IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 1 Issue 3, May 2014.
www.ijiset.com

ISSN 2348 – 7968

# Forensic Analysis for Collaborative Network Security Management System

**M.Sankara Mahalingam[1]** **M.K. Nagarajan[2]**
[1]PG Student, CSE, Kalasalingam Institute of Technology, Krishnankoil.
[2]Assistant Professor/CSE Department, Kalasalingam Institute of Technology, Krishnankoil.

***Abstract*—Network Security Appliances are deployed at the vantage point of the Internet to detect security events and prevent attacks. However, these appliances are not so effective when it comes to distributed attacks such as DDoS. This paper presents a design and implementation of collaborative network security management system (CNSMS), which organize the NetSecu nodes into a hybrid P2P and hierarchy architecture to share the security knowledge. NetSecu nodes are organized into a hierarchy architecture so they could realize different management or security functions. In each level, nodes formed a P2P networks for higher efficiency. To guarantee identity trustworthy and information exchange secure, PKI infrastructure is deployed in CNSMS. Finally experiments are conducted to test the computing and communication cost.**

***Keywords*-network security; collaboration; secure message exchange;**

## I. Introduction

Firewalls, Intrusion Detection System (IDS), Anti-Virus Gateway etc. are now widely deployed in edge-network to protect end-systems from the attacks. When the malicious attacks have fixed patterns, they can be clocked by recording and matching these patterns. This method works well in the past tens of years. However, nowadays sophisticated attacks are distributed in the overall Internet, have fewer characteristics and transform quickly. Especially, the Distributed Denial of service (DDoS), contains very few, if any, signatures strings to identify. It aggregates large quantity of malicious traffic from different sources, then turns the server down quickly by over depleting the server's computing and storage resources. Confronted with such attacks, the traditional security appliances always have poor performance, so better mechanism are necessary to prevent these attacks.

Better flow management is a method to normalize traffic behaviors and prevent burst large flows from depleting the server resources. There has been a lot of research on this. Clean-slate Program [1] uses state-based switch and controller to control the TCP/IP flows. Openflow controller and Openflow Switch [2] is another experiment deployed in campus network for flow management. In the 4D architecture described in [3], [4],a special plane is abstracted for flow management. Based on 4D architecture, Tesseract [5] further implemented direct control of a computer network, but it's limited in a single administrative domain.

Collaboration is another way to be taken as prospective to prevent distributed attacks. In [6],the author reviewed researches in collaborative intrusion detection system. He presented that by collaboration, the system could realize scalability, teamwork, and has a bigger picture of events in the whole network. In [7], an algorithm is presented to improve the alert event's accuracy by aggregate information from different sources. The authors of [8] put forward a similar alert correlation algorithm which is based on Distributed Hash Tables (DHT). This paper aims to develop a new collaboration system, Collaborative Network Security Management System (CNSMS). In [9], we discussed how NetSecu nodes could manage security problems in a sub-domain and provide P2P communication interfaces. In this paper, CNSMS realizes the communication between these NetSecu nodes. More specifically, CNSMS will achieve the following objectives:

1) Security policy collaborative dissemination and enforcement;
2) Security event collaborative notification;
3) Security ruleset library upgrade;
4) Scalability;
5) Trust infrastructure.

In CNSMS, a hierarchical architecture of three levels is implemented. The third level is basic NetSecu nodes. The second level is domain NetSecu nodes to manage the membership in corresponding sub-domains. And to have a big picture of the whole network, the ruleset library is stored in the Central Management System, which is the first level. Considering there would be hundreds of even thousands of NetSecu nodes in the CNSMS, we use P2P communication mechanism in each level to improve the information exchange efficiency. The trustworthy of identity is based on PKI: cryptography primitives such as signatures and encryption signatures are used to ensure the integrity and credentials of message exchanged between peers.

The following parts are organized as follows: Section II

ISSN 2348 – 7968

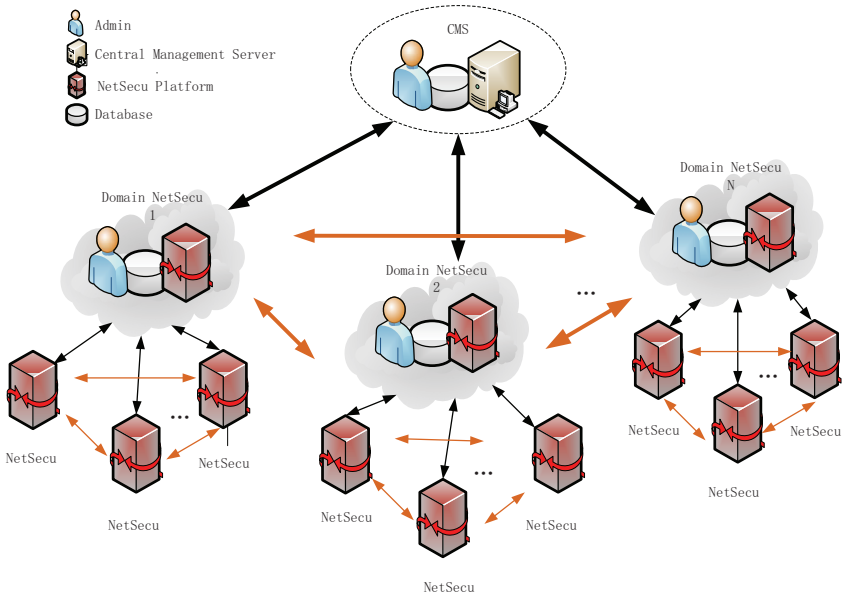


Figure 1. Hybrid Architecture in Metropolitan Area Networks

introduces the structure of CNSMS system. Section III presents neighborhood establishment, message protocols and communication procedures in-between NetSecu nodes in CNSMS. Section IV presents the underlying trust infrastructure based on PKI CA. Section V evaluates the efficiency of secure message exchange and corresponding computation and commutation cost in CNSMS system. Finally the conclusion is given in VI.

## II. The Structure of CNSMS

### A. The hybrid architecture in Metropolitan Area Networks

To achieve scalability, we use a hybrid P2P and hierarchical architecture. There are three levels in this hierarchy. The third level is normal NetSecu nodes, they are arranged into a P2P topology, and are affiliated with one Domain NetSecu node in level two. The domain NetSecu nods in level two act as domain controllers and tracker servers. These domain nodes could also communicate with each other in a P2P network. The first level super node—-CMS has direct connection with every domain NetSecu node of level two. Therefore, the level two nodes and the super node form a star topology, with CMS in the center. CMS will keep the latest record of security events, and manages the overall meta-information of all the nodes such as idnetity, certificates, ruleset and log. The overall architecture is shown in Fig. 1.

### B. The hybrid architecture in Local Area Networks

In local area networks as shown in Fig. 2, there are very limited NetSecu nodes which are probably in the same domain. Hence the Domain Netsecu node is unnecessary;

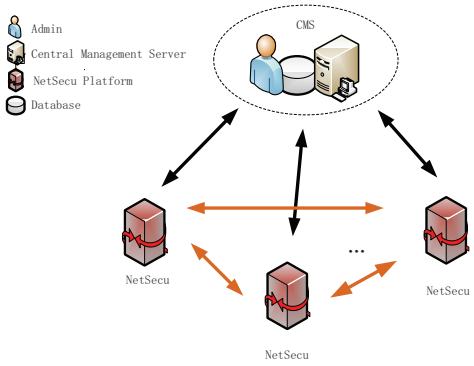


Figure 2. Hybrid Architecture in Local Area Networks

CMS will manage the whole Security System itself. This is the case of our experiment environment.

When a security event occurs at a NetSecu node, it reports the event to CMS and propagates it to peers in its same level. Then the newly informed peers will update their security policy, ruleset library version and other information based on the message. When a NetSecu node wants some kind of special information, it can contact its P2P counter-nodes or CMS for messages. For the integrity of data and convenience of analysis, CMS keeps the latest information on security policy and events. CMS is also responsible for keeping the overall ruleset library updating and membership control.

We use this hybrid architecture for the following discussion.

ISSN 2348 – 7968

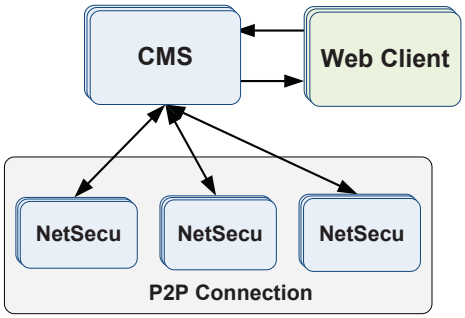


Figure 3. Implement mode of CMS

### C. implement mode

To facilitate the administrators to operate, CMS is implemented in B-S (browser-server) mode: the CNSMS administrator uses a web client to configure or update the CNSMS's policy and ruleset library versions, as shown in Fig. 3.

## III. Data Dissemination Protocol in CNSMS

### A. Neighborhood Establishment

When one NetSecu node wants to exchange information with another NetSecu node, it must exchange identity information with the other node. Or in other words, these two nodes must establish neighborhood. Remember that in CNSMS, CMS manages the overall meta-information of all the nodes, NetSecu nodes needs the assistance of CMS to establish neighborhood. The neighborhood mechanism is described as following:

- Suppose there is a NetSecu node B, wants to establish neighborhood with another NetSecu node A, B will post the "Neighborhood Request" to A. If A intends to accept this request, it asks B to synchronize their meta-info with the CMS. Then they get the authentication and identity with each other as shown in Fig. 4;
- When the neighborhood is established, NetSecu A and B could further update ruleset library to the newest version from the CMS or the other up-to-date NeSecu node. When A discovered that B is updated, it can get data from B then pass the data to other nodes in A's neighborhood list, as shown in Fig. 5;
- Nodes update their database periodically.

### B. Message formats

There are four kinds of messages in the CNSMS: *EVENT*, *INFORM*, *REQUEST* and *REPLY*. They share the same message format shown in Table I.

The field TYPE defines the LOAD's type contained in this message. The field SRC_ID points out the sender of the message. The field LOAD contains the content of the message, which is encrypted. The fields ENCRYPT_ID, LOAD_SIG and SIG are for security purpose, which we will discuss more detailed in Section IV.

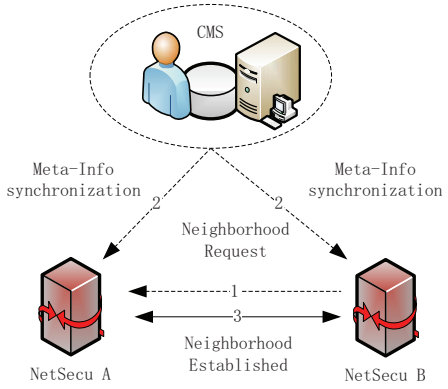


Figure 4. Neighborhood Establishment

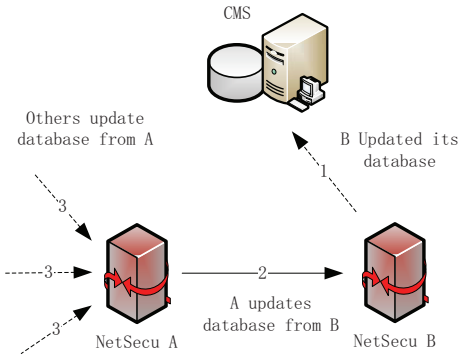


Figure 5. Dadabase updata in P2P mode

Table I
Fields of Control Message

| Fields | Description |
|---|---|
| TYPE | { *EVENT*,*INFORM*, *REQUEST* and *REPLY* }<br>*EVENT*: Notice of new security events<br>*INFORM*: Inform new version of database and Downlaod new data from library<br>*REQUEST* and *REPLY*: Get special information from others |
| SRC_ID | ID of the source of this message |
| ENCRYPT_ID | ID of the signer of LOAD field |
| LOAD | Detailed description of the message |
| LOAD_SIG | Signature of LOAD field |
| SIG | Signature of the message |

### C. Message exchange

To handle different situations and needs, three forms of message exchange are defined in CNSMS: event notification, CMS inform, get and reply.

When a node detects a security event, it creates a new *EVENT* message, fills the LOAD field with the five-tuple information, structural description and action taken on this event. Then it adds signatures to corresponding fields, and publishes it to its neighbors and CMS. When one node receives an authorized message, it could choose whether

ISSN 2348 – 7968

to refine its security policies or not, according to its local situation and the sender's trust level.

When the CMS wants to disseminate a new ruleset library, it creates a new *INFROM* message, fills the LOAD field with release time, version and data length, adds corresponding signatures, and then broadcasts the message to NetSecu nodes. After receiving the *INFORM* message, the NetSecu nodes start to download the new ruleset library in P2P way. That is to say, one node could download directly from CMS or other up-to-date neighbor nodes.

When a node wants some special information, it generates a *REQUEST* message, fills the LOAD field with its requirement and a sequence number, then publishes the message to its neighborhood and CMS. On the other hand, when a node receives a *REQUEST* message, it checks its own local database. If it has an answer, it replies the request with a *REPLY* message, with the LOAD field containing the requested data blocks and the sequence numbers.

## IV. Credibility in CNSMS

### A. PKI and digital certificate

Public Key Infrastructure (PKI) is a widely used mechanism to manage digital certificates. A digital certificate is a chunk of data that contains user's identity, public key and signature signed by a certificate authority. We use the PKI mechanism to realize authentication and validation of information, so the integrity and nonrepudiation of communication among NetSecu nodes and CMS could be guaranteed.

Every node must hold a certificate to identify itself in CNSMS. So every NetSecu node and CMS needs to ask CA to create a digital certificate for it before it starts communication with others. There are two ways for CA to distribute certificates to nodes: out-of-band and in-band. The out-of-band method is adopted in our experiment: CA generates static certificates, nodes downloads these certificates from CA.

In our experiment, we run a special module in CMS to act as CA to generate standard X.509v3 certificates for NetSecu nodes.

Whenever there is a new node, it registers to CA, CA generates a pair of public key and private key, then distributes the certificate which contains the public key to other nodes. When two nodes in CNSMS need communication, the sender first provide its certificate, the receiver will verify the public key and send the certificate to CA for validation. Only when the authentication is passed, nodes can handle messages. Their relationship is shown in Fig. 6.

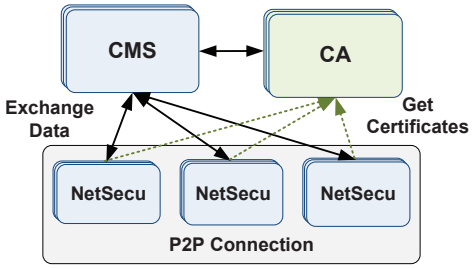


Figure 6. PKI CA in CNSMS

### B. Secure Message Exchange

For secure communication, not only the sender and receiver of the message need to provide certificate when they initiate a message exchange, but also the messages in CNSMS are encrypted to avoid malicious modification and man-in-middle attack.

As described above, every node in the CNSMS has asked CA to create a pair of keys. They keep the private key themselves but send the public key to its neighborhood peer. When a NetSecu node broadcasts an *EVENT* message, it uses the private key to sign/encrypt the LOAD field and the overall message respectively, fills the LOAD_SIG and SIG field with generated signatures. When other nodes receive this message, they would search the public key according to the ENCRYPT_ID, and use it to decipher the signatures to verify the load and message.

*REQUEST* and *REPLY* messages only involve one sender and one receiver in communication. Therefore only SIG field is used to verify the authenticity of the message.

*INFORM* messages are more complex. The CMS uses this kind of message to advertise ruleset library notice to NetSecu nodes. Then nodes could download new version from CMS. To speed up the process, P2P communication is utilized: a NetSecu node could download new data from an updated node it has access to. When the CMS has new data, it publishes an encrypted *INFORM* message to NetSecu nodes. Suppose a NetSecu node A receives such a message, it creates a data-request which contains its signatures and sends it to CMS. After the CMS verified A's signature, A could download data from CMS and update its ruleset library. Now A is updated, it publishes an *INFORM* message to others so other NetSecu nodes could download data from A. In A's notice, the *INFORM* message is generated by A, so the SIG field is signed by A. However, the content of LOAD is from CMS, so the LOAD_SIG is signed by CMS. When another NetSecu node, such as B, received this *INFORM* from A, it needs to both verify LOAD_SIG and SIG before updating.

Generally, the signature and verification consume a lot of computing resources. As shown above, multiple verifications occur during the transmission of one message in some cases for security reasons. We will deal with the CPU and IO consumption problem in the next section in detail.

ISSN 2348 – 7968

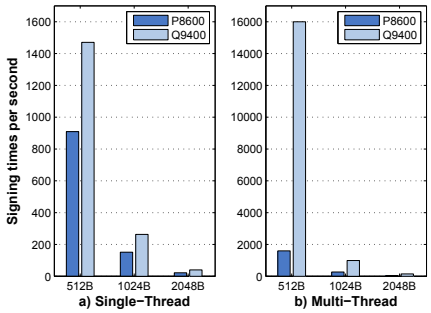


Figure 7. Signing Performance

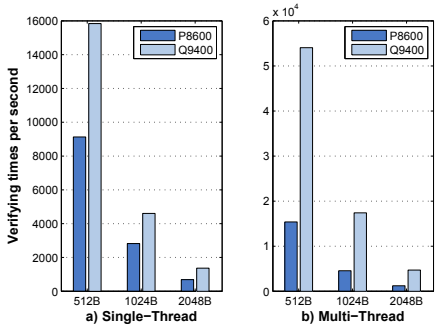


Figure 8. Verifying Performance

## V. Performance Evaluation in Secure Message Exchange

In order to evaluate the computation and communication cost needed, some experiments are carried out. These experiments are also designed to decide the scale of one typical P2P domain.

### A. The Computing Cost

A node's CPU is mainly spent on signing and verifying messages. So we check the computing cost of signing and verifying respectively.

To check the load of CPU in NetSecu nodes, two types of hardware platforms are used: one has an Intel Core 2 Duo P8600 2.4GHz CPU, which represents common calculating ability while the other one has an Intel Core 2 Quad Q9400 2.66GHz CPU, which represents high calculating ability. For each platform, we sign messages of 200 bytes and verify them in both single-thread and multi-thread.

Fig. 7 shows the signing performance of the two hardware platforms. The x axis represents the length of the key, while the y axis stands for the number of signing operated per second. Fig. 7(a) is the single thread mode condition, while Fig. 7(b) is multi-threaded mode.

As shown in the figure, the speed of signing decreases quickly with the increase of the length of keys, especially for multi-core platforms. When the key length is 1024 bits, the signing time goes below 500 per second. When the key reaches 2048 bits, even the high performance Q9400 platform in multi-threads mode can only finish 40 times singing per second.

Based on the facts that performance is not acceptable when the key's length is larger than 1024 bits, we set the key length to 512 bits in the application. Though PKI CA usually requires 1024 bits, and 2048 bits are suggested for better security, we believe 512 bits is safe enough for encrypting the ruleset library as we will refresh the key periodically.

Compared with signing operation, verification costs much less CPU computing resource. Actually verification performance is more important for the super node, because every node reports the change of their ruleset library after they downloaded data blocks. The load may become very large with the increase of peers.

Fig. 8 shows the performance of verification in two platforms. We can see that compared with signing operation, verification requires much less CPU computing resource. Fig. 8 (a) is in single-threaded mode and Fig. 8 (b) is in multi-threaded mode. It is showed that Q9400 platform can process 54054 verifications per second when the length of key is 512 bits. Since each *INFORM* message needs two verifications, the super peer can process as many as 27000 *INFORM* messages per second.

If each key is in the size of 512KB, and each peer sends a *INFORM* message every two second, the super peer will be able to serve about 216K peers at one time in the ideal case.

### B. The Communication Cost

According to the definition of message exchange protocols, the length of flag fields in a message is about 80 bytes. The *INFORM* message has the longest LOAD field, which is decide by the key length. For the worst situation in our system, the key is 512 bits, so LOAD field is about 100 bytes, and SIG field will be about 132 bytes. Then the total length of message is no longer than 300 bytes, which can be included in a single TCP or UDP packet. Even when the key is 1024 bits, one *INFORM* message could still be included in one packet.

Suppose a ruleset library file's size is 100 MB, and is divided into 200 blocks of 512KB, so one node needs 200 *INFORM* messages to download it. In one connection, these messages cost a bandwidth of 60KB. If a node keeps 50 connections with other peers, its total bandwidth consumption is about 3MB per second. This is acceptable comparing with the file size.

## VI. Conclusion

This paper presents a collaborative network security management system (CNSMS). CNSMS consists of one CMS and tens of or hundreds of NetSecu nodes, which are deployed at the vantage points to detect security event

ISSN 2348 – 7968

network-wide. CMS is the command and control center of CNSMS. For scalability and efficiency reasons, CNSMS is organized into a hybrid hierarchy and P2P topology and NetSecu nodes are aligned into two levels. In each level, the NetSecu nodes collaborate to share knowledge about events and policies with each other in a P2P way. The node in higher level acts as coordinator for nodes in lower levels. A special message protocol is designed for message exchange. To make identity trustworthy and message exchange secure, a PKI infrastructure is used to provide certificates for nodes, and each node use cryptography signatures and encryption to protect their messages' security.

The computation and communication cost of cryptography primitives in CNSMS is also evaluated. Then the parameters are adjusted according to the experiment results.

## REFERENCES

[1] "Clean-slate program," *http://cleanslate.stanford.edu/*.

[2] "Openflow switch," *http://www.openflowswitch.org/*.

[3] "4d project program," *http://www.cs.cmu.edu/ 4D/*.

[4] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Y. J. Zhan, and H. Zhang, "A clean slate 4d approach to netwok control and management," in *SIGCOMM Computer and Communication Review*, 2005.

[5] H. Yan, D. A. Maltz, T. S. E. Ng, H. Gogineni, H. Zhang, and Z. Cai, "Tesseract: A 4d network control plane," in *Proceedings of USENIX Symposium on Networked Systems Design and Implementation*, April 2007.

[6] R. Bye, S. A. Camtepe, and S. Albayrak, "Collaborative intrusion detection framework: Characteristics, adversarial opportunities and countermeasures," in *Proceedings of USENIX Symposium on Networked Systems Design and Implementation*, April 2007.

[7] F. Cuppens and A. Mige, "Alert correlation in a cooperative intrusion detection framework," in *Proceedings 2002 IEEE Symposium on Security and Privacy*, 2002.

[8] A. Hofmann, I. Dedinski, B. Sick, and H. de Meer, "A novelty-driven approach to intrusion alert correlation based on distributed hash tables," in *12th IEEE Symposium on Computers and Communications*, 2007.

[9] X. Chen, B. Mu, and Z. Chen, "Netsecu: A collaborative network security platform for in-network security," in *International Conference on Communication and Mobile Computing*, 2011, in press.