y. Mao, "Efficient Querying and Maintenance of Network Provenance at Internet- Scale," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 615-626, 2010.

[4] W. Zhou, Q. Fei, A. Narayan, A. Haeberlen, B. Loo, and M. Sherr, "Secure Network Provenance," Proc. ACMSOSP, pp. 295-310, 2011.

[5] R. Hasan, R. Sion, and M. Winslett, "The Case of the Fake Picasso: Preventing History Forgery with Secure Provenance," Proc. Seventh Conf. File and Storage Technologies (FAST), pp. 1-14, 2009.

[6] A. Syalim, T. Nishide, and K. Sakurai, "Preserving Integrity and Confidentiality of a Directed Acyclic Graph Model of Provenance," Proc. Working Conf. Data and Applications Security and Privacy, pp. 311-318, 2010.

[7] N. Vijayakumar and B. Plale, "Towards Low Overhead Provenance Tracking in Near Real-Time Stream Filtering," Proc. Int'l Conf. Provenance and Annotation of Data (IPAW), pp. 46-54, 2006.

[8] S. Chong, C. Skalka, and J.A. Vaughan, "Self-Identifying Sensor Data," Proc. Ninth ACM/IEEE Int'l Conf. Information Processing in Sensor Networks (IPSN), pp. 82-93, 2010.

[9] H. Lim, Y. Moon, and E. Bertino, "Provenance-Based Trustworthiness Assessment in Sensor Networks," Proc. Seventh Int'l Workshop Data Management for Sensor Networks, pp. 2-7, 2010.

[10] S. Sultana, M. Shehab, and E. Bertino, "Secure Provenance Transmission for Streaming Data," IEEE Trans. Knowledge and Data Eng., vol. 25, no. 8, pp. 1890-1903, Aug. 2013.