Hybrid Modular Approach for Anomaly Detection

A.Laxmi Kanth
Associate Professor,
M.Tech (IT)
Sri Indu College of Engineering
& Technology, Sheriguda, IBP.

Suresh Yadav
Assistant Professor,
(M.Tech),B.Tech, MBA
Sri Indu College of
Engineering &
Technology, Sheriguda, IBP.

M.Sridhar
Associate Professor,
M.Tech (CSE)
Sri Indu College of Engineering
& Technology, Sheriguda, IBP.

ABSTARCT
The traditional approach for detecting novel attacks in network traffic is to model the normal frequency of session IP addresses and server port usage and to signal unusual combinations of these attributes as suspicious. Rather than just modeling user behavior, recent systems model network protocols from the data link through the application layer in order to detect attacks that exploit vulnerabilities in the implementation of these protocols. We describe modular approach for network anomaly detection. Our system analyses the network traffic at three different possible levels (packet, flow, protocol) with the help of three different modules. Total anomaly score is computed from the anomaly scores of the three modules using weighted attribute model. We detect 147 of 185 attacks in the DARPA off-line intrusion detection evaluation data set [1] at 10 false alarms per day (total 100 false alarms), after training on one week of attack-free traffic. We investigate the performance of the system when attack free training data is not available.

Keywords: Intrusion Detection System, Traffic, Attacks

1. INTRODUCTION
An intrusion detection system (IDS) monitors network traffic, operating system events, or the file system to detect unauthorized attempts to access a system. IDS's are classified as Network IDS, which monitors traffic to and from the host, or host based IDS, which monitors the state of the host, depending on the source of data it monitors. The two most common detection techniques are signature detection, which looks for characteristics of known attacks, and anomaly detection, which looks for deviations from normal behavior, signaling a possibly novel attack.

In this paper, we focus on network anomaly detection. Most network anomaly systems such as ADAM [2], NIDES [3], and SPADE [4] monitor most anomalous attributes like IP addresses, ports, and TCP state. This catches user misbehavior, such as attempting to access a password protected service (because the source address is unusual) or probing a nonexistent service. However, this misses attacks on public servers or the TCP/IP stack that might otherwise be detected because of anomalies in other parts of the protocol. PHAD [5], NETAD [6] monitors the entire data link, network, and transport layer, without any preconceptions about which fields might be useful. Our system monitors all header fields of network, transport layers and part of application data at different possible levels: packet level, flow level, protocol level using three different modules. It uses weighted attribute model to compute the total anomaly score from the anomaly scores of all possible attributes. We evaluate our system on the DARPA IDS evaluation data set [1], which simulates a local network under attack, to investigate its performance.

The rest of the paper is organized as fallows. In section 2, we discuss related work in anomaly based network intrusion detection. In section 3, we describe modular approach for network traffic anomaly detection. In section 4, we brief the 1999 DARPA off-line IDS evaluation process. In section 5, we test our system on the 1999 DARPA data set. In section 6 we summarize.

2. RELATED WORK
Early work in anomaly detection was host based. Forrest et.al [7] demonstrated that the system call sequences for processing attack packets deviate significantly from the normal pattern of system calls. Forrest detected these attacks by training an n-gram model (n=3 to 6) as the system ran normally. In [8], utilizing statistical characteristics of n-gram model is described. More recent work has focused on better models, such as state machines [9].

Network intrusion detection is typically rule based. Systems like SNORT [10] and BRO [11] use hand written rules to detect signatures of known attacks, such as specific string in the application payload, or suspicious behavior, such as server requests to unused ports. When a new type of attack is detected, new rules must be added to these systems. Anomaly detection systems such as SPADE [4], ADAM [2], and NIDES [3] learn a statistical model of normal network traffic, and flag deviations from this model. Models are usually based on the distribution of source and destination addresses and ports per transaction (TCP connections, and sometimes UDP and ICMP packets). But they differ in the attributes, they model. These systems use frequency-based models, in which the probability of an event is estimated by its average frequency during training.

NIDES, monitors ports and addresses. SPADE, a SNORT plug-in, monitors addresses and ports of inbound TCP SYN packets. By default, it models only the destination (server) address and port. ADAM combines an anomaly detector trained on attack-free traffic with a classifier trained on traffic containing known, labeled attacks. In addition to addresses and ports (as SPAD does), it also monitors subnets, TCP state flags. Sessions which do not match any learned category (normal or known attack) are flagged as anomalous (novel attack).

PHAD [5], ALAD [12], LERAD [13], NETAD [6] differ in the attributes that they monitor. PHAD (Packet Header Anomaly Detector) has 34 attributes, corresponding to the Ethernet, IP, TCP, UDP, and ICMP packet header fields. It builds a single model of all network traffic, incoming or outgoing. ALAD (Application Layer Anomaly Detector) models incoming server TCP requests: source and destination addresses and ports, opening and closing TCP flags, and the list of commands (the first word on each line) in the

application payload. Depending on the attribute, it builds separate models for each target host, port number (service), or host/port combination. Both of them uses time based modeling of the attributes. LERAD (LEarning Rules for Anomaly Detection) also models TCP connections, but samples the training data to suggest large subsets to model. For example, if it samples two HTTP port requests to the same host, then it might suggest a rule that all requests to this host must be HTTP, and it builds a port model for this host. NETAD models 48 attributes, consisting of the first 48 bytes of the packet starting with the IP header. Each byte is treated as one attribute. It also considers frequency of events along with time-based models.

PHAD, ALAD, and NETAD were tested on the 1999 DARPA off-line intrusion detection evaluation data set [1], by training on one week of attack free traffic (inside sniffer, week 3) and testing on two weeks of traffic (weeks 4 and 5) containing 185 detectable instances of 58 attacks. At a threshold allowing 100 false alarms, PHAD detects 54 attack instances, ALAD detects 60, or 70 when the results were merged with PHAD, LERAD detects 114 and NETAD detects 132.

3. MODULAR APPROACH FOR NETWORK ANOMALY DETECTION

We believe that most of the network based attacks can be detected by analyzing the network traffic at Packet level (e.g. Land), Flow level (e.g. udpstorm) and Protocol level (e.g. SYN flood). Our proposed system analyzes the network traffic at these levels with three different modules, Integrity check, Route statistic and Connecting/Connected/Termination modules. The overall architecture of our proposed system is shown in figure 3.1. These three modules analyze the network traffic simultaneously and assign anomaly score to the packets individually. Decision making/Response module calculates the total anomaly score of each packet after collecting the anomaly score of the packet from each module and raises anomalies for those packets, whose anomaly score exceeds a threshold limit. Reordering/de-fragmentation module assembles the fragmented packets and sends them to the Connecting/Connected/Termination, which assumes the same.

3.1. Integrity Check Module

This module detects anomalies in network packets at byte level. It assigns anomaly score only to the most interesting packets, after filtering out the uninteresting traffic. As most of the attacks are initiated against a target server or operating system, it is usually sufficient to examine only the first few packets of incoming server requests. Hence it filters out all non IP packets, the entire out going traffic, traffic related to the TCP connections initiated from internal network to outside the network, packets to high numbered ports, and packets not in the near start of the TCP connection.

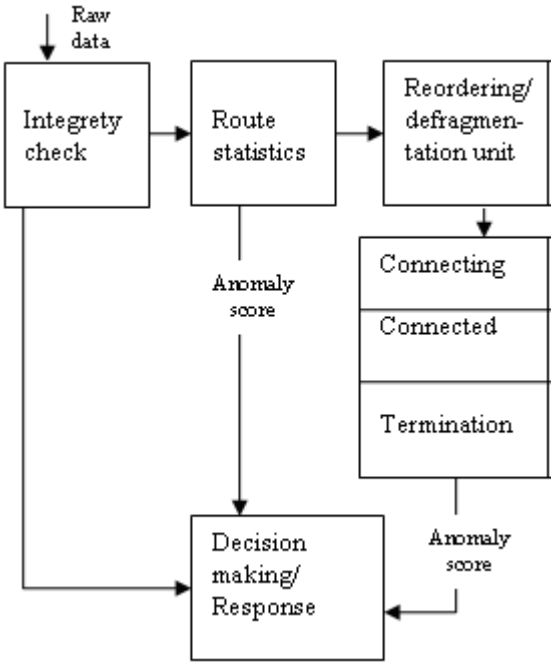

Figure 3.1: architecture of NIDS

This module models 48 attributes, consisting of the first 48 bytes of the packet starting with the IP header. If the packet is less than 48 bytes long, then the extra attributes are set to 0. During training period this module records all the values for each attribute that occur at least once. During testing period it uses these recorded values to assign anomaly score to the packets using the anomaly score function explained in section 3.6. Then it sends the anomaly score value to the decision making/response module for further processing.

3.2. Route Statistic Module

Route statistic module observes behavior on the monitoring network and adaptively learns what is normal for routes between any pair of nodes. This observed behavior will be reported as anomalous if it deviates significantly from the expected behavior. This module models all the header fields of IP layer at byte level. Each byte is treated as a nominal attribute with 256 possible values. And also monitors some conditional attributes. Major attributes modeled by this module are:

P(TTL | src IP, dest IP)
P(TOS | src IP, dest IP)
P(IP flags | src IP, dest IP)

P(dest IP): Probability of packets to a particular node

P(src IP):Probability of packets from a particular node

P(src IP, dest IP):Probability of packets between two given nodes

P(dest IP, dest port) :Probability of packets to a particular port of the node.

P(src IP, dest IP, dest port) :Probability of packets to a port on any node from a particular node

During the training period, this module calculates the values for the above attributes and records the statistics of these attributes. During the testing period, this module observes the values of the above mentioned attributes and assigns anomaly score accordingly using the anomaly score function explained in the section 3.6. Then it sends the anomaly score value to the decision making/response module for further processing.

3.3. Fragmentation reassembly / reordering module

This module reassembles the fragmented packets and sends them in the right order to the next module, as the next module assumes the same. Our system uses traffic normalizer [14] to remove the ambiguities in the network traffic. Hence there won't be any ambiguities in the fragment reassembly and reordering process. Thus this fragmentation reassembly and reordering process is unambiguous.

3.4. Connecting / Connected / Termination module

Every TCP session has to go through three stages connection establishment, connection maintenance and connection termination in that order. This module maintains the transitions of the TCP connections. Specification of the TCP state machine, as observed on a gateway connecting an organization's internal network to the Internet is given in the figure 3.2[15]. One of these components comes into picture dynamically according to the stage of the session at that time. Every valid TCP session has to go through these components in that order. During the process these components look for abnormal transitions. If any abnormality occurred it will be reported with some anomaly score.

This module models various attributes which correspond to common fields of the packet at Network (IP), and Transport (TCP) layer, and part of application layer at byte level. It models 48 attributes; consisting of the first 48 bytes of the packet starting with the IP header. Each byte is treated as a nominal attribute with 256 possible values. This module models all the 48 attributes for every transition separately. This also models the attributes and transitions separately for various application layer protocols like HTTP, SMTP, FTP and telnet to detect anomalies in those protocols also.

During training period, this module records all the values for all possible attributes to form the normal behavior of all the transitions for the above attributes. During testing period, this compares the observed attribute values with the recorded values for abnormalities and assigns anomaly score accordingly. Then it sends the anomaly score value to the decision making/response module for further processing.

3.5. Decision making/response module

This module receives the anomaly score from all the components and computes the total anomaly score of every packet. It reports anomaly if the total anomaly score of a packet crosses the threshold value. The threshold value can be specified by the security administrator. Total anomaly score of the packet is calculated using the weighted model, in which the modules will be assigned weights according to their detection rate. The module, which contributes more for true positives, is assigned more weight and the module, which contributes more for false alarms, is assigned less weight, so that the detection rate is improved with this model for the same false alarm rate. We investigated all the three modules for their detection rates and then we assigned weights to them. Hence the total anomaly score is (W1*A1 + W2*A2 + W3*A3)/ (W1+W2+W3).

3.6 Anomaly Score

Time based modeling of attributes was first used in PHAD. It assigns the score t*n/r to anomalous attributes, where *t* is the time since the attribute was last anomalous (in training or testing), *n* is the number of training instances, and *r* is the number of allowed values (up to 256). NETAD made

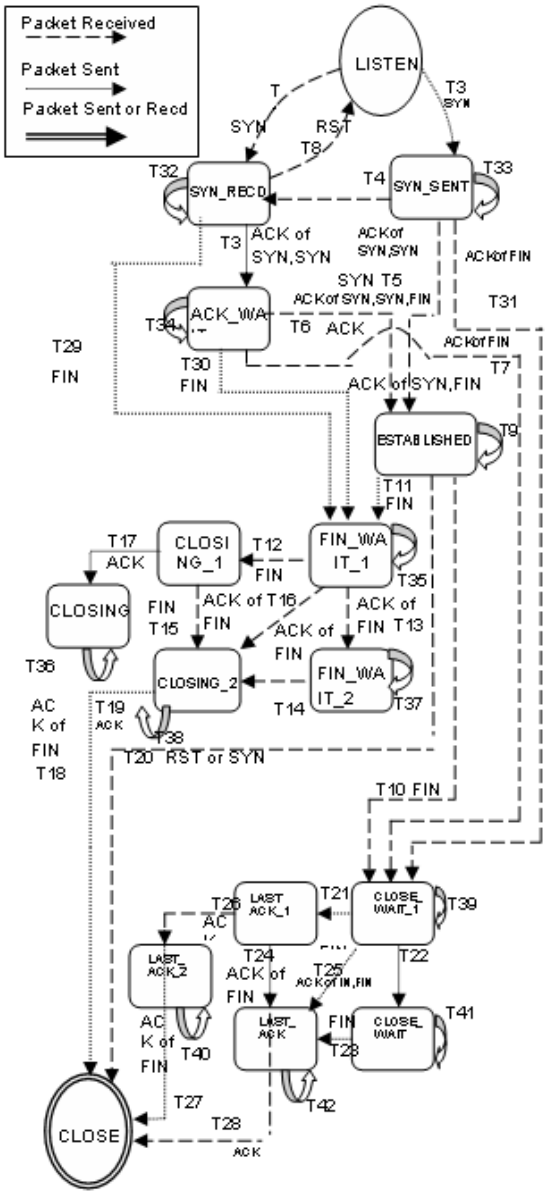

Figure 3.2: TCP state transition diagram

three improvements to the above anomaly score. First, n is set back to zero when an anomaly occurred during training. Second improvement is to decrease the weight of rules when r is near the maximum of 256. Third is that it considers the frequency of normal (not anomalous) events. Thus the NETAD anomaly score for an attribute is $t*n_a(1 - r/256) /r + t_i/(f_i + r/256)$. where n_a is the number of training packets from the last anomaly to the end of the training period, t_i is the time (packet count in the modeled subset) since the value i (0-255) was last observed (in either training or testing), and f_i is the frequency in training, the number of times i was observed among training packets.

Though NETAD considers the frequency of normal events it ignores the frequency of anomalous events. If a value does not occur at least once during training period, it is treated as anomalous value. NETAD assigns maximum score, through first component $t*n_a(1 - r/256) /r$, to all further occurrences of that value during testing period, ignoring its frequency. Our system considers the frequency of the anomalous values during testing period. We add another factor to the NETAD anomaly score, $T_i/(F_i+r/256)$, where T_i is the time(packet count) since the value i(0-256) was last observed during testing, and F_i is the frequency in testing, the number of times i was observed among the packets occurred till that time from the beginning of testing period. This model assigns highest score for values that occur rarely (small F_i) and lowest score for values that occur frequently (large F_i). Thus it reduces the anomaly score for normal values that have not occurred at least once during training period, but occur frequently during testing period. Thus the anomaly score S of an attribute is given by

$$S = t*n_a(1- r/256) /r+ t_i/(f_i+ r/256)+T_i/(F_i+r/256).$$

A criticism of PHAD and NETAD is that they do not have any preconceptions about which fields might be useful, and hence they give equal weight to all the attributes. This causes more false alarms to be generated because of uninteresting fields. To correct this we introduced weighted attribute model, in which we assign weights to the attributes based on their anomalous behavior. So that most anomalous attributes, like source IP address and TCP flags, get more weight and uninteresting attributes get less weight. We assign zero weight to the TTL field (simulation artifact). This reduces the false alarms score and increases the correct alarms score, so that more number of detections are possible at a given false alarm rate. Thus the anomaly score for a packet is WjSj, Where Wj and Sj are the weight and anomaly score of jth attribute respectively.

4. THE 1999 DARPA OFF-LINE IDS EVALUATION

In 1999, DARPA sponsored a project at Lincoln Labs to evaluate intrusion detection systems [1]. They set up a simulated local area network with a variety of different hosts and a simulated Internet connection and attacked it with a variety of published exploits. A number of intrusion detection systems were evaluated on their ability to detect these attacks given the network traffic (inside and outside the gateway), daily file system dumps, audit logs, and BSM (Solaris system call) logs in two phases. During the first phase, participants were given 3 weeks of data to develop their systems that included both attack free periods and labeled attacks (time, victim IP address, and description of the attack). During the second phase, about six months later, the participants were given 2 weeks of new data sets with unlabeled attacks, some of them new, and were rated by the number of detections as a function of false alarm rate. After the evaluation, the data sets and results were made available to other researchers in intrusion detection. Table 4.1 shows the official results of the top 4 systems in the original evaluation.

Table 4.1: Top results of the 1999 DARPA IDS evaluation at 10 false alarms per day

System	In-Spec Detections
Expert 1	85/169 (50%)
Expert 2	81/173 (47%)
Dmine	41/102 (40%)
Forensics	15/27 (55%)

The systems were evaluated on the percentage of attacks detected out of those they were designed to detect. None were designed to detect all attacks. Lippmann reports in [1] that 77 of the 201 attack instances were poorly detected, in that no system detected more than half of the instances of that type.

Our system examines only the inside network traffic logs because the inside data contains the evidence of attacks both from inside and outside the network, although we miss outside attacks against the router. Although there are 201 labeled attacks, the inside traffic is missing for one day (week 4, day 2) containing 12 attacks, leaving 189. There is also one unlabeled attack (*apache2*) and there are five external attacks (one *queso* and four *snmpget*) against the router which are not visible from inside the local network. This leaves 185 detectable attacks, of which 68 were poorly detected.

5. EXPERIMENTAL RESULTS

We trained our system on week 3 (7 days of attack free traffic) of the inside tcpdump files provided by DARPA [1], then tested the system on weeks 4 and 5 of the inside tcpdump files. We did not use week 2 (labeled attacks) because we do not look for any attack specific features. We used the same evaluation criteria for our system as was used in the original evaluation, counting an attack as detected if the there is at least one alarm within 60 seconds any portion of the attack, but counting all the false alarms separately. If there is more than one alarm identifying the same target within a 60 second period, then only the highest scoring alarm is evaluated and the others are discarded. This technique can be used to reduce the false alarm rate of any IDS, because multiple detections of the same attack are counted only once, but each false alarm would be counted separately.

We zeroed out the TTL (time to live) field value, which we believe to be simulation artifacts. TTL is an 8-bit counter decremented each time an IP packet is routed in order to expire packets to avoid infinite routing loops. Although small TTL values might be used to elude an IDS by expiring the packet between the IDS and the target [16], this was not the case because the observed values were large, usually 126 or 253. Such artifacts are unfortunate, but probably inevitable, given the difficulty of simulating the Internet [17]. A likely explanation for these artifacts is that the machine used to simulate the attacks was a different real distance from the inside sniffer than the machines used to simulate the background traffic.

Our evaluation of the system at 10 False alarms per day detects 147 of 185 detectable attack instances (80%). We avoid comparison of our system performance with the performance of the original participants. This would be biased in our favor as we had access to the test data, even though we did not use it to train the system or write attack-specific techniques, simply having this data available to test our system helps us.

5.1 Effects of the scoring Function

Table 5.1 shows the results of using various anomaly score functions to evaluate our system. It shows the number of detections by our system at various false alarm rates ranging from 20 to 5000. The combined scoring function (7) detects the most attacks at 100 false alarms, but all of the three components (2 or 4 or 6) do well by themselves. Function (4) gives better results than the model that considers only novel events. Function (6) gives best result especially at low false alarm rates.

It shows that the improvement, modeling frequency of anomalous events (Ti/(Fi+r/256)), to the anomaly score function of NETAD described in section 3.6, reduces the score of false alarms so that the detection rate is improved for a given false alarm rate. This component gives best result at very low false alarm rate. This indicates that addition of this factor improves the anomaly score of true positives and decreases the anomaly score of false alarms.

Scoring Function	20	50	100	500	5000
$\Sigma W_j*(t*n_a)/r$	61	92	119	148	152
$\Sigma W_j*(t*n_a)*(1 - r/256)/r$	66	110	126	149	152
$\Sigma W_j*t_i/(f_i + 1)$	54	81	97	130	158
$\Sigma W_j*t_i/(f_i + r/256)$	94	124	131	145	156
$\Sigma W_j*T_i/(F_i + 1)$	89	103	106	128	154
$\Sigma W_j*T_i/(F_i + r/256)$	96	106	115	138	155
Our Model:					
$\Sigma W_j*(t*n_a/r+t_i/(f_i+r/256)+T_i/(F_i + r/256))$	88	119	147	152	154

Table 5.1: Attacks detected at 20 to 5000 false alarms using various anomaly scoring functions.

5.2. Detections by Category

Table 5.2 lists the number of detections (at 100 false alarms) for each category of attack described by kendall [18]. Our system performs well on probe, DOS, and R2L attacks. Like most other network intrusion detection systems, it performs poorly on U2R attacks. Detecting such attacks requires the IDS to interpret user commands, which might be entered locally or hidden by using a secure shell. Our system detects most of these attacks by anomalous source address.

The category poorly detected includes the 77 (68 detectable) instances of attack types for which none of the original 18 evaluated systems in 1999 were able to detect more than half of the instances. Our system detects these at almost the same rate as other attacks, indicating that there is not a lot of overlap between the attacks detected by our system and the systems developed by other techniques (signature, host based, etc.). This suggests that integrating our system with existing systems improves the overall detection rate.

Attack Category	Detected at 100 False Alarms
Probe	32/36 (89%)
Denial of Service (DOS)	50/63 (79%)
Remote to Local (R2L)	45/49 (90%)
User to Root (U2R)	19/33 (55%)
Data	1/4 (25%)
Total	147/185 (78%)
Poorly Detected in 1999	46/68 (68%)

Table 5.2: Attacks detected by category.

5.3. Analysis of detected attacks

Table 5.3 lists the attacks detected by each module of the proposed system at 100 false alarms, when they evaluated separately followed by the total number of detections by the entire system. Detections field gives the number of detections by the module. Attack types detected field contains the number of attack types for which at least one instance is detected by that module. Most of the attacks are detected by more than one module. Hence the sum of the detections by individual modules is more than the number of detections by the system

Among the three modules Connecting/Connected/Termination module detects more attacks. This is because this module models the attributes for all the transitions of the TCP state transition diagram and also models the application layer protocols separately. Route statistic module detects more attacks if TTL value is not assigned to zero, because of the attribute P(TTL | src IP, dst IP).

Module	Detections	Attack types detected
Integrity check	39	16 -- Probe: insidesniffer, *ipsweep, mscan, portsweep, satan; DoS: *arppoison, pod, smurf, syslogd, teardrop, udpstorm; R2L: framespoofer, *netbus, netcat_setup,ppmacro; U2R:ffbconfig;
Route Statistic	33	15 -- Probe: insidesniffer, *ipsweep, *queso, *portsweep, mscan, satan; DoS: pod, smurf, Neptune, teardrop, mailbomb; R2L: named, guesstelnet, ncftp; U2R: ffbconfig;
Connecting/ Connected/Termination	121	49 -- Probe: insidesniffer, *ls_domain, mscan, ntinfoscan, *portsweep, *queso, *resetscan, satan; DoS: apach2, *arppoison, back, crachiis, *dosnuke, land, mailbomb, Neptune, processtable, syslogd, *tcpreset, *warezclient, warezmaster; R2L: dict, ftpwrite,guessftp, guesspop, guesstelenet, guest, imap, named, *ncftp, *netbus, *netcat, phf, ppmacro, sendmail, *sshtrojan, xlock, xsnoop; U2R: anypw, casesen, eject, fdformat, ffbconfig, *perl, ps, *sechole, *squlattack, *xterm, yaga.
Entire system	147	

Table 5.3: Number of attack types detected by each module at 100 false alarms

Five of 58 attack types are not detected by our system. Httptunnel is a backdoor which disguises its communication with the attacker as web client requests. Our system misses this because it does not monitor outgoing traffic or incoming client responses. Selfping and ntfsdos generate no traffic directly, but could theoretically be detected because they reboot the target, interrupting TCP connections. Snmpget is an external router attack, not visible on the inside sniffer. Loadmodule is U2R, thus hard to detect.

5.4 Effects of attacks in Training

DARPA provided training (contains no attacks) and test (contains some attacks) data explicitly. But it might not be possible to have explicit training and test data in real network settings. We just have network traffic, which could contain unknown attacks at any point of time. To study the performance of our system in real network settings, we evaluated our system for two cases. In first case we assumed that the rate of attacks is low (compared to the volume of normal traffic) and ran our system in the training mode for all the three weeks. Second case is the more realistic case where attacks might occur at any point of time. For this we ran our system in the training mode during the attack period (weeks 4-5) without using attack free data (week 3).

	Week 3	**Weeks 4-5**	**20**	**50**	**100**	**500**	**5000**
Normal	Train	Test	88	119	147	152	154
Case1	Train	Train	72	99	108	132	154
Case2	Not used	Train	58	89	102	122	152

Table 5.4: Attacks detected at 20 to 5000 false alarms when our system is left in training mode.

Table 5.4 shows the results for both the cases described above. If we run the system in training mode during attack period (weeks 4-5), then the anomalies present in the traffic will be added to the model and further instances of the same or similar attacks might be missed. There are 58 types of attacks present in the DARAPA data set. If all the instances of these attacks have the identical signatures, then we should not expect to detect more than one instance of each. But our system can detect 108 instances (at 100 false alarms) when it is left in training mode for all three weeks, and 102 instances for the second and more realistic case, which indicates that there are subtle differences between instances of the same type.

6. CONCLUSION

We described a network anomaly detection system that analyses network traffic at three different levels (packet, flow, protocol) with three different modules. We considered frequency of anomalous events to reduce the weight of the further occurrences of that event. It uses weighted attribute model to improve the detection rate for a given false alarm rate. We investigated all the attributes and assigned weights to the attributes according to their anomalous behavior. This reduces the anomaly score of false alarms generated by uninteresting fields and improves the score of true positives.

Our system performs well on the DARPA IDS evaluation data set, detecting 80% of the total attacks. It detects those attacks that were poorly detected in the original evaluation at 70%. None of the original systems detected more than half of the poorly detected attacks. This indicates, Integrating our system with the systems participated in the original evaluation might improve the detection rate considerably.

We must caution that the data provided by DARPA is synthetic. Although great care was taken to make the DARPA background traffic realistic, artifacts due to simulation errors (such as the TTL field) or overly clean background traffic might make attacks easier to detect. Furthermore, we have assumed that attack free traffic is available for training. This would not be true in a real environment. We have evaluated the system in training mode on the attack data (weeks 4,5) with out using attack free data to train the system and found that there is a 30% decrease in the detection rate.

REFERENCES

[1] Lippmann, R., et al., "The 1999 DARPA Off-Line Intrusion Detection Evaluation", Computer Networks 34(4) 579-595, 2000.

[2] Barbará, D., N. Wu, S. Jajodia, "Detecting Novel Network Intrusions using Bayes Estimators", First SIAM International Conference on Data Mining, 2001.

[3] Anderson, D. et. al., "Detecting unusual program behavior using the statistical component of the Next-generation Intrusion Detection Expert System (NIDES)", Computer Science Laboratory SRI-CSL 95-06 May 1995.

[4] SPADE, Silicon Defense, http://www.silicondefense.com/software/spice/

[5] Mahoney, M., P. K. Chan, "PHAD: Packet Header Anomaly Detection for Identifying Hostile Network Traffic", Florida Tech. technical report 2001-04, http://cs.fit.edu/~tr/

[6] M. Mahoney, "Network Traffic Anomaly Detection Based on Packet Bytes", Proc. ACM-SAC, 346-350, 2003.

[7] Forrest, S., S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, "A Sense of Self for Unix Processes", Proceedings of IEEE Symposium on Computer Security and Privacy, 1996.

[8] L Zhuowei, A Das and S Nandi, "Utilizing Statistical Characteristics of N-grams for Intrusion Detection", International Conference on Cyberworlds, Singapore, December 2003.

[9] Sekar, R., M. Bendre, D. Dhurjati, P. Bollineni, "A Fast Automaton-based Method for Detecting Anomalous Program Behaviors". Proceedings of the 2001 IEEE symposium on Security and Privacy.

[10] Roesch, Martin, "Snort - Lightweight Intrusion Detection for Networks", Proc. USENIX Lisa '99, Seattle: Nov. 7-12, 1999.

[11] Paxson, Vern, "Bro: A System for Detecting Network Intruders in Real-Time", Lawrence Berkeley National Laboratory Proceedings, 7'th USENIX Security Symposium, Jan. 26-29, 1998.

[12] Mahoney, M., P. K. Chan, "Learning Models of Network Traffic for Detecting Novel Attacks", Florida Tech. technical report 2002-08, http://cs.fit.edu/~tr/

[13] Mahoney, M., P. K. Chan, "Learning Nonstationary Models of Normal Network Traffic for Detecting Novel Attacks ", Edmonton, Alberta: Proc. SIGKDD, 376-385, 2002.

[14] Handley, M., C. Kreibich and V. Paxson, "Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics", Proc. USENIX Security Symposium, 2001.

[15] Sekar, R., A.Gupta, J.Frullo, T.Shanbhag, A.Tiwari, H.Yang and S.Zhou, Specification-based Anomaly Detection: A New Approach for Detecting Network Intrusions. CCS'02, Washington, USA, Nov. 18-22, 2002.

[16] Ptacek, Thomas H., and Timothy N. Newsham, "Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection", January, 1998, http://www.robertgraham.com/mirror/Ptacek-Newsham-Evasion-98.html

[17] Floyd, S. and V. Paxson, "Difficulties in Simulating the Internet." IEEE/ACM Transactions on Networking Vol. 9, no. 4, pp. 392-403, Aug. 2001. http://www.icir.org/vern/papers.html.

[18] Kendall, Kristopher, "A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems", Masters Thesis, MIT, 1999.

AUTHORS PROFILE

A.Lxxmi Kanth Associate Professor, M.Tech Information Technology, with 11 years of teaching experience.

Mr. M. Sridhar, Assistant Professor, B.E Computer science ,M.Tech (CSE) has 6 years of experience in teaching and the areas of interest are Data Mining, Databases, Advanced Data Structure, OOPS through JAVA.

Mr. Suresh Yadav , Assistant Professor, B.Tech. (Computer science & Engg.), MBA (E-Business) ,M.Tech (CSE) has 6 years of experience in teaching and the areas of interest are Data Mining, Datawarehousing, Artificial Intelligence, Databases, Network Security.