Detection of cyber attack through probability based Data mining technique

R.Kiran kumar[1]
Asst.Prof, Department of Computer Science, Krishna University Asst.Prof, Department

M.Nanchariah[2]
of Master of Computer Applications,
S V H College of Engineering

B.H.H.Pavan kumar[3]
Department of Computer Science,Krishna University

Abstract

This paper describes overall system architecture and design as well as data mining technique applied to achieve the research goal of the dissertation work, which was to design and implement a cyber defense technique suitable for Organization information system network by applying probabilistic data mining algorithm. This first paper describes the system design and architecture by providing a general description of the development environment and tools utilized in order to implement the system. It also includes list of all the database items developed for this dissertation work and their purposes in order to provide a clear understanding of the database items mentioned in this . It then categorizes the overall system into four main modules and explains them in detail. This paper describes the network traffic capture and storage module of the system which dealt with the capturing of network traffic header using Net flow tool

1. Introduction

This paper describes the notification or response module which was responsible for generating alert in the form of email message to the administrator immediately whenever an attack was detected. In of this, 3.2.4, paper describes the fourth module of the system whose functionalities were to correlate attacks and to generate attack signature for future use. This provides an overall system flow. Finally, this paper describes the cyber defense management site, which was designed to provide a secure web interface in order to facilitate authorized administrator to access and to manipulate network traffic database and attack information.

The cyber defense system was designed using MySQL relational database management system server version 5.1 available for Windows operating system. All the modules were implemented within the database where Net flow version 5 network traffic IP header information was collected using a Net flow collector named Ntop. The total system development was accomplished by developing various stored procedures, triggers and events necessary to perform all the operations including attack detection and Generation of alert message in real-time. Accessing database for attack information and manipulating data directly in the database would be inefficient and difficult from a remote location. In order to address this issue and to provide administrators a secure interface to interact with the database and to send them additional attack notification, a cyber defense management site was also designed using PHP version 5,MySQL 5.1 and Apache web server on local host. However, this management site was not intended as the primary alert notification tool but it was designed as a supporting tool for administrative management of the network traffic database. The cyber defense system architecture in this dissertation work consisted of the following four modules:

- Network traffic capture and storage module
- Attack detection module
- Response module
- Attack signature module

2. Network Traffic

Net Flow version 5 was used to collect Transmission Control Protocol (TCP) connection information also known as IP packet header generated from Cisco Internet Operating System (IOS) enable perimeter firewalls. Cisco firewalls are most popular and are widely used for perimeter security of networked environment. Net Flow is a protocol developed by Cisco Systems Inc. to capture packet level information. Initially, it was developed for network traffic monitoring purposes. Net Flow contains two main functionalities such as capturing and storing IP flow summaries and exporting captured IP flow data to specific NetFlow collector for further analysis.

2.1 Network Traffic Capture and Storage Module

A number of commercial and open source NetFlow collectors are available. NetFlow is becoming a de facto industry standard for network traffic flow export because of its flexibility and convenient export format. For example, NetFlow version 9 is becoming the Internet Engineering Task Force (IETF) standard as Internet Protocol Flow Information Export (IPFIX). NetFlow captures flows that contain network level IP address and transport level source and destination port numbers and creates unidirectional flow. Each IP flow or IP packet header is summarized by seven key fields combined together. Therefore, each unique IP flow captured by NetFlow version 5 is composed of the following seven key fields: source IP address, destination IP address, source port number, destination port number, protocol type, TOS Byte and input logical interface (if Index). Depending on the NetFlow 5 configuration, a flow may also contain some additional fields such as version format type, start and end time of a flow, number of packets in a flow, next hop number, bytes sent, bytes received etc. This information can be useful for various purposes such as accounting, and auditing. Above Figure shows the NetFlow export version 5 record format and header format field names and values. Table 2 shows list of IP header fields and descriptions.

For this dissertation work, only incoming flow records were considered because the scope of this dissertation was limited to defend cyber attacks originated from outside of a protected Organization network. The combination of the key seven fields and some additional generated fields such as flow time (first and last) and tcpflags were also considered as key characteristics to define a cyber attack. Table 4 shows the selected fields from NetFlow 5 captured IP headers utilized for this dissertation work. Ntop, a NetFlow collector was installed on a server running Windows Server 2008 operating system to collect network IP packet header flows generated by NetFlow version 5 from the perimeter of the Organization network. Ntop collected IP flows exported by NetFlow version 5 and stored them directly into a MYSQL database on the same server based on a predefined Perl script accompanied with Ntop. The packet flows were stored directly into a predefined table called flows in a predefined database ntop into MySQL database. Each incoming flow was an input to the CatchTheFlow attack detection algorithm described in the following

2.2 Attack Detection Module

The core of this module was CatchTheFlow algorithm. The input to the algorithm was the IP header fields collected from NetFlow by the traffic capture and storage module. The algorithm was trained by using normal Network traffic data collected from real Organization network which were free of attack. In the training phase, the algorithm calculated a normality score for each of the IP flows collected using a score schema described in the following sub. After calculating the normality scores for the entire normal IP header Flows, the lowest score in the training data was considered as the threshold. In the evaluation phase, any score below the threshold was considered as abnormal and as a possible attack. It was not possible to set up any false positive or false negative threshold in the evaluation phase because the network traffic data used in the evaluation was collected from a live Organization network in real-time. Lazarevic et al. Allowed 2% false alarm in the Evaluation of a suite of data mining algorithms for intrusion detection which utilized preprocessed simulated network traffic data. Each of the detected attack IP headers were passed to the response module.

2.2.1 CatchTheFlow Algorithm

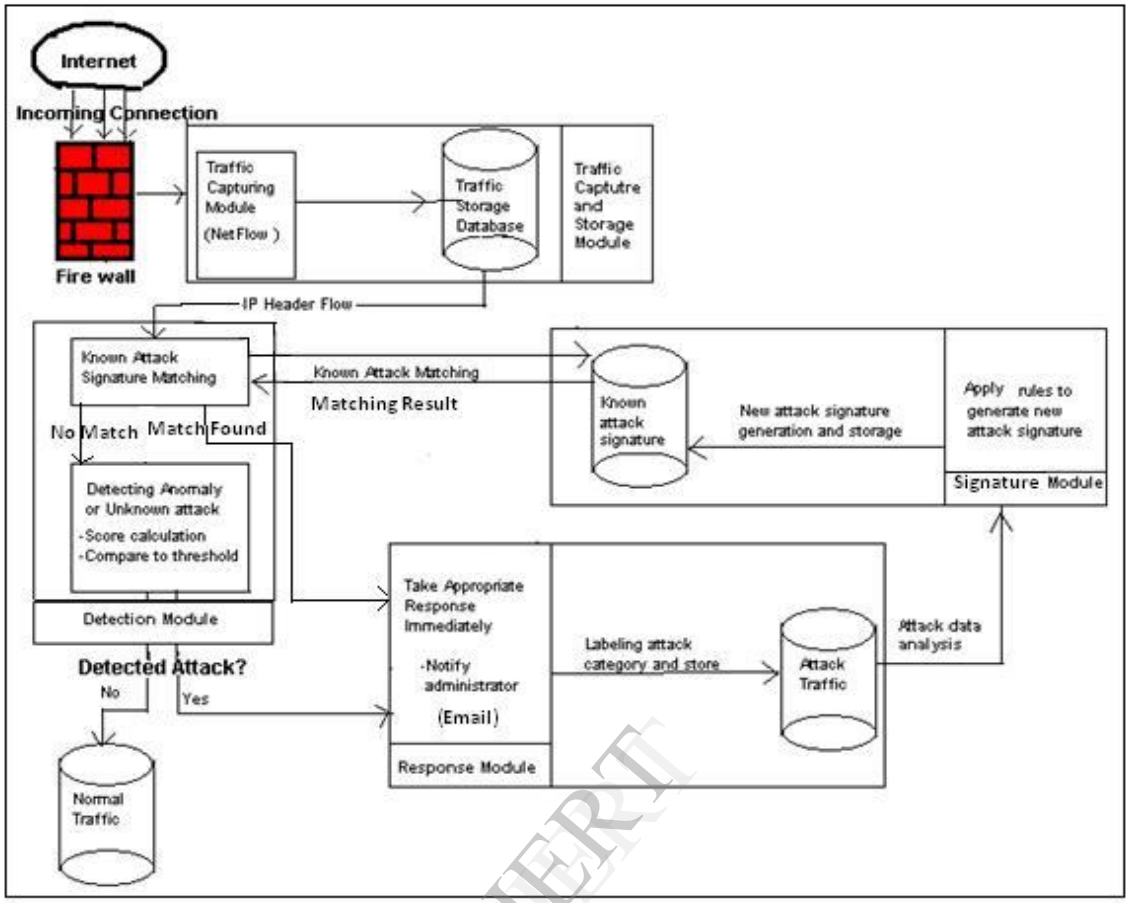
2.2.1.1 The Score Schema

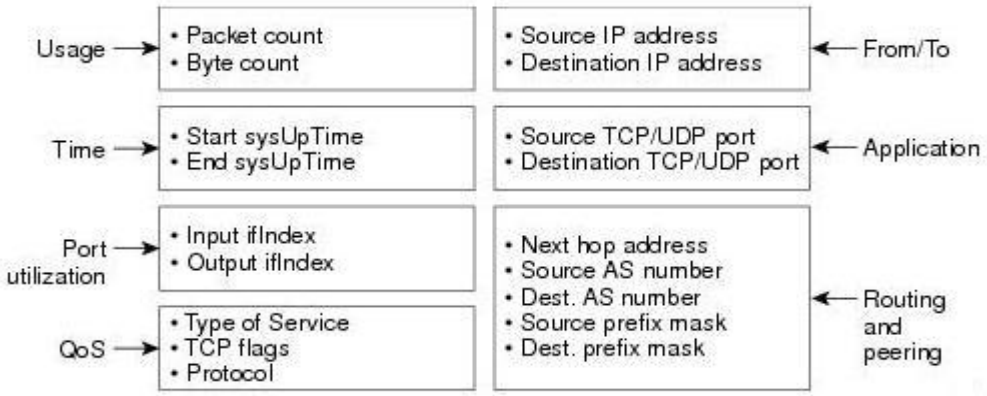
Let $p1$ be the trust value of the source IP address, which is a probability of how frequently the IP address was seen in the past one week normal data. More often an IP address was seen in normal traffic, the higher its trust value (higher value of p1) would be. An IP address that had not been seen before would have the lowest trust value. Therefore, higher $p1$ value would contribute more to the normality score of an IP packet.

$P = p_1 + p_2 + p_3 + p_4 + p_5 + p_6 + p_7$

So $p1$ was defined as follow for a source IP X:

Now, let $p2$ be the normality of a destination port Y communicating with a source port Z, that is how normal was the communication between Y and Z among all communications to Y in the normal traffic. The more frequent the communication between Y and Z in normal traffic was found, the higher it would contribute to the normality of a connection. So normality value of a connection to the destination port, $p2$, was defined as follows:Let $p3$ be the normality of a source port Z communicating to a destination port Y using a specific tcpflag value T. The value of T was an ex-ored value produced by the NetFlow 5 according to the flags of the packet. The probability of the combination of these three fields (Y,Z, and T) in the past week normal traffic would define how normal was the connection that contained these three field values (Y, Z, and T) among all connections between destination port Y and source port Z in the past week Normal data. So $p3$ was defined as the probability of a connection between a source port Z and a destination


Internet
Incoming Connection
Fire wall
Traffic Capturing Module (NetFlow)
Traffic Storage Database
Traffic Captutre and Storage Module
IP Header Flow
Known Attack Signature Matching
Known Attack Matching
Matching Result
No Match
Match Found
Known attack signature
New attack signature generation and storage
Apply rules to generate new attack signature
Signature Module
Detecting Anomaly or Unknown attack
-Score calculation
-Compare to threshold
Detection Module
Take Appropriate Response Immediately
-Notify administrator
(Email)
Response Module
Labeling attack category and store
Attack Traffic
Attack data analysis
Detected Attack?
No
Yes
Normal Traffic


Usage
• Packet count
• Byte count
• Source IP address
• Destination IP address
From/To
Time
• Start sysUpTime
• End sysUpTime
• Source TCP/UDP port
• Destination TCP/UDP port
Application
Port utilization
• Input ifIndex
• Output ifIndex
QoS
• Type of Service
• TCP flags
• Protocol
• Next hop address
• Source AS number
• Dest. AS number
• Source prefix mask
• Dest. prefix mask
Routing and peering

Port Y containing a tcpflags T among all connections between Y and Z in the past week normal traffic. Therefore, *p*3 was defined as

The higher value of *p*3 would contribute more towards the overall normality of a connection. Let *p*4 be the normality value of a connection from source port Z to a destination port Y among all connections coming from Z compared to the past week normal traffic. So *p*4 would define how common was a connection between a source port and a destination port among all connections originating from the source port. Therefore, *p*4 was defined as the probability of observing a combination of Y and Z in the normal traffic among all connections from Z in the past week normal traffic data. In the algorithm p4 was defined as follow:

The higher value of *p*4 would indicate more normality of connection coming from a source port to a destination port and therefore, it would contribute more towards the normality of a network connection.

Let *p5* be the normality value of a connection between a source port Z and a destination port Y using a specific protocol L. In almost all cases in IP network, NetFlow 5 would generate a value of 6 for the protocol field for all connections between Z and Y, which indicates TCP protocol. Any deviation from this protocol could be an indicator of possible spoof in the network traffic. A different protocol value in the protocol field would significantly lower the normality of a connection. Therefore, the value of $p_5$would contributes significantly towards spoof detection in the network traffic. In the detection algorithm, p5 was defined as follow:

Let p_6 be the normality value based on how many connections were made to the destination port Y. It was argued in this dissertation work that if too many connections were coming to a specific destination port, it would be abnormal. Too many connections coming to a specific destination port in a short period of time would likely be a port scanning attack. Therefore, the number of connections coming to a specific destination port for a specific period of time would be inversely related to the normality of the connection. For the algorithm in this dissertation work, number of connections came to a specific destination port in the last one hour of the arrival time of a new connection was considered. Therefore, if *t* was the total number of connections to the destination port Y in last one hour of the flow time of the current connection that was being processed, then it would contribute 1/*t* to the normality of the entire new connection.

Let *p7* be the normality value contributed to a connection based on the packet size sent in this connection and let *m* be the number of connections came to the Organization network in the last hour with a packet size of at least the largest packet size in the past week normal traffic. It was also argued in this dissertation that if there were too many connections coming from outside of the Organization network with larger packet sizes within a short period of time, it would be possible indication of traffic over congestion. In reality, many cyber attacks occur in the form of traffic over congestion to disable a network. In order to incorporate this characteristic, a normality value was assigned based on the number of connections came to the Organization network in the last hour with a packet size of at least the maximum packet size of the past week normal traffic. The higher the number of such connections came would contribute less in the overall normality of a connection.

Where *i* is the number of normality values contributing to the overall normality score of a connection. Higher value of P would indicate a higher likelihood of a connection being normal. In the training phase, each of the past week network traffic flows was assigned a normality score according to the above calculation of *P*. For the evaluation phase, the smallest normality score in the training phase was set as the threshold score in order to detect any cyber attack. Any new connection with a normality score smaller than the threshold would be detected as an attack. Statistical Packet Anomaly Detection Engine (SPADE), 87, 88 a Snort plug-in program, which is the anomaly detection module of Snort, is also based on probability model. It only focuses on connection between source port and destination port to calculate an anomaly score. Score calculations in the CatchTheFlow algorithm in this dissertation work was different than Spade because besides source port and destination port, it also applied source IP, TCP flag, protocol, flow time, and calculated field maximum packet size and maximum number of connections in a specific time period. The CatchTheFlow algorithm was based on data mining technique, which utilized the probabilistic score schema described above to calculate normality score for each network traffic flow it encountered and compared the score to the threshold. It was verified that the CatchTheFlow algorithm required a linear running time in the evaluation phase

2.2.2 Implementation of the Detection Module
Functionalities of the detection module was achieved by designing and implementing stored procedures get_weekly_training_data, training_score_calculation And core_on_new_connection; and database event named training and a database trigger named NEW_CONNECTION.

2.2.2.1 CatchTheFlow Algorithm for Training Phase
For gathering and training past week normal network traffic, a database event called training was designed which was set to execute every seventh day from a predefined beginning day. The event was designed to call another stored procedure called get_weekly_training_data which was designed to first delete all the previously trained normal data from flows_1 table and to gather all the normal traffic data from the new_conn_tbl (a table that stored all the processed new connections) and to store them into the flows_1 table. The stored procedure then called training_score_calculation procedure within the database to assign normality score to each of the newly gathered network traffic using the CatchTheFlow algorithm. It is important to mention that flows_1 table in the database was designed to store training data and the training data was previous week's attack free data automatically collected every seventh day from a predefined date.

2.2.2.2 CatchTheFlow Algorithm for Evaluation Phase
For evaluating the algorithm and processing every new connection coming from the network traffic capture and storage module, the algorithm was implemented within the database where captured Organization network traffic data were stored. To accomplish the functionalities of the algorithm, it was implemented as a stored procedure called score_on_new_connection within the MYSQL database and a trigger called NEW_CONNECTION was also designed to call the stored procedure whenever a new connection arrived in the database. The store procedure then assigned a normality score to the new connection based on the CatchTheFlow algorithm and compared the normality score to the threshold. Once the normality score of a new connection was at least equal to or greater than the threshold, the new connection was not labeled as attack and was simply stored in the new_conn_tbl (a table that stored all the processed new connections) table. If the value was smaller than the threshold, the new connection was labeled as „attack " and was stored into the new_conn_tbl table in the database with assigned score. The attack connection was then inserted into the attack_tbl (a table where all the attack traffic were stored) table. At this point the response module would be activated immediately.

2.3 Response Module
The response module in this dissertation work was designed to generate an alert message in the form of email message immediately when any new cyber attack was detected by the detection module. MYSQL 5.1 does not provide any message sending functionality to send message from the database. Therefore, the email messaging functionality of this module was accomplished within the MYSQL database by developing a database trigger and a stored procedure. The trigger named send_email was fired whenever an attack was inserted in the attack_tbl table. When activated, the trigger called a stored procedure named send_email_on_attack which was capable of sending an email from the database to administrator. This stored procedure first created an output file containing the network traffic attack information for a particular flow including attack time and also the predefined administrator's email address. The file was structured according to the standard email format outlined by email standard of *RFC* 2821.
The stored procedure then wrote the file into the Internet Information Service's (IIS) email Pickup folder on the local host in order to send it to the administrator. A real-time alert message was sent using this technique whenever an attack was detected. The alert message would help administrator to take immediate action according to the organizational policies in order to defend the attack.

2.4 Attack signature module
The purpose of this module was to generate attack signatures every week based on new attacks detected in that week to complement the attack detection module to match known attacks quickly without executing the entire detection algorithm. The functionality of this module was accomplished within the database by developing a database event that executed every seventh day from a predefined date. The event named nerate_signature was designed to call a stored procedure named generate_attack_signature when executed. The stored procedure checked the attack_tbl and calculated the frequencies of each of the following six tulles separately:

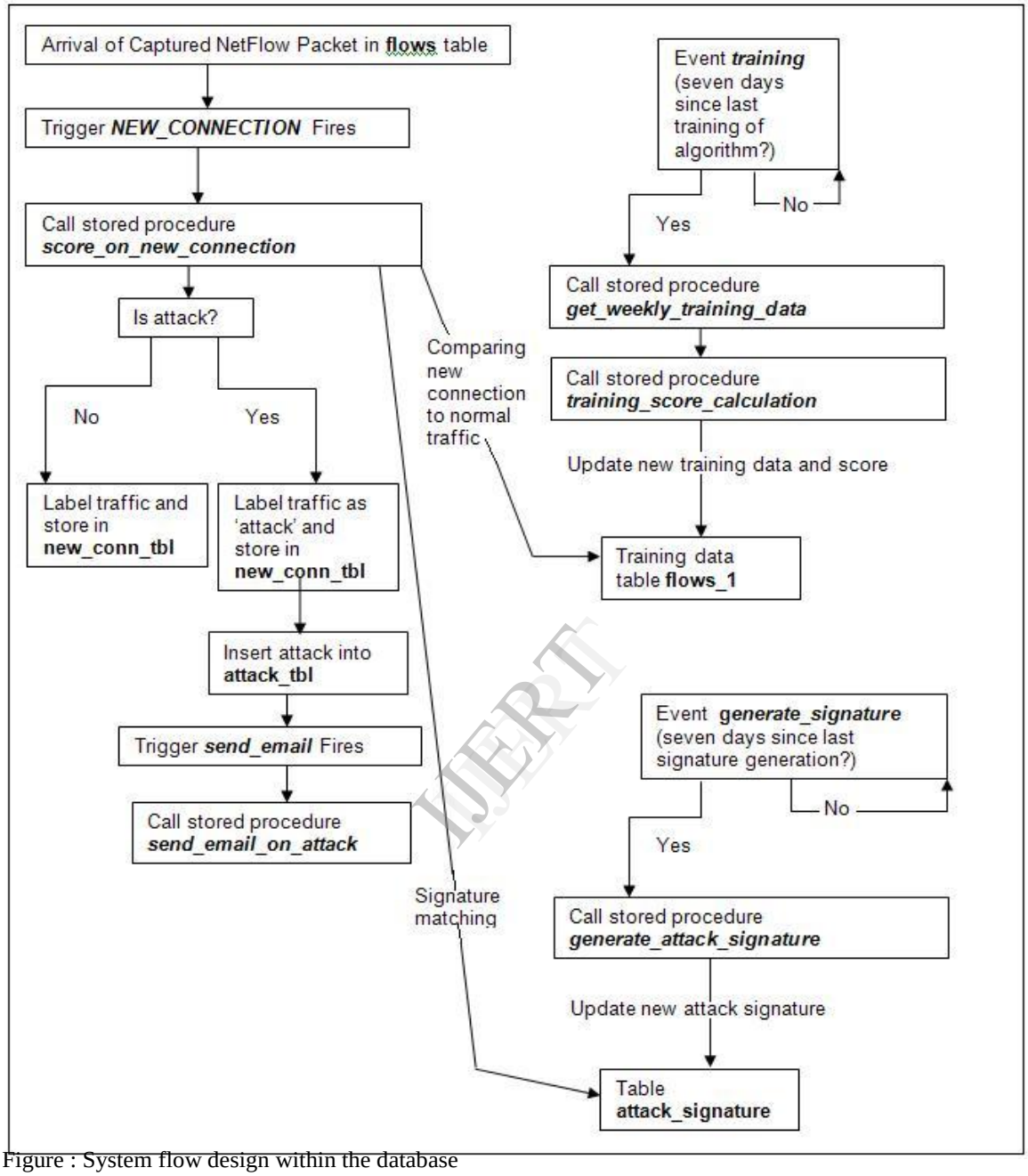


Figure : System flow design within the database

3 Overall System Flow Design

Above Figure illustrates the overall system design within the database and how the database items (tables, triggers, stored procedures and events) function within the database to accomplish the system development. For this dissertation work, it was considered to generate an attack signature for three occurrences of any of the tulles in the attack table. If three attacks occurred with the same combination of field values as any of the tulles, the stored procedure would generate an attack signature and would store it in the attack signature table in the database. The generated attack signatures were considered as known attacks. What differentiated these attack signatures from traditional predefined attack signatures widely used for signature based intrusion detection systems was that these signatures were generated based on the attacks detected within the database locally. When a new connection came to the database, the detection module first searched in the attack signature table to find a match according to each of the tulles mentioned above. If there was a match found, the detection module would directly activate the response module without going through the detection algorithm in order to make immediate alert notification.

4 Cyber Defense Management Site

The purpose of the cyber defense management site was to provide a secure web interface in order to facilitate authorized administrator with the capabilities of accessing and managing the database where incoming Organization network traffic data and attack information were stored and processed for cyber attack detection. . The management site was developed using PHP scripting language version 5 and MYSQL version 5.1 to access the Organization network traffic database running on a MySQL server version 5.1 on Windows 2008 Server.

5. Conclusion

This also provides descriptions of some of the fields of network interchange headers captured by Net flow and the fields selected for the algorithm development in this dissertation work. this paper describes the attack detection module, which was responsible for detecting any unusual network traffic by applying a probability-based data mining algorithm to detect any cyber attack by comparing incoming network traffic to normal network traffic in a Organization network environment. This also paper describes details of the detection algorithm effectively.

6. References

1. Lippmann R. The Role of Network Intrusion Detection. In Proceedings of the Workshop on Network Intrusion detection; Mar 19-20, 2002;Aberdeen, MD.

2. Lunt TF, Jagannathan R. A Prototype Real-Time Intrusion-Detection Expert System. IEEE Symposium on Security and Privacy; Apr 18-21, 1988; Oakland, CA.

3. Javitz HS, Valdes A. The SRI IDES Statistical Anomaly Detector. 1991 IEEE Symposium on Security and Privacy; May 1991; Oakland, CA.

4. Lunt TF. (1989). Real-Time Intrusion Detection. Com Sec J. 1989; VI(1): 9-14.

5. Anderson D, Lunt T, Javitz H, Tamaru A, Valdes A. Detecting Unusual Program Behavior Using the Statistical Component of the Next-generation Intrusion Detection Expert System (NIDES). SRI International Computer Science Laboratory Technical Report ; May 1995; SRI-CSL-95-06.

6. Lindqvist U, Porras PA. Detecting Computer and Network Misuse Through the Production-Based Expert System Toolset (P-BEST). 1999 IEEE Symposium on Security and Privacy; May 9-12,1999; Oakland,CA.

7. Sebring M, Shellhouse E, Hanna M, Whitehurst R. Expert Systems in Intrusion Detection: A Case Study. Proceedings of the11th National Computer Security Conference; 1998; Baltimore, MD.

8. Porras PA, Neumann PG. Emerald: Event monitoring enabling responses to anomalous live disturbances. In *Proceedings of the 20th National Information Systems Security Conference;* Oct 7-10, 1997; Baltimore, MD.

9. Neumann PG, Porras PA. Ex-perience with Emerald to date. USENIX-In Proceedings of the 1st Workshop on Intrusion Detection and Network Monitoring; Apr 9-12, 1999; Santa Clara, CA.

10. Clem A, Galwankar S, Buck G. Health implications of cyber-terrorism.Prehosp Disaster Med. 2003; 18 (3):272–275.

11. Mike C. Cyber Security. H&HN: Hospitals and Health Networks. 2004 May 1; 78(5):1068-8838.

12. HIPAA - General Information. Available at: http://www.cms.hhs.gov/HIPAAGenInfo/. Accessed Oct 20, 2008.

13. Richardson R. 2008 CSI Computer Crime and Security Survey. New York, NY: Computer Security Institute, 2008.

14. Layden J. Zombies attack Seattle hospital. Channel Register: Software & Security , May 5, 2006. Available at: http://www.channelregister.co.uk/2006/05/05/hospital_zombie_attack/.Accessed June 20, 2008.