Cooperative Anomaly and Intrusion Detection for Alert Correlation in Networked Computing Systems

Kai Hwang, *Fellow IEEE*, Hua Liu, *Student Member* and Ying Chen, *Student Member*

Abstract: Network-centric computing systems manifest as Grids, clusters, Intranets, LANs, or P2P networks, etc. These distributed systems are subject to security breaches in an open network environment. Conventional *intrusion detection systems* (IDS) use the misuse model at the packet level. An *anomaly detection system* (ADS) follows a normal-use model at Internet connection level. We integrate these two alert systems into a *Cooperative Anomaly and Intrusion Detection System* (CAIDS). This integrated system detects not only known attacks but also unknown anomalies. The system integration is enabled by *Internet episode datamining, anomaly classification, alert correlation,* and *automated signature generation*.

We have tested the CAIDS performance at USC with an Internet trace of 23.35 millions of traffic packets, intermixed with 200 attacks from the Lincoln Lab IDS dataset. We achieved a combined detection rate up to 75% at a false alarm rate lower than 5%. These results are sharply improved from 38% in using the network IDS Snort and from 50% in using the ADS alone. Our system detects many novel attacks hidden in ***telnet, http, ftp, smtp, Email, pop3,*** and ***authentication*** services. The CAIDS offers correlated alerts to intrusions on distributed hosts or anomalies with disrupted connectivity in the component networks.

Index Terms: Internet datamining, network security, intrusion detection, anomaly detection, alert classification, signature generation, false alarms, machine learning, Grid computing, and distributed computing.

---

* **Manuscript submitted to *IEEE Transaction on Dependable and Secure Computing*, November 24, 2004. The paper is substantially extended from a preliminary version submitted to the *IEEE IPDPS-2005 Conference.* This research work was fully supported by an NSF ITR Grant ACI-0325409.** **The corresponding author : Kai Hwang. EEB 212, University of Southern California, Los Angeles, CA. 90089-2562. Contact him by Email: kaihwang@usc.edu or by Tel: 213 740 4470.**

1. Introduction

Network attacks may trigger *intrusion alerts* to host systems or *anomaly alerts* in Internet connections [6]. Intrusions are often checked at packet level over a single connection. On the other hand, anomalies are mostly observed at the network level involving multiple connections. Malicious intrusions to networked computing systems may destroy valuable hosts, network connections, and storage resources [6, 36]. To cope with these problems, we need to detect all intrusions on hosts and all connection anomalies jointly and cooperatively. The difficulty lies in fully capturing the Internet behavior [9, 11] to give correlated alert from attacks of different kinds from various unknown sources.

The need of a cooperative approach to solving the intrusion detection problem has been elaborated in [8, 20, 21, 27, 29]. In this paper, we present a new alert-correlation architecture with built-in cooperative mechanisms to enable the detection of intrusions and anomalies simultaneously. This work is significantly extended from our preliminary results reported in an IPDPS-2005 paper [16]. To protect ***telnet, http, ftp, smtp, pop3, Email,*** and ***authentication*** services, early detection of Internet anomalies in routers, gateways, hosts, and servers is a necessity. Our work aims at ecuring network-centric applications on computational Grids, clusters of computers, and public-resource networks [14].

1.1 Intrusion versus Anomaly Detection Systems

In this paper, we consider signature-matching *intrusion detection systems* (IDS), based on the *misuse model.* We design *anomaly detection system* (ADS) based on the *anomaly-based model* characterized in Table 1. The misuse IDS model matches attack signature with pre-stored signatures from known attacks. The detection is claimed successful, if the matched signature is found in the attack database. An IDS operates at the packet scanning level and fails to detect unknown attacks or some encrypted packets. The need of frequent signature update is a major drawback in using the misuse IDS [34, 38].

The *anomaly-based* ADS compares the incoming traffic records with the normal traffic profiles. An anomaly is revealed, if the incoming traffic pattern deviates from the normal traffic rules significantly. An ADS checks the connection patterns or protocol violations against normal traffic profiles to reveal the anomalies in connection episodes. This anomaly detection process is usually time-consuming and often triggers more false alarms than that of using the signature-matching IDS. However, the ADS is often stateful and better adaptive to changes in network or attack conditions [40].

Table 1. Comparison of Misuse Model and Anomaly Model for Intrusion Detection

Attack Characteristics	Misuse Intrusion Detection System (IDS)	Anomaly Detection System (ADS)
Profiling level	Packet scanning to reveal misuse, abnormal, and guilty patterns	Mining TCP/UDP/ICMP connections to save normal traffic profiles
Detection mechanisms	Signature matching with known attack patterns and misuse logs	Profile matching with association or episode rules against normal traffic
Implementation requirements	Signature database and signature update mechanisms	Mining of audit data, episode rule generation, and database update
advantages and shortcomings	Detecting known attacks with low false alarms, fail to detect unknown attacks	Detect all attacks with higher false alarms, adaptive to network changes
Representative Systems	Snort [35], JAM [22], ADCPAR [13], MADAM ID [23]	Fuzzy Logic IDS [4], ADAM [3], EMERALD [31]

Conventional IDS detects only *single-connection attacks* at the packet level. Unknown or burst types of attacks often involve multiple connections, which can only be detected by a well-designed ADS. Several representative IDS and ADS built in the past are listed in Table 1. They model either the attack behavior or the normal traffic, but not both. The IDS runs much faster with routine signature checking and thus is easier to implement. One obvious advantage of the ADS is the capability of detecting many unknown or multi-connection attacks. The alerts generated by the anomaly-based ADS and signature-based IDS must be correlated before they can be used, jointly. This alert correlation was not done between these two different systems in the past. This work offers the first integration solution to this problem.

1.2 Related Previous Works

In the past, *association rules* [1, 3] and *datamining techniques* [3, 4, 19, 22, 23, 29] were suggested to implement ADS. Lazarevic and Kumar et al. [21] have distinguished the differences between *single-connection* and *multi-connection* attacks. Their study reveals the tradeoffs between intrusion detection rate and false alarm rate. Both IDS and ADS are sensitive to attack characteristics, system training history, the services provided, and the underlying network conditions [8, 38].

Lee and Stolfo [22] have developed a level-wise datamining algorithm for building ADS. Fan et al. [10] extended Lee's work to discover accurate boundaries between known attacks and unknown anomalies. However, all previously designed ADS and IDS did not support each other due to different detection mechanisms and databases applied. Kaleton Internet [18] has built a prototype system by combining the two detection systems, but working independently. We consider close interactions and cooperation between the two subsystems.

The concept of *episode rules* started with the work of Mannila and Toivonen [26]. Subsequently, *frequent episode rules* (FER) were defined by Lee and Stolfo [23] and by Qin and Hwang [32] for anomaly detection against normal traffic profiles. We use the adaptive *base-support threshold* reported in [32] to mine normal traffic records. The FERs are derived from attack-free TCP, UDP, ICMP, or ARP profiles. Unfortunately, network anomalies were detected at the connection level. No packet-level signatures can be easily induced from the anomalous connections detected.

The *hybrid IDS* defined by security industry combines host-based IDS with network-based IDS. These hybrid IDSs do not combine the two different detection models. Others attempted to solve the intrusion detection and response problems at the same time [16, 20, 33]. Integration of IDS and access control techniques were also studied in [21, 39]. Alternatively, the IDS design could be inspired by analogies from the human immune system

[2]. The ADS must be designed to monitor the connection features at the network, transport, and application layers.

1.3 Organization of the Paper

The rest of the paper is organized as follows: In Sec. 2, we introduce the new integrated system approach being proposed. We describe in Sec. 3 the key concepts of FERs with some TCP connection examples. In Sec. 4, we describe the CAIDS simulator architecture and the experimental settings and attack datasets used. Section 5 presents how to apply the base-support Internet datamining for building anomaly-based IDS. Section 6 presents an alert classification algorithm for generating new signatures from detected anomalies. Section 7 shows the experimental results on detection rate and timing analysis. Section 8 reveals the effects of false alarms. Finally, we identify lessons learned and further research challenges.

2. CAIDS: A New Integrated System Approach

Our work was motivated to enhance cyber security by using complementary techniques to cope with both known and unknown attacks at the same time. We propose an integrated approach to building future cyber defense systems by combining all positive features in IDS and ADS. It is possible to have a win-win situation, in which the two subsystems act in support of each other to yield an improved detection result.

2.1 The New Integrated CAIDS Approach

The new idea is illustrated in Fig.1. We integrate the IDS and ADS by making them supporting each other. The signature-matching IDS (Snort) is connected in cascade with a custom-designed ADS. These two subsystems join hand to cover all traffic events initiated by both legitimate and malicious users. Single-connection intrusions, such as the **Pod, Dosnuke,** and **Portsweep** are detectable by the Snort IDS at the packet level. Multi-connection attacks must reply on the ADS to detect through datamining of the Internet traffic episodes.

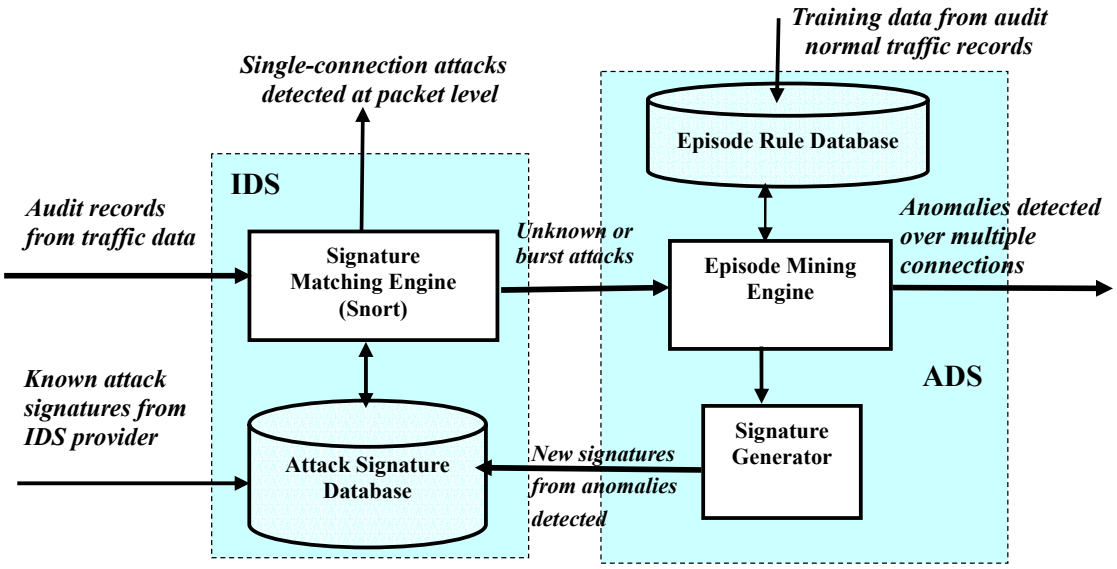


Figure 1. Basic idea of a cooperative anomaly and intrusion detection system (CAIDS), built with an intrusion detection system (IDS) in cascade with an anomaly detection system (ADS) and a feedback loop for automated signature update.

Unknown, burst, or multi-connection attacks are detectable by the ADS. These include the attacks **Apache2, Guesstelnet, Neptune, Udpstorm, Dictionary,** and **Smurf,** which cannot be detected by the Snort IDS. A novel *signature generator* is proposed to bridge the two subsystems. Details of generating new signatures from anomalies detected by the ADS are given in Section 6. Our new approach performs much better than the IDS or the ADS alone. Performance results will be presented and analyzed in Sections 7 and 8.

2.2 System Parameters and Raw Data Collected

In Table 2, we summarize important system parameters and raw data collected in 10 days of simulated Internet datamining and detection experiments. We use the IDS Snort and an ADS built at our Lab to process the incoming traffic sequentially. In total, 23.35M (million) of packets were processed by the Snort. Most traffic involves the TCP, UDP, and ICMP packets, plus smaller number of ARP and other packets.

The number of incidents detected by the Snort was recorded with 24,619 alerts or log files. The total time of using Snort to scan all traffic packets was recorded 282.73 sec. The

ADS was used in 333 minutes to generate 187,059 traffic episode rules. Most of these rules are normal from legitimate users, only a handful of 60 episode rules were detected as anomalies. We will use these raw data to plot and analyze the performance results of three different detection systems in subsequent sections.

Table 2. Packet Counts, Snort Log Files Reported, and Processing Times of the Snort and ADS in 10 Days of Internet Trace Experiments on the Simulated CAIDS at USC Internet and Grid Computing Laboratory during August 2004

Day	Packet Count (M)	TCP Packets	UDP Packets	ICMP Packets	ARP Packets	Other Packets	Snort Log files	Episode Rules by ADS	Snort Time (Sec)	ADS Time (Min)
Day 1	1.64 M	1.29 M	289,436	52,205	5,393	9,223	487	342	15.34	22
Day 2	1.30 M	1.22 M	13,127	850	19,183	57,453	830	1,711	12.39	19
Day 3	1.76 M	1.34 M	399,490	6,015	5,612	9,223	971	1,206	19.70	28
Day 4	2.35 M	1.69 M	651,985	1,322	6,716	9,223	3,301	3,106	37.35	45
Day 5	1.94 M	1.35 M	569,838	6,512	6,401	9,223	568	444	18.35	25
Day 6	2.29 M	1.41 M	851,999	9,829	5,684	9,224	903	1,487	27.83	33
Day 7	3.40 M	2.62 M	770,318	1,227	6,294	9,226	4,043	8,353	33.33	36
Day 8	2.08 M	1.51 M	561,295	1,119	4,854	9,225	1,773	2,517	26.83	30
Day 9	3.20 M	2.47 M	706,281	3,625	11,017	9,223	7,593	57,003	47.92	50
Day 10	3.39 M	2.72 M	674,102	1,249	6,071	9,223	4,150	111,430	43.69	45
Total	23.35 M	17.62 M	5.48 M	83,953	77,225	140,466	24,619	187,059	282.73	333

3. Basic Concept of Frequent Internet Episodes

An Internet *episode* is represented by a sequence of TCP, UDP, ICMP, or other connections. An episode can be generated by legitimate users or by malicious attackers. The *frequent episodes* are mostly resulted from normal users. A *rare episode* is likely caused by intruders. Our purpose is to build an ADS, that can automatically distinguish the rare or abnormal episodes from the normal or frequent episodes through a rule classification process.

3.1 Frequent Episode Rules

In Fig.2a, we show a typical stream of Internet traffic, represented by a sequence of *connection events* labeled in order by **E, D, F, A, . . . , F.** The time instants of these

connections in second are marked below the events. A *frequent episode* is a set of connection events, exceeding the occurrence threshold in a scanning window.

A *frequent episode rule* (FER) is generated out of a collection of frequent episodes. The FER is defined over episode sequences corresponding to multiple connection events in a roll. For the first window in Fig.2.a, we observe a 3-event sequence: **E, D, F.** We generate an FER: **E** $\rightarrow$ **D, F** with a *confidence level* computed by $freq(\gamma \cup \beta) / freq(\beta) = 0.8$ in Fig.2.b. Each FER must have a confidence larger than a *minimum confidence threshold* preset by the designer. If the event β occurs with 5% and the joint event of β and γ has 4% to occur, there is a (0.04/0.05) = 80% chance that both **D** and **F** events occur in the same window.

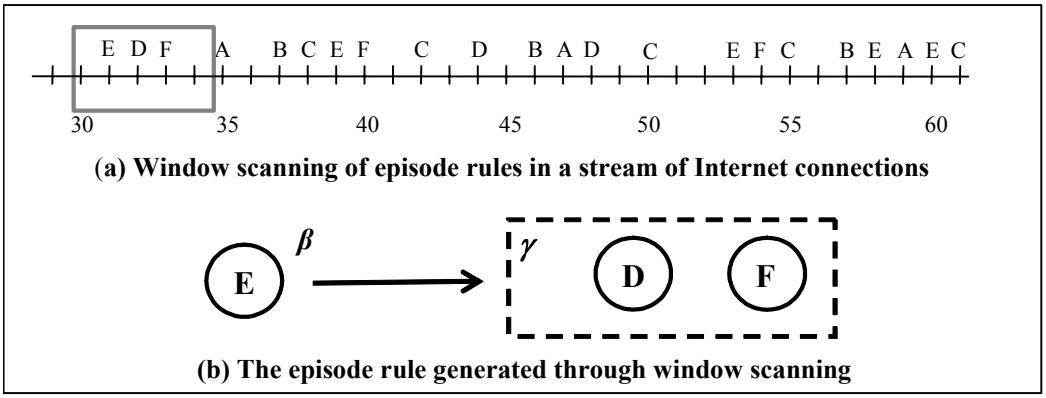


Figure 2. Scanning episodes in a stream of Internet connections using a window of 2 seconds.

In reality, the event **E** could be an authentication service requested at time 31 and presented by two attributes **(service = authentication, flag=SF).** The events **D, F** correspond to two sequential **smtp** requests denoted by: **(service = smtp) (service = smtp)**. We derive the following FER with a *confidence level* of $\boldsymbol{c}$ = 80% followed by two **smtp** services within a window $\boldsymbol{w}$ = 2 sec. The three joint traffic events accounts a *support level* $\boldsymbol{s}$ = 10% out of all network connections. The number 1200 indicates the minimal occurrences of the rule.

(service = authentication) $\rightarrow$ **(service = smtp) (service = smtp)**
(0.8, 0.1, 2 sec, 1200) **(1)**

An *association rule* is aimed at finding interesting intra-relationship inside a single connection record. The FER describes the inter-relationship among multiple connection records. Let $\boldsymbol{T}$ be a set of traffic connections and $\boldsymbol{A}$ be a set of attributes to affect the traffic feature characterization. In general, an FER is specified by the following expression:

$$\boldsymbol{L_1, L_{2,}, \ldots, L_n \rightarrow R_1, \ldots, R_m \quad (c, s, window, frequency)} \qquad \textbf{(2)}$$

where $\boldsymbol{L_i}$ $(1 \le i \le n)$ and $\boldsymbol{R_j}$ $(1 \le j \le m)$ are ordered connection events in a traffic record set $\boldsymbol{T}$. We call $\boldsymbol{L_1, L_2, \ldots L_n}$ the LHS (*left hand side*) *episode* and $\boldsymbol{R_1, \ldots R_m}$ the RHS (*right hand side*) of the *episode rule*.

3.2 Support and Confidence of Episode Rules

The s*upport* and *confidence* of an FER are two probability functions. We consider the *minimal episode occurrence* defined by Mannila and Toivonen [26] over the traffic stream.

$$s = \boldsymbol{Support}\ (\boldsymbol{L_1 \cup L_2 \ldots \cup R_1 \ldots \cup R_m}) \ge s_0 \qquad \textbf{(3.a)}$$

$$c = \frac{\boldsymbol{Support}\ (\boldsymbol{L_1 \cup L_2 \cup \ldots \cup R_1 \cup \ldots \cup R_m})}{\boldsymbol{Support}\ (\boldsymbol{L_1 \cup L_2 \ldots \cup L_n})} \ge c_0 \qquad \textbf{(3.b)}$$

The *support value* $\boldsymbol{s}$ is defined by the percentage of minimum occurrences of the episode within the parentheses out of the total number of traffic records audited. The *confidence level* $\boldsymbol{c}$ is the joint probability of the minimal occurrence of the joint episodes out of the support for the LHS episode. Both parameters are bounded below by the *minimum support threshold* $\boldsymbol{s_o}$ and the *minimum confidence level* $\boldsymbol{c_o}$, respectively, where $\boldsymbol{s_o}$ and $\boldsymbol{c_o}$ are preset by experienced designers. The window size is the scanning period of the window.

3.3 Pruning of Ineffective Episode Rules:

In general, we desire to have *shorter LHS, higher confidence level, longer RHS, no*

redundancies, FERs not implied by other rules, etc. These rule properties will accelerate the rule matching process. Several rule pruning techniques were developed in our earlier work [32]. One can apply the pruning techniques to remove redundancy and ineffective episode rules. This will significantly reduce the rule search space and thus results in higher efficiency in Internet datamining of episode rules.

For example, if **A → B, C** and **A, B → C** are both generated from a frequent episode **A, B, C** with a *Confidence* **(A → B, C)** = *freq* **(A, B, C)** / *freq* **(A)** and *Confidence* **(A, B → C)** = *freq* **(A, B, C)** / *freq* **(A, B).** Because *freq* (**A, B) < *freq* (A),** we keep only the first rule **A → B, C** with a shorter LHS and drop the second rule: **A, B → C**. Consider another situation that both **A → B, C** and **A → B** are in the normal FER database. We retain the first with longer RHS, because **A → B, C** is harder to generate than **A → B.** By the same reasoning, we remove another rule **A → C,** once the rule **A → B, C** is already established.

4. CAIDS Simulator Design and Attack Datasets

This section introduces the CAIDS simulator architecture, the attack dataset used, and the Internet trace applied in our security experiments to generate the performance results.

4.1 The CAIDS Simulator Architecture

Figure 3 shows the functional blocks in the CAIDS simulator built at USC. This system was simulated on a Dell Linux server. The system consists of a Snort 2.1 for detecting known attacks, an FER-based ADS was built locally to detect unknown anomalies, and the *signature generator* between them. The installed Snort 2.1 is a lightweight network IDS with over 2,000 signatures in its database [35]. The packet eliminator is used to remove those attacks already detected by Snort. This packet eliminator applies the timestamps to eliminate packets o avoid repeated work by the ADS. Packets are classified by their joint occurrence when multiple connections are involved.

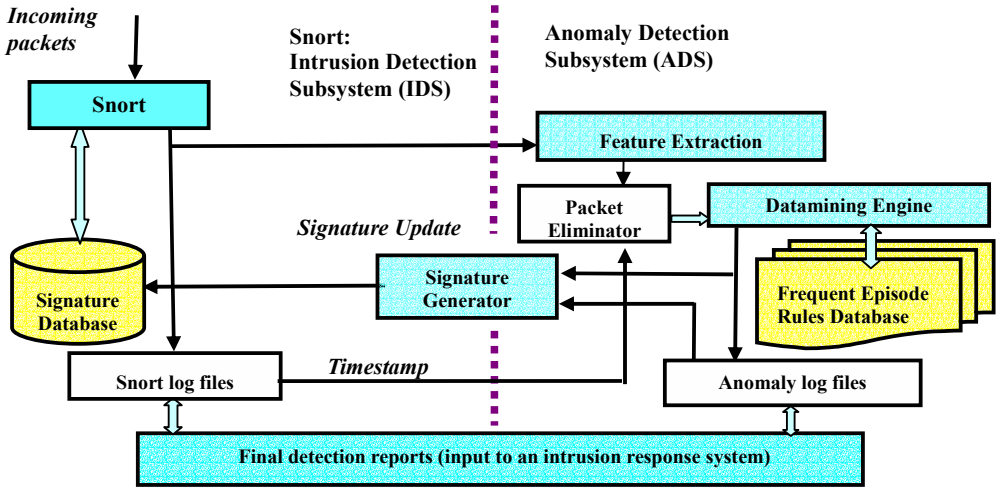


Figure 3. The architecture of the CAIDS simulator built with a 2,000-signature Snort IDS, followed by an anomaly detection subsystem (ADS) built with 60 FERs after two weeks of training over the Lincoln Lab IDS evaluation dataset.

A *datamining engine* was built in the simulator to generate the *FER database* and the *anomaly log files*. The size of the FER database is decided by the training data processed. We applied the MIT/LL weeks 1 and 3 attack-free, training dataset to obtain 60 FER rules for normal traffic. This FER database is sufficient for our simulation study. The storage requirements for both signature database and the FER database are rather small in our simulator. In real-life practice, the FER database may grow to several thousands of rules. We estimated that 256 MB memory and 1 GB of disk space should be sufficient to run a production CAIDS built with over 5,000 FERs.

4.2 Attack Spectrum in Internet Trace Dataset

The Internet trace is obtained from an ISP through the Los Nettos network in Southern California. It contains of 4 GB real traffic data including both attacks and background traffic. The attack spectrum across 10 days is shown in Fig.4. We use the toolkits by Mahoney and Chan [25] to mix USC Internet trace with the MIT/LL attack data [27]. The NetAttack adds real-life background traffic to the MIT/LL attack data to make up the entire Internet trace dataset used in experiments [24].

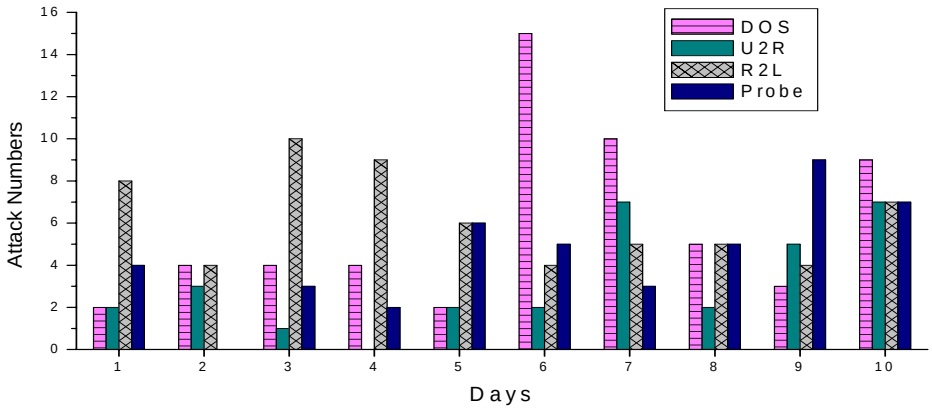


Figure 4. Attack spectrum in 10 days of Internet trace at USC mixed with over 200 attacks in the MIT/LL dataset.

There are in total 201 attacks in the MIT/LL data set, consisting of 58 DoS **(***Denial-of-Service*) attacks, 62 R2L **(***Remote to Local*) attacks, 31 U2R (*User to Root*) attacks, 44 PROBE or port scanning attacks, and 6 secret or unknown attacks. Some of the attacks repeat many times at different days. These attacks are collected at the weeks 4 and 5 datasets of the Lincoln Lab experiments.

5. Mining Internet Episodes for Anomaly Detection

Most datamining techniques exclude infrequent traffic patterns due to the use of normal traffic patterns. This will make the IDS ineffective in detecting rare network events. For example, authentication is infrequently performed in a common traffic. If we lower the support threshold, a large number of useless patterns will be generated.

5.1 Internet Connection and Traffic Features

The performance of ADS is directly decided by the features used in training, rule generation and classification. There are three types of features used in our ADS, namely the connection-level features, temporal statistics, and content features [23] as summarized in Table 3. All features are used in the training, rule generation, and alert classification processes. Connection level features are extracted from raw TCPdump. These features

provide basic information of each connection, and are used in the training process for FER generation. Temporal statistic features are based on a set of connections with the same reference feature, while content features are based on each connection. All

Table 3. Internet Connection Features, Temporal Statistics, and content features often used in Anomaly Detection by Mining Traffic Audit Records

Category	Feature Name	Short Description
Connection Features	**Timestamp**	Time when the connection begins
	Connection Duration	Duration of the connection in second
	Source_IP	Source IP address
	Destination_IP	Destination IP address
	Service port	Protocol used at a particular port
	Byte sent	Bytes sent by the originator
	Byte received	Bytes sent by the responder
	Flags	**SF:** Both SYN and FIN packets are known, **S0:** Only SYN packet seen in a connection, **REJ:** Connection rejected by the destination
Temporal statistical features	**Destination_Count**	No.of connections to the same destination
	Service_Count	No. of connections for the same service
	SYN_dst%	% of connections with same feature and SYN errors
	Service_dst%	% of connections per service to the same destination
	SYN_Service%	% of connections with SYN error to the same port
	Ave_duration	Average duration of connections for the same service
	Ave_byte_sent	Average bytes sent by the originator.
	Ave_byte_received	Average bytes sent by the responder.
	Source_Count	No. of connections from the same source
Content features	**Unauth_login**	**Unauth_login = True** for an unauthorized user login.
	Root_shell	**Root_shell=True** when a root shell is granted
	SU (Super_User)	**SU = True**, if the command **SU** is executed successfully.

Most features are applied in the FER rule training stage. The connection features are mostly used in FER rule generation. The flags **SF, S0** and **REJ** are used to signal special connection status. In the process of attack classification and cooperative detection, we need to use the temporal statistics, which represent traffic features of the connections within the window time. These temporal statistics can be used to improve the accuracy of the

classification model. The statistical values are often calculated based on the hosts and services requested [23, 32]. Content features are mainly used to detect U2R attacks.

5.2 Base-Support Datamining Algorithm for Anomaly Detection

Lee and Stolfo [22] have proposed a *level-wise datamining scheme* to solve the above problem by iteratively lowering the support threshold. A high support value is initially applied to find episodes related to frequent axis attribute values. The scheme iteratively lowers the support threshold by half. The process terminates when a very small threshold is reached. To use Lee's algorithm, it is likely that a common service appears in an episode rule with extremely low support value. Figure 5 shows the FER generation and rule matching process in anomaly detection, based on the base-support algorithm by Qin and Hwang [32].

We use the last two weeks' test data from MIT Lincoln Lab to produce the traffic profiles. When the anomaly is detected by Snort, the *timestamps* are passed to the packets eliminator and the relevant traffic flows are deleted. The rest traffics are passed to the ADS. When the traffic rule is not matched from the normal FER database, the *stealth alerts* are suspected. When the matched rule occurs beyond the threshold, the *massive alerts* are suspected. The stealth alerts are confirmed by checking some error flags and temporal statistics listed in Table 3. Otherwise, the traffic event is classified as normal.

5.3 Episode Training and Correlated Alert Types

To generate the FERs for the normal traffic profiles, the attack-free training connection records are fed into the datamining engine. We use the audit data sets collected from Week 1 and Week 3 of the MIT/Lincoln IDS Evaluation package [27]. We generated 60 FERs with the limited training time. We do not use FERs with extremely low support values. After finding FERs from each day's audit record, we simply merge them into a large rule set by removing all redundant rules using the pruning techniques developed in [32]. Towards this

end, we kept a keen interest on rare attributes of both single and multiple connections.

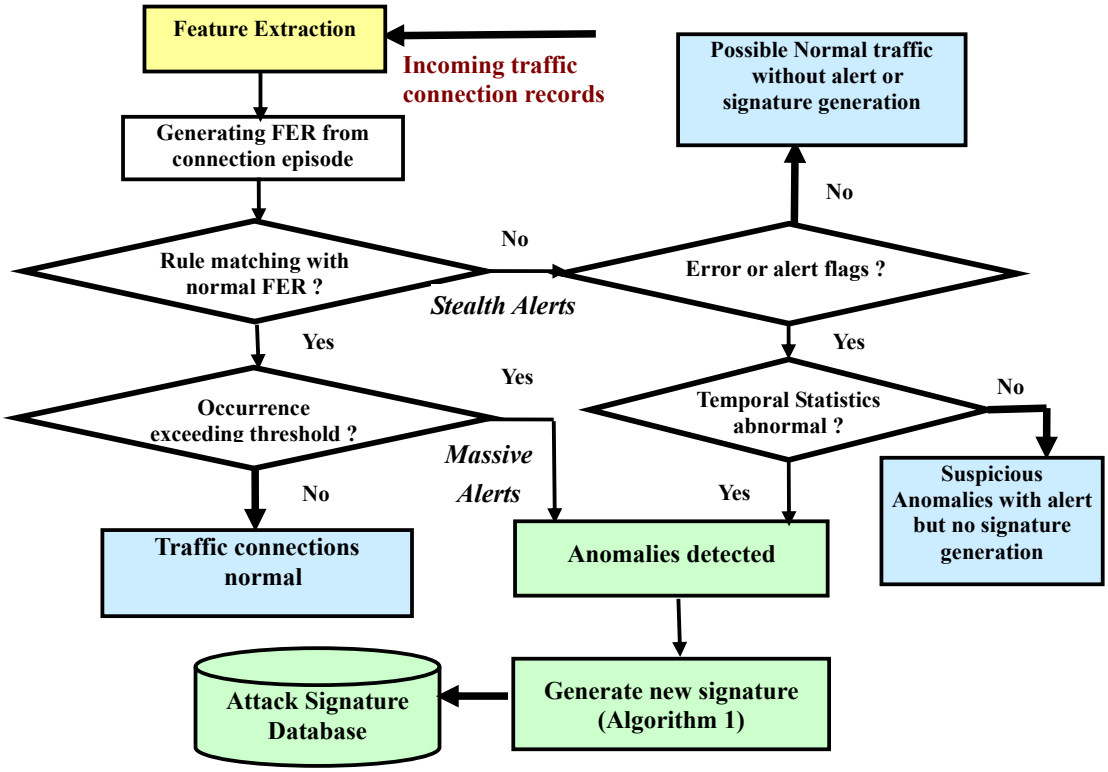


Figure 5. Matching with frequent episode rules to detect anomalies in traffic connections and automated signature generation from massive and stealth alerts.

We use the Bro toolkit [30] to extract useful features from traffic connection records. Snort sends its *Timestamp* to the packet eliminator in order to eliminate that packet from repeatedly checking by the ADS. A major task is to determine the minimum occurrence of the traffic episodes. We distinguish three alert types from three possible sources : the *packet-level alerts* from the Snort, the *stealth alerts* after no matching rules, and the *massive alerts* due to burst attacks. The ADS generated 187,059 traffic episode rules in 10 days.

Interesting connection features to be checked include those listed in Table 3. When the matching FER is not found, we classify the episode an anomaly. Another problem is that a single attack may last for a long time. To solve this problem, we use the connection sequence numbers to check connections with the same destination, instead of using the timestamps. The temporal statistics are collected from the network connection features.

A *relaxation factor* β is defined by the scaling ratio of the minimal frequency F of an FER in a window to the maximum occurrences M of the same FER rule during the training period. This parameter decides the *threshold* value actually used to determine an alert. The factor affects the false alarm rate illustrated in Fig.6.

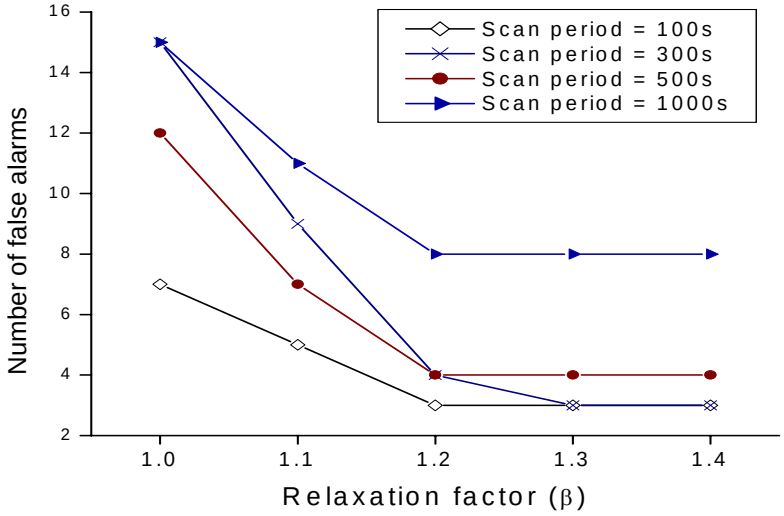


Figure 6. Effect of relaxation factor β on false alarms in an anomaly detection system. The critical value $\beta = 1.2$ was set for all window sizes based on experiments.

The maximum false alarm is observed when $\beta = 1$ and usually we choose $\beta \geq 1$. The threshold M is scaled by the relaxation factor β as follows:

$$F > \beta M \qquad (4)$$

When the condition in Eq.(4) is met, we signal an anomaly alert. We need to adjust the value of β to reduce the false alarm rate. When β exceeds a certain limit, the false alarm rate will not be reduced further. Our experiments indicate that such a critical value of β should be set at 1.20 as in Fig.6 for all window sizes.

5.4 Episode Rules for Attacks and Alert Log Files

To apply a FER in a window period, we associate the minimum occurrence number F of the FER with the rule. For example, a **SYN flood** attack is detected by checking the

number of SYN packets without seeing any FIN packets destined for the same IP from various sources. Thus the router buffer gets flooded and all future services denied. The **SYN flood** attack is specified below by an FER consisting of three repeated **http** requests:

(service=http, flag=S0) → (service=http, flag=S0)
(service=http, flag=S0) (0.1, 0.8, 100 sec, 150) **(5)**

This FER corresponds to the case of an attacker keeping sending **http** requests to the same destination. All **http** connections are SYN packets and no FIN packets were found. The **flag=S0** signals that only SYN packets were seen in the TCP connections. With a 100 sec window, this episode rule has occurred F = 150 times. Suppose in our normal rule database, there are M =100 occurrence of this rule and the relaxation factor β = 1.2. The frequency F=150 > 1.2 x 100 = 120. This is sufficiently high to trigger the **SYN flood** alert.

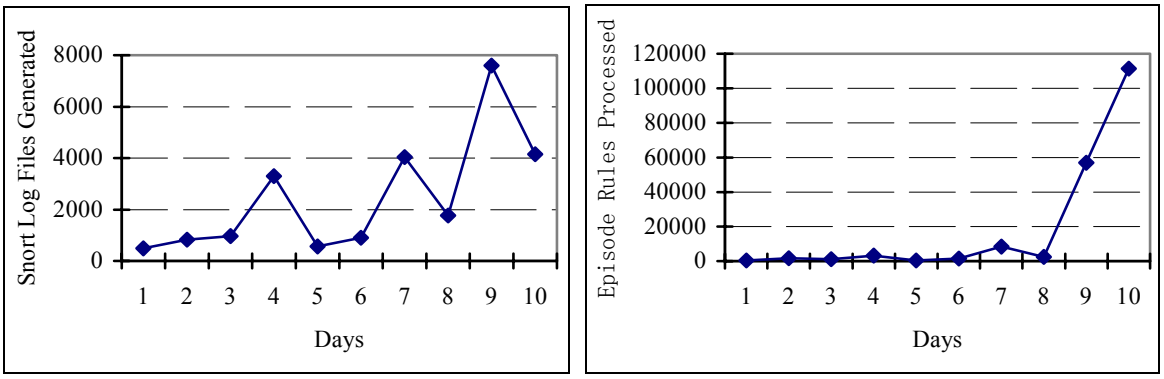
Consider another example of a Probe attack, **IPSweep.** The attacker launches a surveillance sweep to identify vulnerable machines to stage the attacks. To detect the **IPSweep** attack, one needs to trace on many Ping packets destined for almost all machines on the target network, all coming from the same source, say **172.16.118.60** as shown below. An example FER for describing an **IPSweep** attack is given below:

(service=ICMP, src=172.16.118.60, dst=any) → (service=ICMP, src=172.16.118.60, dst=any) (service=ICMP, src=172.16.118.60, dst=any) (0.1, 0.6, 200 sec, 289) **(6)**

A **Mailbomb** attack sends many messages to the same host machine with the intention to flood the target's mail queue. This kind of attack is characterized below by a FER. Millions of repeated mail messages could come from an attacker to the same target machine within a short period of time. This FER is resulted from the attacker passing the authentication check and then launch the **Mailbomb** attack, repeatedly. The window period is 200 second and there are 1089 regular mail requests sent to the victim host **172.10.5.49.** We can thus raise an anomaly alert of the **Mailbomb** attack with confidence.

(service=authentication) → (service=smtp, dst=172.10.5.49)
(0.1, 0.7, 200sec, 1089) (7)

In 10 days of Internet trace experiments, we have processed 23.35 millions of packets, consisting of TCP, UDP, ICMP, and other packets as summarized in Table 2. The number of log files reported by the Snort is totaled at 24,619 incidents distributed over 10 days as plotted in Fig.7(a). Many new or unknown attacks were forwarded to the ADS for detection.. With a 300 sec window, 64 anomalies were detected out of 201 attacks. Meanwhile, only 11 false alarms were observed over all traffic records in 10 days.



(a) Snort log files generated **(b) Episode rules processed by ADS**

Figure 7. Snort log files generated and the number of episode rules processed by the ADS in 10 days of CAIDS experiments

In Fig.7(b), we plot the number of episode rules processed by the ADS in 10 days of traffic trace. Some episode rules appeared with different frequencies in different windows. The rule generation is the most time-consuming part of the ADS. The time used to compare the network traffic rules with the normal rules is comparable with Snort. In total, there are 187,059 episode rules generated in 333 minutes spread over the 10 days. But, most of the rules are generated at the last two days as shown in Table 3 and in Fig.7(b).

6. Signature Generation from Classified Alerts

Automated signature generation is desired in Snort or other misuse-based IDS. We

desire to extract useful attributes from the anomaly rules to generate some new signatures to add into the Snort database, automatically. The *signature generator* was designed to insert new signature into the Snort database [35]. The most important information of a Snort signature is embedded in its rule header. The rule is made of 4 fields covering the *rule action, protocol, source* and *destination*.

By default, a Snort rule has 5 actions to take: namely *alert, log, pass, activate,* and *dynamic.* Generating signature at packet level is similar to the original Snort signature formation. To generate Snort signature for the anomaly FERs detected, we have to extract at least the *source address*, *destination address*, and the *services* provided in the connections. We need also the *connection count* during the scanning window period. Other axis attributes may include the *bytes sent or received*, *special flags* raised, etc. as listed in [32]. Packet filtering discovers some anomalies in single connections.

6.1 RIPPER Classifiers for Alert Classification

Burst attacks [21] result in two subclasses of alerts: the *stealth alerts* and *massive alerts* shown in Fig.5. When the traffic FER does not match with any normal FERs, we may generate the stealth alerts. Massive alerts take place when the matched FER exceeds the minimum threshold applied in traffic datamining. We generate the signatures for these two anomaly types using the RIPPER classifiers, which are based on an IREP (*incremental reduced error pruning*) machine learning algorithm introduced by William Cohen [7].

Burst attacks include both stealth alerts with small traffic flows and massive alerts with high traffic flows. All the packet or connection records must be labeled with the same identity to associate with an anomaly detected by the ADS. We have to extract some common features (Table 3) from the audit records with the same labeling. The next step is to check the error flags or some temporal statistics to generate the attack signature.

Massive alerts are often resulted from a large amount of traffic flows in the anomalous connections. RIPPER classifiers are used to predict attack type and help extract interesting features to generate high-level signatures for a pre-defined class of attacks. Classifiers are based on machine learning to assign traffic records to one of a set of predefined target classes. Attack class could be any of the 4 classes (DoS, R2L, U2R, and Probe). Table 4 shows four example RIIPPLR classifier rules for 4 major classes of attacks.

Table 4. Example RIPPER Rules for Alert Classification

Attack Type	RIPPER Classifier Rules	Short Description
SYN flood (DoS)	**SYN flood :- Service=http, flag = s0, Destination_count>1000.**	For all **http** connections to the same IP destination, only SYN packet seen, and the connection exceeds 1000, leading to a **SYN flood** DoS attack
Overflow (U2R)	**Overflow :- Service=telnet, Unauth_login=true, Root_shell =True, SU= False.**	An unauthorized user logged in and a **root shell** is granted without a successful **'su'**, leading to an U2R **overflow** attack.
Passwordguess (R2L)	**Telnet_Guess :- Service=telnet, flag=Rej, SYN_Service% > 80%.**	For all **telnet** connections to the same IP, the percentage of rejected connections exceeds 80%, thus leading to a **password guess** attack.
IPSweep (Probes)	**IPSweep :- Service=icmp, diff_dst% > 80%, source_count > 500.**	All ping packets have the same source IP address, more than 80% destination IP addresses are different, thus leading to an **IPSweep** attack.

6.2 Signature Generation from Alert Classifiers

We use the RIPPER classifiers [7] to help generate attack signatures automatically. Tracing the flowchart in Fig.5, we face the situation of massive alerts triggered by the matched episode rules exceeding the threshold frequency. First, we have to check different intrinsic features and content features, such as *the number of failed login attempts, root-shell access or not.* Then, additional information (e.g., count and percentage) are considered. Finally, we use the classifiers to predict the most suspicious attacks. Table 4 summarized 4 RIPPER classifier rules for 4 major classes of attacks.

Algorithm 1 specifies the procedure to generate the attack signatures out of anomalies detected from massive alerts. We use the destination IP address as a reference attribute. All

connections related to the same FER should have the same destination IP. Labels are used to bind traffic connections to specific FER. The stealth attacks often involve only smaller number of connections. Massive attacks trigger larger volumes of connections. Only the anomalies exceeding certain threshold can be distinguished with unique signatures. It is possible that only a few signatures are generated for a large number of similar anomalies.

Algorithm 1: ***Signature Generation from Anomaly Alerts***

Input: Labeled connections matching normal FER database and exceeding occurrence threshold

Output: **New signatures generated to add into the existing Snort database**

Begin For each FER having the same destination IP address,

1. Calculate statistical features from all relevant audit traffic records:

Count (Destination_IP), to the same destination IP address

Service_count (port), using the same port number,

Ave_duration (port), using the same port number,

SYN_dst_percent (Destination_IP), % of connections having SYN errors,

Service_dst_percent (Destination_IP, port), % of related connections,

Ave_byte_sent, average bytes sent by the originator of the connection,

Ave_byte_received, average bytes sent by the responder.

2. Choose the right RIPPER classifier to predict the alert type

3. Find all relevant connections affected by the anomaly detected

4. Extract common features in all associated connections

5. Generate the signature to uniquely identify the attack

EndBegin

The purpose is to generate new signatures for the Snort to use in future attacks. Labeling makes it possible to extract common features and to conduct a second pass of the datamining process to verify the new attacks detected. The Ripper rules in classifiers are very important to extract signature features. In the case of massive alerts, the flag and temporal information are used to distinguish different attacks under the same attack type.

6.3 An Illustrative Example and Experimental Results

In using Algorithm 1, we need first determine all the relevant features, extract

common features of the connections, and then provide these feature information to the RIPPER classifiers. An example RIPPER classifier rule (10) is given below to illustrate the signature generation process. The selected classifier helps predict the attack class and extract the common features from all 150 connections.

SYN flood : - Service = http, flag = s0, count > 100 Classifier Rule (10) (8)

The expression to the right of the operator :- is the classification rules consisting of three conditions separated by commas. This RIPPER rule means that if all three conditions: **service=http, flag=s0,** and **count > 100,** are met, then this is an SYN flood alert. Table 5 lists all 150 connections involved with this anomaly exceeding the FER threshold of 100.

Table 5. All TCP Connections Associated with the SYN Flood Episode Rule

Duration	Protocol	Bytes sent	Bytes rec'd	Destination IP	Source IP	flag	FER Label
0.08 sec	http	0	0	010.020.030.040	172.016.112.050	S0	10
0.05 sec	http	0	0	010.020.030.040	172.016.112.050	S0	10
0.01 sec	http	0	0	010.020.030.040	172.016.112.050	S0	10
…..	…..	…	….	…	….	…	…
0.08 sec	http	0	0	010.020.030.040	172.016.112.051	S0	10
0.01 sec	http	0	0	010.020.030.040	172.016.112.050	S0	10
0.01 sec	http	0	0	010.020.030.040	172.016.112.049	S0	10
0.01 sec	http	0	0	010.020.030.040	172.016.112.050	S0	10

When the ADS sees more than 100 SYN packets sending to the same port 80 on the same host **(010.020.030.040)** within the same window of 100 sec, it is detected as an SYN flood attack. An alert is thus raised. A Snort signature is thus generated below by extracting common features and using some of the temporal statistics. Each distinct massive attack matches a separate RIPPER classifier. The **$HOME_NET** denotes all possible IP addresses inside an intranet. The ***threshold: type both, track by_dst, count 100, seconds 100*** was used to raise the alert within the time interval of 100 sec.

Alert tcp ! $HOME_NET any → $HOME_NET 80 (***msg: "SYN Flood Alert Generated by ADS"; flags: S; threshold: type both, track by_dst, count 100, seconds 100*** **)** **(9)**

From the FER in Eq.(6), we identify one suspicious machine with an address 172.16.118.60, which keeps sending the Ping packets specified by request service ICMP with port number 7 to varies destinations. In a window of 200 sec, this FER occurs 89 times, which is relatively a high frequency. Suppose in our normal FER database, the maximum allowed frequency is 20, and the relaxation factor β is 1.2, since 89 > 20 x 1.2 = 24, we have to alert this anomaly and generate the corresponding signature using algorithm 1.

The curve in Fig.8 shows that 52 out of 64 anomalies were identified as candidate FERs, by which we can generate new attack signatures to add into the Snort database. In the best case, all candidates generate new signatures successfully. This will add up to 52 more signatures on top of 2000 signatures already in the Snort database. Signatures generated by both stealth and massive alerts will be added into Snort's signature database.

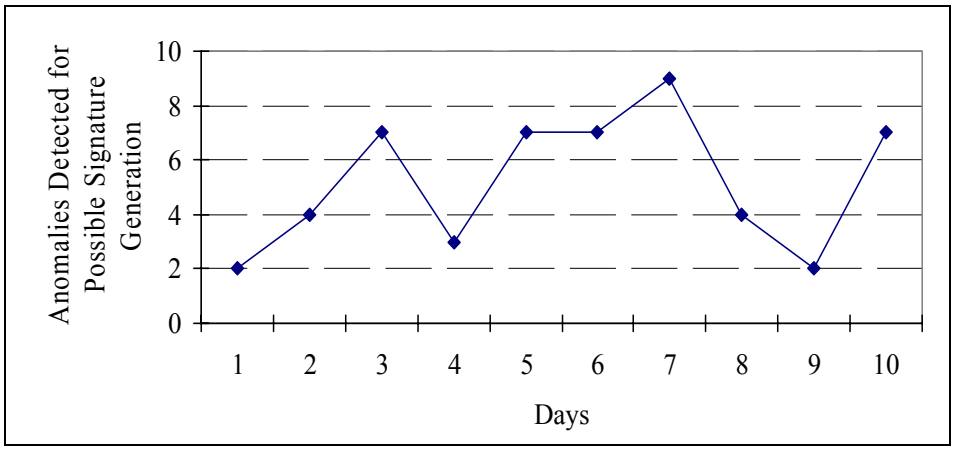


Figure 8. The number of new signatures generated from anomalies detected by ADS in 10 days of CAISD simulation experiments.

We have to improve the IDS capability to detect some new attacks. Only unknown attacks are forwarded to the ADS for follow-up actions. We still see attacks detected by both the IDS and ADS, but the overlapped cases are rather small. One drawback in adding more

signatures to the IDS database is the increase of false alarms, because those anomaly-induced signatures may not be accurate enough to capture all unique features in unknown attacks

7. Measured Detection Results and Timing Analysis

The detection performances of the IDS (Snort), the ADS, and the CAIDS are reported below. The Snort detects well-known intrusive attacks, while the ADS detects anomalous Internet connections. The CAIDS intends to cope with both types of attacks. Through interactive machine learning, the combined system has enhanced its sensitivity to detect all kinds of intrusions or anomalies, effectively. More than 200 attacks were tested on the simulated CAIDS in 10 days of experiments.

7.1 Performance Metrics of Detection Systems

Three performance metrics are used to present the detection performance results. All three are measurable in our experiments. The *detection rate* is formally defined below:

$$\textit{Detection rate} = d / n \qquad (10)$$

where ***d*** is the number of attacks detected successfully and ***n*** is the number of all possible attacks including known and unknown attacks. The *false alarm rate* measures the percentage of false positives among all normal traffic events. A formal definition is given by:

$$\textit{False alarm rate} = p / m \qquad (11)$$

where ***p*** is the total number of false-positive alarms and ***m*** accounts for the number of normal traffic events. The third measurement is the *ROC,* standing for *Receiver Operating Characteristic*. Here, the receiver refers to the client or the network security manager. The ROC curve plots the detection rate with respect to the variation of the false alarm rate.

ROC = A 2-dimensional plot of the detection rate against the false alarm rate **(12)**

Other interesting performance metrics include the *processing times,* the *detection speed*, and

the *system efficiency* of an IDS or ADS, etc. Some of the above measures are plotted and discussed in subsequent sections.

7.2 Intrusion Detection Rate

Figure 9 plots the daily average detection rates of using three detection systems, separately. The top curve shows clearly that the CAIDS outperforms both Snort and the ADS on all 10 days. The ADS also performs better than the Snort, because many of the MIT/LL attacks are unknown to the Snort. The CAIDS has an enhanced performance by capturing all attacks by either Snort or ADS.

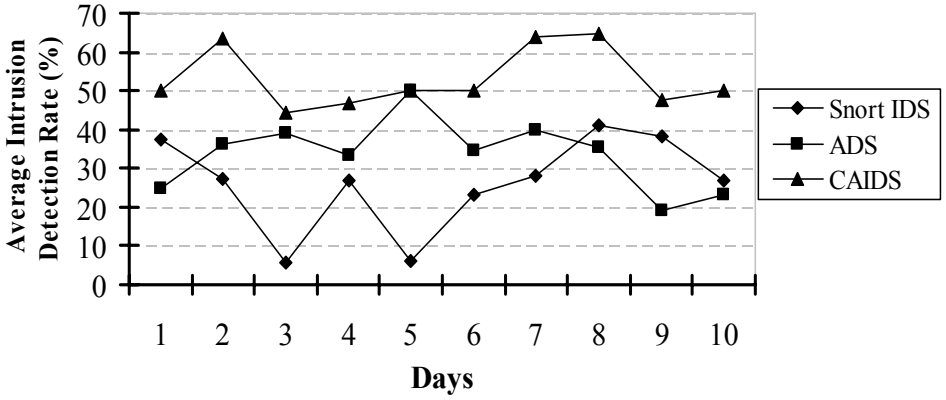


Figure 9. Daily average detection results in using three different intrusion detection systems: Snort, ADS, and CAIDS, separately

With automated signature generation, the false alarms in using the ADS are reduced, since the Snort has a higher accuracy with almost no false alarms. The major gain in using an expanded Snort signature database lies in speeding up the whole detection process, since the Snort has lower overhead, compared with that of using the ADS. We plot in Fig.10 the *intrusion detection rates* of four attack categories in using the detection systems, separately.

Each bar in Fig.10 represents the whole range of detection rates measured in 10 days of experiments. The top of the bar is the maximum value and the lower end is the minimum rate in 10 days. The Snort performance varies from zero to rather high, depending on whether the attack is known or not. Meanwhile, the ADS is effective to detect some unknown attacks,

which cannot be detected by Snort. The CAIDS bars have rather high minimum and maximum detection rates for the DoS, R2L, and Probe attacks. For the U2R attacks, the Snort performs the worst and ADS and CAIDS both are low. The message being conveyed here is that the CAIDS outperform the Snort and the ADS at both best and worst cases.

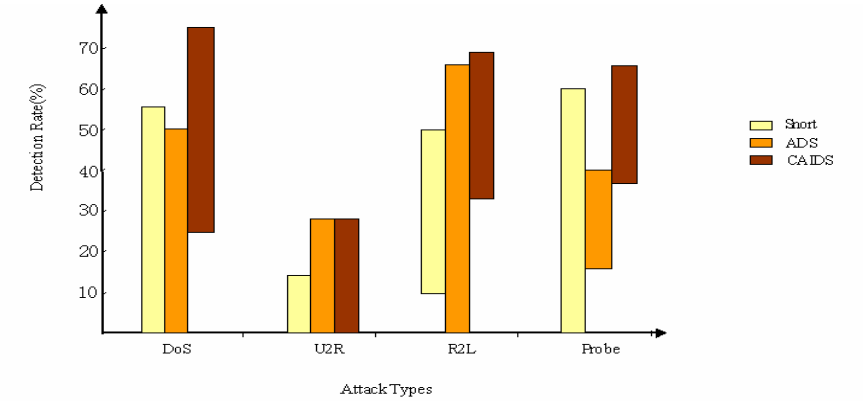


Figure 10. Range of detection rates of various attack types over 10 days of experimentation in using Snort, ADS, and CAIDS, separately.

In Fig.11, we plot the *average detection rates* over 10 days. The "Total" corresponds to the average detection rates across all attack types. We observed the higher detection rate of the CAISD in all attack categories except the U2R attacks. The highest detection rate of CAIDS is 75% and the minimum is 30%. The detection accuracy of ADS fluctuates due to the appearance of higher false alarms. The ADS performs even lower than Snort in detecting the PROBE attacks. All three systems perform poorly on the U2R attacks.

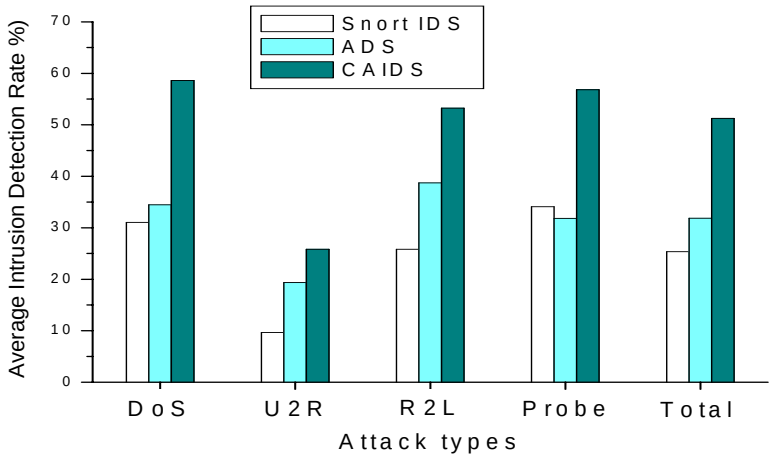
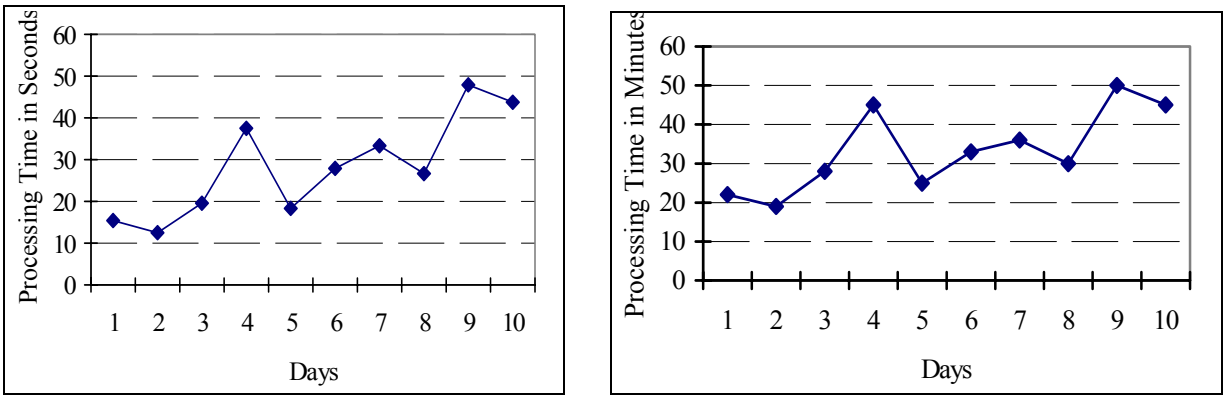


Figure 11. Average detection rates of various attacks in using Snort, ADS, and CAIDS, separately

7.3 Timing Analysis of Snort and ADS Performance

The Snort is very fast in processing traffic packets as shown in Fig.12(a). To detect 24,614 attacking incidents over 23.35 M packets in 10 days, the total time spent by Snort was 283 seconds. This implies a *Snort processing speed* of 86.4 K packets/sec or an *average detection performance* of 8 intrusions/sec. With a window size of 300 sec, the ADS generated 187,059 traffic episode rules in 333 minutes. This implies 9.4 rules/sec on the average.



(a) Snort processing time in seconds **(b) ADS processing time in minutes**

Figure 12. Processing times of using the Snort and ADS, separately, over 23.35 millions of packets processed.

With limited testing period, the ADS detected only 64 anomalous episodes. As shown in Fig.12(b), the total ADS processing time is almost 70 times longer than that of using the Snort. The 9.4 rules/sec rate is two orders of magnitude slower than that of using Snort. The time consumed by ADS is mainly attributed to the complexity of feature extraction, episode classification, episode rule generation, and matching with of the FER database.

Due to sequential processing of IDS followed by ADS, the total CAIDS time is the sum of the two subsystems. The ADS time dominates the total processing time. Therefore, generating new signatures to will reduce the total detection time, because more packet timestamps will be passed on to the ADS. Techniques to eliminate redundant episode rules are given in [31] to reduce the FER space and thus reduce the anomaly detection time.

8. Effects of False Alarms on Detection Rate

False alarms and intrusion detection accuracy are two related concepts. Tradeoffs do exist between these performance measures. Accuracy is tied to intrusion detection success rate. We report below the effects of false alarms on the detection results using the ROC curves. This plots the probability of successful detection against that of false alarms.

8.1 False Alarms under Different Window Sizes

In Fig.13, the false alarms increase with the increasing of scanning window sizes. The Snort contributed very little to false alarms. It is the ADS causes most false alarms. With a window of 300 sec or less, we see 11 false alarms in 10 days. For a large window size of 7200 sec (2 hours), the number of false alarms increases to 17. The U2R has the lowest number of false alarms. The DoS and R2L result in 4 to 6 false alarms and the Probe attacks result in 2 to 4 false alarms. To avoid false alarms, one must use smaller scanning windows. Our recommendation is to limit the window size to be less than 300 sec.

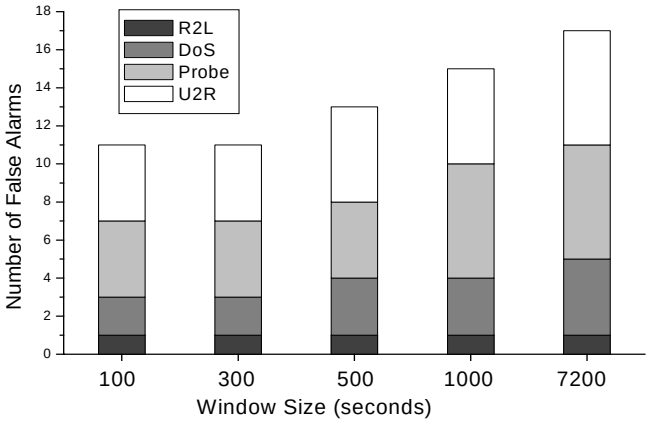


Figure 13. False alarms in 10 days of experimentation for various window sizes.

8.2 Design Tradeoffs from the ROC Curve

In Fig.14, we plot the ROC curves corresponding to 4 attack classes. The detection rate grows quickly to its peak value within a small increase of false alarm rate. The detection rate gets saturated after large false alarms. For example, to achieve a detection rate above

75% of DoS attacks, we have to tolerate 5% or more false alarms. The R2L attacks have the second best performance. The Probe attacks perform about the same as R2L attacks. The U2R attacks have the lowest 25% detection rate at 10 % false alarms due to their stealth nature.

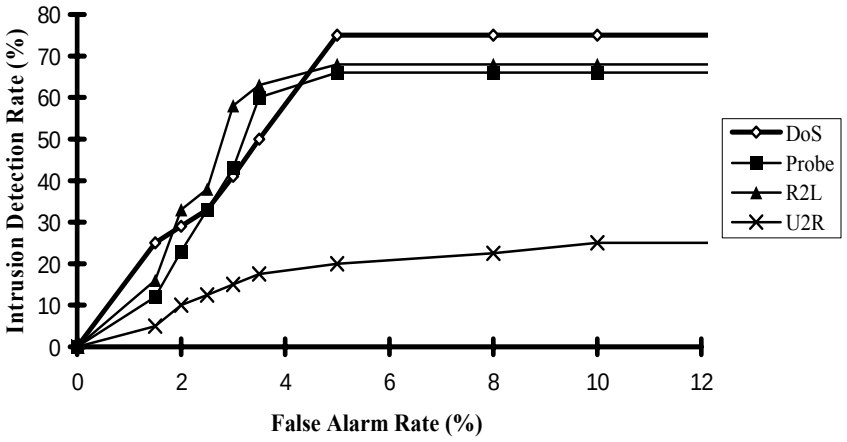


Figure 14. ROC performance of four attack classes in using the CAIDS

Figure 15 shows the average performance of three detection system over all attack types. The CAIDS achieved an average of 64% detection rate at 7% of false alarms. The Snort has almost constant 38% detection rate and the ADS can reach at most 50% performance at 10% false alarms. These results support the claimed advantages of CAIDS. These ROC curves that the balancing point between detection rate and false alarms is possible and it is up to the designer's choice between accuracy tolerance, and flexility.

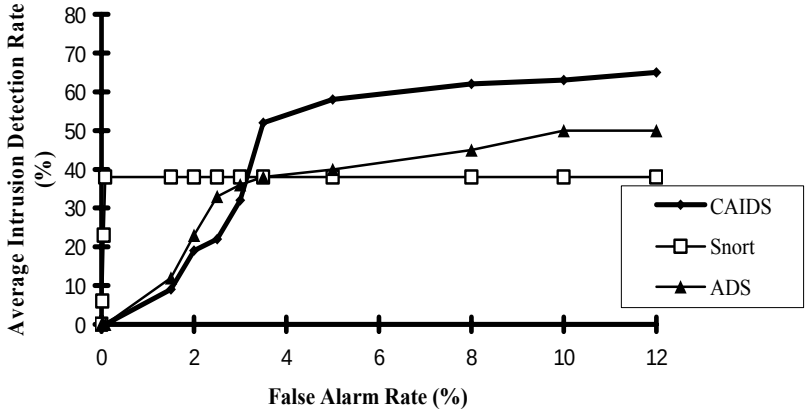


Figure 15. ROC curves showing the variation of the average intrusion detection rate of 3 detection systems as the false alarm rate increases

9. Conclusions and Suggestions

We summarize below the research contributions and make suggestions for further work on automated detection of intrusions and anomalies in open network environments.

9.1 Contributions and Lessons Learned

The CAIDS can be applied to protect any networked or Internet-based computing systems, including LAN-based clusters or Intranets, large-scale computational Grids, information/data Grids, and P2P service networks, etc. The system can be also tested over any network traffic trace data. Summarized below are unique contributions and lessons learned from our CAIDS construction and extensive simulation experiments performed.

(a) We use the *base-support datamining* scheme [32] to facilitate episode rule generation and to improve detection rate. The CAIDS has combined the advantages of Snort and our custom-designed ADS. The advantages of using CAIDS architecture come from being able to adjust the threshold frequency and to change the scanning window size based on the performance demand and false alarms that can be tolerated.

(b) New techniques for connection labeling and alert classification are developed to automate the signature generation process. Our experimental results prove the advantages of cooperative intrusion and anomaly detection over the audit traffic records. The Snort processing time is two orders of magnitude shorter than using the ADS for anomaly detection. Generating more signatures from anomalies enhances the overall CAIDS speed and system efficiency.

(c) Our CAIDS results in a peak detection rate of up to 75%, which is improved from 38% in using the Snort alone and from 50% in using a pure ADS alone. To achieve the high detection performance, the false alarms must be maintained as low as 5% or less. The CAIDS outperforms the Snort and the rule-based ADS in detection accuracy. The

detection rate grows rapidly to reach the saturated level at small increase in false alarms.

(d) Intrusion alerts from signature matching and anomaly alerts from massive and stealth attacks on network connections are correlated with small overhead. Our experimental results supported the claimed advantages. One shortcoming is the limited training data over normal traffic trace. This was the main source of false alarms in our ADS. More benchmark experiments are needed to make further improvement in this area.

9.2 Suggestions for Further Research

For further research, we suggest the following topics for continued effort. Some of these tasks are challenged at USC and some are concurrently pursued by other researchers working on distributed and cooperative intrusion detection and response systems [5, 8, 17, 19, 21, 28, 33, 39].

(e) The proposed new CAIDS was evaluated over a limited Internet trace data mixed with artificially synthesized MIT/LL attack dataset. We need to extend the simulation work to a prototype CAIDS production system. We will use the DETER Testbed [9] in continued effort to analyze Internet episodes without disturbing other legitimate users. Extensive experiments are being planned on the DETER testbed for this purpose.

(f) For each attack class such as DoS, R2L, L2R, and Probe, we need to develop accurate and efficient classifiers to predict the attack type and to help extract the common features for automated signature generation. The RIPPER [7] classifiers need to be extended or optimized for this purpose.

(g) Extending the CAIDS to *distributed edition* is the logical solution in protecting Grids, clusters, intranets, etc. [5, 16, 39]. Cyber trust negotiations and frequent alert information exchanges among distributed IDS sites are the key research issues [5, 12, 28] yet to be addressed satisfactorily. The USC GridSec project [16, 19, 32] attempts to

develop the NetShield software toolkits for this purpose.

(h) Other meaningful extensions of the CAIDS work could be directed towards pushback techniques against DDoS flood attacks [15, 17, 19, 37], and using techniques inspired by human immune systems [2]. Our continued research efforts aim at achieving distributed security enforcement for trusted Grid or P2P computing.

Acknowledgements: The funding support of this work by the NSF ITR Grant ACI-0325409 is appreciated. The critical comments from Ricky Kwok of Hong Kong University are useful to improve the contents. We thank Shanshan Song of USC in enhancing some figure illustrations and making suggestions to improve the readability of the paper.

References:

[1] R. Agrawal, T. Imielinski, and A. Swami, "Mining Associations between Sets of Items in Massive Databases", *Proc. of ACM-SIGMOD 1993 Int'l Conf. on Management of Data*, Wash. D.C., May 1993, pp.207-216.

[2] U. Aickelin, J. Greensmith and J. Twycross, "Immune System Approaches to Intrusion Detection – A Review", *Proc. of Third Int'l Conf. on Artificial Immune Systems,* 2004.

[3] D. Barbara, J. Couto, S. Jajodia, L. Popyack, and N. Wu. "ADAM: Detecting Intrusions by Data Mining". *IEEE Workshop on Information Assurance and Security*, 2001.

[4] S. Bridges and R. M. Vaughn, "Fuzzy Data Mining and Genetic Algorithms Applied to Intrusion Detection", *Proc. of the 23rd National Information Systems Security Conference*, Baltimore, MD, October 2000.

[5] D. J. Burroughs, L. F. Wilson, and G. V. Cybenko, "Analysis of Distributed Intrusion Detection Systems using Bayesian Methods Performance," *IEEE Int'l Computing, and Communications Conference,* 2002. pp. 329 -334.

[6] W. R. Cheswick, S. M. Bellovin, and A.D. Rubin, *Firewalls and Internet Security: Repelling the Wily Hacker,* Addison-Wesley, 2003..

[7] W. Cohen, "Fast Effective Rule Induction", *Proc. of the 12th International Conf. on Machine Learning*, Lake Tahoe, CA, Morgan Kaufman, S.F., CA. 1995.

[8] F. Cuppens and A. Miege, " Alert Correlation in a Cooperative Intrusion Detection Framework", *Proc. of 2002 IEEE Symp. on Security and Privacy* , 2002, pp.187-200.

[9] DETER and EMIST Projects, "Cyber Defense Technology: Networking and Evaluation: *Communications of The ACM,* March 2004, pp. 58 - 61. Also from DETER Website: http://www.isi.edu/deter/docs/acmpaper.pdf

[10] W. Fan, M. Miller, S. Stolfo, W. Lee, and P. Chan. "Using Artificial Anomalies to Detect Unknown and Known Network Intrusions." *Proc. of The First IEEE Int'l Conf. on Data Mining*, San Jose, CA, November 2001.

[11] S. Floyd and V. Paxson, "Difficulties in Simulating The Internet", *IEEE/ACM Trans. on Networking,* Vol.9, No.4, August 2001, pp. 392 – 403.

[12] J. Haines, D.K. Ryder, L. Tinnel and S. Taylor, "Validation of Sensor Alert Correlators," *IEEE Security and Privacy*, January 2003

[13] Helmer, G., Wong, J., Honavar, V., and Miller, L. (1999). "Automated Discovery of Concise Predictive Rules for Intrusion Detection". Tech. Report TR 99-01, Dept. of Computer Science, Iowa State University, Ames, IA., 1999.

[14] J. L. Hennessy, D. A.. Patterson, and H. S. Lin, Editors, *Information Technology for Counterterrorism: Immediate Actions and Future Possibilities,* National Academies Press, 2003.

[15] A. Hussain, J. Heidemann, and C. Papadopoulos, " A Framework for Classifying Denial of service Attacks", *Proc. of ACM SIGCOMM*, 2003.

[16] K. Hwang, H. Liu, and Y. Chen, "Protecting Network-Centric Systems with Joint Anomaly and Intrusion Detection over Internet Episodes", *IEEE IPDPS- 2005*, submitted Oct.8, 2004.

[17] J. Ioannidis and S. M. Bellovin, "Implementing Pushback: Router-Based Defence Against DDoS Attacks", *Proc. Network and Distributed System Security Symp.,*2002.

[18] Kaleton Internet, "Combination of Misuse and Anomaly intrusion Detection systems", Chonburi, Thailand, Web: http://www.kaleton.com, March 2002.

[19] Y.K. Kwok, M. Cai, and K. Hwang,"A Scalable Set-Union Approach to Pushing Back DDoS Attacks", *IEEE Symposium on Security and Privacy,* submitted Nov.5, 2004.

[20] K. S. Killourhy and R. A. Maxion. "Undermining an Anomaly-Based Intrusion Detection System using Common Exploits", *Proc. of Int'l Symp. on Recent Advances in Intrusion Detection* (RAID2002), Sept. 2002, pp. 54-73.

[21] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava, "A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection", *Third SIAM Conf. on Data Mining,* S.F., 2003, also Web site: http://www.users.cs.umn.edu/~kumar/papers

[22] W. Lee, S. J. Stolfo, and K. Mok. "Adaptive Intrusion Detection: a Data Mining Approach", *Artificial Intelligence Review*, Kluwer Academic Publishers, 14(6), Dec. 2000, pp. 533-567.

[23] W. Lee and S. Stolfo, "A Framework for Constructing Features and Models for Intrusion Detection Systems", *ACM Trans. On Information and System Security* (TISSec), 2000

[24] R.P.Lippmann and J.Haines, "Analysis and Results of the 1999 DARPA Off-Line Intrusion Detection Evaluation," *Proc. Recent Advances in Intrusion Detection (3rd Int'l Workshop RAID 2000)*, H. Debar, L. Me, and S.F. Wu, eds., Springer-Verlag, New York, 2000, pp. 162-182.

[25] M. V. Mahoney and P. K. Chan. "An Analysis of the 1999 DARPA/Lincoln Lab Evaluation Data for Network Anomaly Detection". *Proc. of Int'l Symp. on Recent Advances in Intrusion Detection,* Sept. 2003.pp. 220-237.

[26] H. Mannila and H. Toivonen. "Discovering Generalized Episodes using Minimal Occurrences", *Proc. of Second Int'l Conf. on Knowledge Discovery and Datamining,* Portland, August, 1996.

[27] J. McHugh, "Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Off-line Intrusion Detection System Evaluation as Performed by Lincoln Laboratory". *ACM Trans. on Information and System Security, 3(4)*. Nov. 2000

[28] P. Ning, S. Jajodia, and X. S. Wang, "Abstraction-based Intrusion Detection in Distributed Environments," *ACM Trans. on Information and System Security (TISSEC)*, 4(4):407-452, Nov. 2001.

[29] S. Noel, D. Wijesekera, and C. Youman, "Modern Intrusion Detection, Data Mining, and Degrees of Attack Guilt" in *Applications of Data Mining in Computer Security*, Barbarà and Jajodia, eds. Kluwer Publishers, Boston, 2002.

[30] V. Paxson, " Bro: A System for Detecting Network Intrusions in Real Time", *Proc. of the Seventh USENIX Security Symposium,* San Antonio, TX. 1998.

[31] P. A. Porras and P. G. Neumann. "EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances", *Proc. of the 19th National Computer Security Conf.*, Baltimore, MD. Oct. 1997, pp. 353-365.

[32] M. Qin and K. Hwang. "Frequent Episode Rules for Internet Traffic Analysis and Anomaly Detection", *Proc. of IEEE Network Computing and Applications,* (NAC2004), Cambridge, MA. September 2004.

[33] D. J. Ragsdale, C.A. Carver, J. Humphries, and U. Pooch, "Adaptation Techniques for Intrusion Detection and Response Systems", *Proc. of the IEEE Int'l Conf. on Systems, Man, and Cybernetics,* Nashville, TN., Oct. 8-11, 2000, pp. 2344-2349.

[34] M.J. Ranum, " Experiences Benchmarking Intrusion Detection Systems", NFR Security Technical Publication. 2002, http://www.nfr.com/publications/

[35] M. Roesch, "Snort - Lightweight Intrusion Detection for Networks". *Proc. of USENIX 13th Systems Administration Conf.* (LISA '99), Berkeley, CA.,1999, pp. 229 -238.

[36] F. Schneider, S. M. Bellovin, and A. Inouye, "Building Trustworthy Systems", *IEEE Internet Computing,* Nov./Dec. 1999, Vol.3, No.6, pp. 64-72

[37] S.M. Specht and R. B. Lee, "Distributed Denial of Service: Taxonomies of Attacks, Tools, and Countermeasures", *Proc. of International Conf. on Parallel Computing Systems* (PDCS-2004), San Francisco, October 19, 2004.

[38] G. Vigna, M. Eckmann, and R.A. Kemmerer, "The STAT Tool Suite," *Proceedings of DISCEX 2000*, IEEE Press, Hilton Head, SC January 2000, pp. 46-55.

[39] G. B. White, E. A. Fisch, and U. W. Pooch, " Cooperating Security Managers: A Peer-Based Intrusion Detection System". *IEEE Network*, Jan. 1996, pp. 20-23.

[40] Y. Xie, H. Kim, D. R. O'Hallaron, M. K. Reiter and H. Zhang, "Seurat: A Pointillist Approach to Anomaly Detection", *7th Int'l Symp. on Recent Advances in Intrusion Detection* (RAID-2004).

Biographical Sketches:

Kai Hwang is a Professor and Director of Internet and Grid Computing Laboratory at the University of Southern California. He received the Ph.D. in EECS from the University of California, Berkeley. An IEEE Fellow, he specializes in computer architecture, parallel processing, Internet and wireless security, and distributed computing systems. Presently, he leads a USC research group on the GridSec Project. He can be reached at **kaihwang@usc.edu.**

The team develops trust models and game-theoretic strategies for Grid computing under security and selfishness constraints. security-driven heuristics and genetic algorithms for on-line Grid job scheduling, and self-defense toolkits and distributed intrusion detection systems for protecting clusters or Grids from malicious attacks and pushing back the DDoS attacks to the originating ingress routers. Visit the web site: http://GridSec.usc.edu for details of related published works.

Hua Liu received the B.S in Computer Science from Fudan University, China in 2001 and the M.S. in Computer Science from University of Louisiana at Lafayette in 2003. She is presently pursuing the Ph.D. degree in Computer Science at the University of Southern California. Her research interest lies in automated intrusion detection and response systems and Grid security infrastructure. She can be reached at **hual@usc.edu**

Ying Chen received the B.S. degree in Computer Science from Huazhong University of Science and Technology, China in 2001 and the M.E degree from the Oregon Graduate Institute, Portland in 2004. She is presently pursuing Ph.D. degree in Computer Engineering at the University of Southern California. Her research interest includes Internet security and distributed computing systems. She can be reached at **chen2@usc.edu**