Defending Distributed Systems Against Malicious Intrusions and Network Anomalies*

Kai Hwang, Ying Chen, Hua Liu
University of Southern California, Los Angeles, CA 90089
{kaihwang, chen2, hual}@usc.edu

Keynote Presentation by Kai Hwang
at the IEEE International Workshop on Security in Systems and Networks (SSN-2005), held in conjunction with the IEEE International Parallel and Distributed Processing Symposium (IPDPS-2005), Denver, Colorado, April 8, 2005

Abstract

Network security breaches hinder the application of distributed computing systems manifested as the Grids, clusters, intranets, extranets, or P2P systems. A new integrated approach is presented for building future, network-based intrusion detection systems (NIDS). We integrate the Snort (a NIDS) with a custom-designed anomaly detection system (ADS) to yield a powerful cyber defense system, called CAIDS. This system detects known attacks through signature matching and reveals network anomalies by Internet traffic datamining.

The CAIDS design integrates two different detection engines for alert correlation between intrusions and anomalies. We aim to automate signature generation into Snort database. The system was tested over an Internet trace of 24 millions of packets containing 200 attacks. Our simulation experiments result in a 75% detection rate on all attacks with a low 5% false alarm rate. The system generates alerts on both intrusive attacks to distributed resources and anomalies detected in the Internet, intranet, and extranet connections.

1. Introduction

Network-centric computing systems manifest as Grids, Intranets, clusters, P2P systems, etc. [1, 4]. Malicious intrusions to these systems may destroy valuable hosts, network, and storage resources. Network anomalies may appear in many Internet connections for telnet, http, ftp, smtp, Email, and authentication services. These anomalies cause even more damages. Internet anomalies found in routers, gateways, and distributed hosts may hinder the acceptance of Grids, clusters, and public-resource networks [3].

The misuse IDS model used in Snort is based on matching of attack signature with pre-stored signatures associated with known attacks like the PoD, portsweep, Dosnuke, Teardrop, and Saint, etc. The Anomaly-based IDS (called ADS) compares the incoming traffic records with the normal traffic profiles from stored historical audit records [6, 13]. Attacks like Dict (a dictionary attack), Neptune (a SYN flood attack), UDPStorm, Apache2, Guesstelnet, Smurf, etc can be detected by an ADS, but not by the Snort.

In the past, most IDS detect only single-connection attacks at the packet level. They cannot detect unknown attacks or encrypted packets. Unknown or bursting attacks involving multiple connections can only be detected by an ADS. Association rules and datamining techniques [7, 8, 10, 11] were suggested to implement ADS. Lazarevic and Kumar et al [7] have assessed the differences between single-connection and multi-connection attacks.

Lee and Stolfo [8] have developed a level-wise datamining algorithm for building ADS. Qin and Hwang [13] have developed a base-support algorithm for anomaly detection. As pointed out by Kumar and associates [7] that both IDS and ADS are sensitive to the attack characteristics, system training history, and the underlying network conditions. In an ADS, the incoming traffic is compared with the huge database of normal traffic profiles to reveal patterns that are significantly deviating from the norms of the Internet traffic episodes.

Keleton Internet [5] has built a simple prototype system by using an IDS and ADS, independently. In our approach, we integrate the IDS and ADS interactively. The purpose is to protect network-centric systems from intrusions or anomalies. The attacks may come with viruses, worms, backdoor, or malicious programs. We present the new integration methodology, CAIDS system architecture, enabling mechanisms, and some simulation results with performance analysis.

* This research work was fully supported by an NSF ITR Grant ACI-0325409. The corresponding author is Kai Hwang. Contact him at Email: kaihwang@usc.edu or Tel: 213 740 4470.
2. CAIDS: A New Integrated Approach

Our proposed new approach is briefly introduced below. Enabling mechanisms and various building blocks will be detailed in subsequent sections. Our work on developing a Cooperative Anomaly and Intrusion Detection System (CAIDS) is motivated to enhance cybersecurity with complementary techniques [3].

We want to cope with both known and unknown attacks at the same time. The integrated approach is depicted in Fig. 1 for building future cyber defense systems. Our experimental findings clearly reveal the advantages of combining the NIDS and ADS. These two subsystems complement each other to yield much improved detection results.

We integrate the NIDS and ADS by making them supporting each other. The signature-matching NIDS are cascaded with a custom designed ADS. These two subsystems join hand to cover all traffic flow events, initiated by both legitimate and malicious users. Single-connection intrusive attacks are primarily detected by NIDS at the packet level by signature matching with the attack database.

Unknown, burst, or multi-connection traffic, which cannot be detected by signature-based NIDS, are passed on to the ADS. A novel signature generator is proposed to bridge the two subsystems. Details of generating new signatures from anomalous connections to add into the Snort database are given in Section 5.

We installed the NIDS Snort and build an ADS at the USC Internet and Grid Computing Laboratory to process the incoming traffic sequentially. In total, there are 23.35 M packets processed by the Snort. Most traffic involves the TCP, UDP, and ICMP packets. The Snort recorded 24,619 alerts or log files. The total Snort time was recorded 282.73 sec. The ADS has total processing time of 333 minutes. We will use these raw data to plot and analyze the performance results of three different intrusion detection systems in subsequent sections.

The main purpose of mining Internet traffic data is to reveal interesting relationship among Internet connection patterns. An Internet episode is represented by a sequence of TCP connections. We consider all connections initiated by the same attack. This connection definition may not need handshake to establish a session.

A connection episode can be generated by legitimate users or malicious attackers. The frequent episode appears very often, most likely initiated by legitimate users. A rare episode is likely caused by intruders or attackers. Our purpose is to distinguish rare or abnormal episodes from the normal or frequent episodes through a rule matching process.

A typical stream of Internet traffic is represented by a sequence of connection events. The time instants of these connections in second are marked below the events. A frequent episode is a set of episodes exceeding certain 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 with multiple connection events.

For an example, we envison a window where we observe a 3-event sequence: E, D, and F. An FER is generated as: E → D, F with a confidence level \( \frac{\text{freq}(\gamma \cup \beta)}{\text{freq}(\beta)} = 0.8 \), where \( \gamma \) represents the event E on the LHS and \( \beta \) corresponds to the two events D and F on the RHS of the rule.

Each FER must have a confidence larger than a certain minimum confidence threshold. If the \( \beta \) occurs with 5% and the joint event \( \gamma \) and \( \beta \) has 4% to occur, there is a \((0.04/0.05) = 80\%\) chance that D and F will follow in the same window.

In practice, the event E could be an authentication service characterized by two attributes \( \text{(service = authentication, flag=SF)} \). The events D, F may be two sequential smtp requests denoted by \( \text{(service = smtp)} \) (service = smtp). Thus we can derive an FER with a confidence level of \( c = 80\% \), that two smtp services will follow the authentication service within a window \( w = 2 \) sec. The three joint traffic events accounts with a support level \( s = 10\% \) out of all the network connections being evaluated. This FER is formally stated as follows:

\[
\text{(service = authentication)} \rightarrow \text{(service = smtp)} \\
\text{(service = smtp)} (0.8, 0.1, 2 \text{ sec})
\] (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. In general, an FER is specified by the following expression:

\[ L_1, L_2, \ldots, L_n \rightarrow R_1, \ldots, R_m (c, s, \text{window}) \]  

(2)

where \( L_i \) (1 \( \leq i \leq n \)) and \( R_j \) (1 \( \leq j \leq m \)) are ordered traffic connection events. We call \( L_1, L_2, \ldots, L_n \) the LHS episode and \( R_1, \ldots, R_m \) the RHS of the episode rule.

We consider the minimal occurrence of the episode defined by Mannila and Toivonen [10] over the entire traffic stream. The support value \( s \) 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 \( c \) is the joint probability of the minimal occurrence of the joint episodes out of the support for the LHS episode.

It is desired to have shorter LHS, higher confidence level, and longer RHS in FERs. Several rule pruning techniques were developed in [13]. These rule pruning techniques can be applied to remove redundancy and ineffective episode rules. This will significantly reduce the rule search space and thus results in faster search through the normal FER database and achieves higher efficiency in the Internet datamining of traffic episodes.

3. CAIDS Simulator and Attack Datasets

Figure 2 shows the functional blocks in the CAIDS simulator design at USC. This system was simulated on a Dell Linux server with dual 2.6G Hz Xeon processors running the Redhat Linux 9.0 Kernel with 2 GB of main memory. The installed Snort 2.1 is a lightweight NIDS with over 2,000 signatures in its database. It only works with those pre-stored signatures [15].

An FER-based ADS was built locally to detect unknown anomalies. A datamining engine was built in the simulator to generate the FER database and the anomaly log files. The signature generator receives the anomaly rules and their log files to update the signature database. The size of the FER database is decided by the training data processed.

The Internet trace is obtained from an ISP through the Los Nettos network in Southern California. It contains of 4 GB real traffic data to provide the background traffic. We use the SAD toolkits by Mahoney and Chan [9] to mix the USC Internet trace with the MIT/LL attack dataset.

The NetAttack adds real-life background traffic to the MIT/LL attack data. There are in total 201 attacks in the MIT/LL data set, consisting of 58 DoS (Denial-of-Service) attacks, 62 R2L (Root to Local) attacks, 31 U2R (User to Root) attacks, 44 PROBE or port scanning

![Diagram](image-url)
attacks, and 6 secret or unknown attacks. Some of the attacks repeat many times at different days.

![Attack spectrum in 10 days of Internet trace mixed with 200 attacks from MIT/LL dataset](image)

**Figure 3.** Attack spectrum in 10 days of Internet trace mixed with 200 attacks from MIT/LL dataset

### 4. 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.

We apply the normal profile database and construct the anomaly detection engine. The detection engine is capable of detecting anomalous episodes that is caused by attacks or unauthorized accesses. To generate the FERs for normal traffic profiles, the attack-free training connection records are fed into the datamining engine. The anomaly detection process in using FERs is specified in Fig.4.

We use the BRO toolkit [12] to extract useful features from traffic connection records. When an intrusion is detected at the packet level, the Snort sends its *timestamp* to packet eliminator. A major task is to determine the minimum occurrence number of traffic episodes. We generated 60 FERs with limited training time. We do not use FERs with extremely low support values.

After finding FERs from each day’s audit record, we merge them into a large rule set by removing redundant rules. We kept a keen interest on rare attributes of both single and multiple connections. Four levels of decision boxes in Fig.4 distinguish three possible sources of attacks, *packet-level alerts* from the Snort, *stealthy attacks*, and *massive attacks*.

The ADS generated 187,059 traffic episode rules in 333 minutes, which was spread across the 10 days of experimentation. Interesting connection features checked include the *source or destination addresses, port numbers, time durations, bytes sent, special error flags*, etc. 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 sequence numbers to check connections with the same destination, instead of using the timestamps.

![Feature Extraction](image)

**Figure 4.** Matching episode rules to detect anomalies using the base-support Internet datamining algorithm by Qin and Hwang [13]

After many anomaly rules are generated in a scanning window, we calculate the temporal statistics from the network connections. Temporal statistics include the *average network traffic, number of error flags, number of connections* to the same destination, etc. More background of this algorithm can be found in our earlier work [10]. A SYN flood attack is distinguished from normal traffic by looking at the number of SYN packets destined for the same IP from unknown sources. The SYN flood attack is specified by the following episode rule.

\[(\text{service} = \text{http}, \text{flag} = \text{s0}) \rightarrow (\text{service} = \text{http}, \text{flag} = \text{s0}) \]

\[(\text{service} = \text{http}, \text{flag} = \text{s0}) (0.1, 0.8, 100 \text{ sec}) \] (3)

Here, the event \((\text{service} = \text{http}, \text{flag} = \text{s0})\) has a flag=s0 implying only SYN packets were seen in the TCP connection. This FER corresponds to the case of an attacker keeping sending \text{http} requests to the same destination. All \text{http} connections are SYN packets and no FIN packets were found. Thus the router buffer gets flooded and all future services are denied.

### 5. Semi-Automated Signature Generation

Automated signature generation is very much desired in SNORT or any other misuse-based IDS. It is a major challenge to synthesize attack signatures from anomaly rules. We attempted to extract useful attributes from the anomaly FERs to generate new signatures to add into the Snort database. Our signature generation method is guided...
by predefined alert RIPPER classifiers [1]. This offers only a semi-automated solution.

The signature generator in Fig.3 is designed to do so. This signature insertion operation is similar to the way Snort agency updates signatures to user groups. In what follows, we introduce the connection labeling and alert classification mechanisms to automate the signature generation process. This method may introduce some false alarms due to added noise with machine learning [1, 11].

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. To generate Snort signature for the anomaly FERs detected, we have to extract at least the source address, destination address, and the service 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 [13]. Packet filtering discovers some anomalies in single connections. Generating signature at packet level is similar to the original Snort signature formation.

Two anomaly paths in Fig.4 correspond to multiple-connection episodes, which were called burst attacks by Kumar, et al [7]. Burst attacks can be divided into two subclasses called stealthy and massive attacks. When the traffic FER does not match with any of the stored FERs for normal traffic, we encounter the stealthy attacks. Massive attacks take place when the matched FER exceeds the minimum threshold on connection count.

We want to generate the signatures for these two types of detected anomalies, separately, using different signature extraction mechanisms: the statistic method [5] was used for stealthy attacks and the classification method [8] used for handling massive attacks. We use the destination IP address as a reference attribute.

Labels are used to bind traffic connections to specific FER. Stealthy attacks often involve only smaller number of connections. Massive attacks trigger larger volumes of connections. Only the anomalies exceeding certain threshold can be assigned with unique signatures. It is possible that only a few signatures are generated for a large number of similar anomalies.

Burst attacks include both stealthy attacks with small traffic flows and massive attacks 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, such as the common destination IP and port number, common source IP and port, protocol applied, connection duration, bytes sent or received, etc. out of 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 as accurately as possible.

Massive attacks often cause a large amount of traffic flows triggering the anomalous connections. We suggest using the RIPPER classifiers [1] to predict attack type and help extract interesting features to generate high-level signatures for a pre-defined class of attacks. Classifiers are the output of the process of classification.

Alert classification is a common datamining practice to assign traffic records to one of a set of predefined target classes. For example, the attack could be classified any of 4 classes (DoS, R2L, L2R, and Probe). Different attack classes use different classifiers to guide the feature selection and attack classification.

Here, we concentrate on how to use classifiers to help generate signatures. Tracing the flow in Fig.4, we face the situation of massive attacks triggered by the matched episode exceeding the threshold frequency. First, according to matched FER rule, we have to check different intrinsic features and content features, such as the flags, connection duration, failed login attempts, root-shell access, etc. Then, we use the classifiers to predict the most suspicious attacks.

The purpose to generate new signatures for the Snort is to use in the next round of attacks of the same type. 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 attacks, the flag and temporal information are used to distinguish different attacks under the same attack type. For example, using the connection period and SYN or FIN flags to distinguish the Dosnuke attack from the Pod attack, both fall under the DoS attacks. Other temporal statistics often applied include the connection count, to the same destination, the percentage of connections to the same port, etc.

The episode rule specified in Eq.(3) is associated with an SYN Flood attack. We explain next how to generate the corresponding Snort signature. To simulate a SYN Flood attack, we set the threshold count = 20. Consider the case of 24 SYN packets heading for the same http port during the scanning period. A typical RIPPER classifier rule is selected below:

\[
\text{SYN flood : - service=http, flag=s0, count>20} \quad (4)
\]

The selected classifier helps predict the attack class and extract the common features from all relevant connections to specify a Snort signature. In this case, there
are 24 relevant connections exceeding the FER threshold of 20. A Snort rule is thus generated as follows:

\[ \text{Alert tcp } !\text{HOME_NET} \text{ any } \rightarrow \text{HOME_NET 80} \]
\[ \text{(msg: "SYN Flood Alert Generated by ADS"; flags: S; threshold: type both, track by_dst, count 20, seconds 120;)} \]

When the Snort sees more than 20 SYN packets sent to the same port 80 on the same host within the same window of 120 sec, it is detected as an SYN flood attack and an alert is raised. Each distinct massive attack matches a separate RIPPER classifier. In the Snort rule: \text{HOME_NET} represents all IP addresses inside an intranet. In the signature sequence, the threshold: type both, track by_dst, count 20, seconds 120 refers to the alert once signal.

6. Detection Rate and False Alarms

The detection performances of the IDS (Snort), the ADS, and the CAIDS are reported below based on simulation and Internet trace experiments at USC Internet and Grid Computing Laboratory during 2004. The Snort detects well-known intrusive attacks, while the ADS detects anomalous Internet connections. The CAIDS intends to cope with both types of attacks.

Relative strengths and weaknesses of these three systems are assessed based on numerical results obtained. 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. We report the enhanced detection rate with small false alarms.

Three performance measures are used to present the detection results of an IDS. The detection rate is the successfully detected attacks over all possible attacks within an observation period. The false alarm rate measures the percentage of false-positives among all traffics. The third measure is the ROC curve, which plots the variation of the detection rate with respect to variation in false alarm rate. We plot in Fig.5 the intrusion detection rates of 3 detection systems: Snort, ADS, and CAIDS with respect to 4 attack classes in 10 days of experimentation.

Each bar in Fig.5 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 performances of the Snort and ADS are not stable. These two systems may fail down to a zero detection rate on certain days.

In Fig.6, we plot the average detection rates over 10 days. The rightmost bars correspond to the average detection rates across all 4 attack types. We observe the higher minimum detection rate of the CAIDS in all attack categories except the U2R attacks. We see distinct advantages of the CAIDS in detecting DoS and PROBE attacks. The highest detection rate of CAIDS is 75% and the minimum is 30%. The detection accuracy of ADS fluctuates due to some false alarms. The ADS performs lower than Snort in detecting the PROBE attacks. All three systems perform poorly on the U2R attacks.

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 ROC curves represent the probability of successful detection against the probability of false alarms. The ROC curve reveals the tradeoffs between successful detection against false alarms.

In Fig.7, we plot the ROC curves for 4 attack classes. The detection rate gets saturated quickly after experiencing some 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 of 68% detection rate at 5% false alarms. The Probe attacks have 66% detection rate at 5% false alarms. The U2R attacks have the lowest detection rate of 25% at 10% false alarms.

![Figure 7](image)

**Figure 7.** ROC curves showing the variation of intrusion detection rates of 4 attack classes

Figure 8 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 an average 38% detection rate and the ADS can reach at most 50% performance at 10% false alarms. All of these results support the claimed advantages of the CAIDS over the use of the Snort or ADS alone.

![Figure 8](image)

**Figure 8.** Variation of intrusion detection rates of 3 detection systems with the false alarm rate

The Snort has a flat low detection rate of 38% with any rate of false alarms. The ADS is 28% lower than the CAIDS in detection performance, as the false alarms increases beyond 10%. The balancing point between detection rate and false alarms is up to the designer’s choice. The decision depends on particular performance metric chosen in different network applications.

### 7. Conclusions and Suggestions

This work demands a major change of design philosophy from the traditional misuse model or anomaly model for NIDS. Extended R/D efforts are needed to perfect the new idea of integrating ADS with NIDS to build real-time, stateful system that can adapt to changes in network conditions [12, 14]. Summarized below are unique contributions of this work and suggestions for further research challenges.

(a) Our new integrated system is based on the base-support algorithm for Internet traffic datamining [13]. The CAIDS has combined the advantages of both Snort and ADS. The two subsystems complement each other nicely to achieve enhanced performance. The advantages of CAIDS come from (i) mining threshold adjustment, (ii) variable window size, and (iii) semi-automatic generation of attack signatures. Fully automated signature generation is still a wide open research problem yet to be solved [1, 11].

(b) Connection labeling and alert classification help automate the signature generation from anomalies detected. This is crucial to enable the integrated detection of intrusions and anomalies over the same audit traffic records. The Snort speed is two orders of magnitude faster than using that of using ADS for anomaly detection. Shifting the detection workload to Snort by generating more signatures from anomalies will enhance the overall speed and efficiency. Tradeoffs between detection rate and false alarms should be considered as a design choice.

(c) Testing the CAIDS scheme over the NetAttack suite at USC, we result in a peak detection rate of up to 75%, which is improved from 38% in using the Snort and 50% in using a pure ADS alone. To achieve the high detection performance, the false alarms must be maintained low such as 5% or less. The CAIDS outperforms Snort and the rule-based ADS in detection accuracy with reduced false alarms. The detection rate grows rapidly to reach the saturated level at small increase in false alarms.

(d) We need to extend the simulation work by building the prototype production system. We will use the DETER Testbed [2] in continued effort to analyze Internet episodes without hurting legitimate users. Extensive experiments are being planned on the DETER testbed against DDoS or other attacks. [6, 16] Extending the CAIDS to distributed IDS is the logical solution for protecting Grids, clusters, intranets, etc. Security policy negotiations and frequent alert correlation among multiple ISP domains are the key research issues ahead [3, 4].
Acknowledgements: The work reported herein was supported by NSF ITR grant No. ACI-0325409. Discussions with the GridSec research group at the University of Southern California are useful in improving the contents of this paper.

References


Biographical Sketches:

Kai Hwang is a Professor and Director of the Internet and Grid Computing Laboratory at the University of Southern California (USC). He received the Ph.D. from University of California, Berkeley. An IEEE Fellow, he specializes in computer architecture, parallel processing, Internet and wireless security, and distributed computing systems. He is the founding Editor-in-Chief of the Journal of Parallel and Distributed Computing.

Presently, Dr. Hwang leads a USC research group developing fuzzy-theoretic and reputation-based trust models, automated intrusion detection and response systems, and distributed security infrastructure for network-centric Grid and P2P computing. Visit the Project web site: http://GridSec.usc.edu for details. Hwang can be reached at kaihwang@usc.edu.

Ying Chen received her 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 Grid computing systems. She can be reached at chen2@usc.edu

Hua Liu received her B.S in Computer Science from Fudan University, China in 2001 and her 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 quality of service and wireless security areas. She can be reached at hual@usc.edu