Multi-Tier Intrusion Detection System

Jun Li
University of California, Los Angeles
Lijun@cs.ucla.edu

Gerhard Eschelbeck
University of Linz, Austria
Gerhard@eschelbeck.com

ABSTRACT

There have been many different intrusion detection approaches, but none has been accurate enough to avoid false negatives and false positives. In this paper we propose a novel approach to more precisely deter mine whether or not an intrusion has happened. In this approach different intrusion detection techniques are smoothly integrated. This integrated system consists of a number of detectors, each sitting on a specific tier and employing a different intrusio n detection method.

A detector has three main components: a consumer, a filter and a producer. The consumer collects signaling events from detectors at lower tiers or primary event sources and forwards them to the filter. The filter amplifies the intru sion signal and reduces the noise, and exports the resulting signaling events to the producer. From the producer, these signaling events are further exported to detectors at higher tiers, or directly to system administrators. This multi -tiered structure makes it possible to refine chaotic information from primitive sources from the bottom up, until system administrators on top finally receive a clear signal (if an intrusion does happen).

This system is extensible—a new detector, thus a new intrusion detec tion technique, can easily be incorporated by following interface specification for detector interaction. It is also configurable—each detector can configure its own policy by following internal detector interface. Finally, as long as communication authe nticity and secrecy between detectors can be guaranteed, this system can be readily distributed over multiple machines and geographic areas, further facilitating to detect intrusions that cross a wider area.

Key Words: intrusion detection, false negative, false positive

1 INTRODUCTION

1.1 Brief introduction of intrusion detection

Computer intrusion aims to compromise the integrity, confidentiality or resource availability of a computer system. Motivated by the vast amount of political or financial benefits that successful attacks could bring, the number and the complexity of intrusions has been on the rise. These intrusions can penetrate even more quickly and easily by utilizing state-of-the-art communication techniques.

A computer is usually subject to vulner abilities of applications, system software and its communication methodology. Although a well-designed system can reduce the chance of being attacked, designing an absolutely secure system is simply not feasible. For example, a firewall at the boundary of a domain cannot completely "purify" the traffic; many attacks happen inside a domain [12]. To promptly respond to an intrusion, an effective intrusion detection system must be designed.

Intrusion detection systems, called ***IDS***, fall into one of two categories: anomaly detection and misuse detection [11]. Anomaly detection, which assumes that all intrusions are anomalous, determines an action

to be an intrusion by comparing it with the normal pro file of an application, a user or a system. Misuse detection, in contrast, assumes that an intrusion can be characterized by some specific pattern or signatures.

Misuse detection uses such techniques as expert systems, keystroke monitoring, model based intrusion detection, state transition analysis, pattern matching, to establish patterns or signatures that identify an attack [11]. Traffic monitoring has also been used. In contrast, anomaly detection uses statistical approac hes, predictive pattern generation and neural networks [11]. One system that employs anomaly detection is Dorothy Denning's Generic Intrusion Detection Model, which maintains a set of profiles for subjects and matches each audi t record against this profile to determine whether to update it or signal an abnormal behavior as an intrusion [4].

1.2 Challenges

There are several challenges in designing an effective and efficient intrusion detection system. Typical topics include, but not limited to, how to avoid false negatives and false positives, how to cooperate with other intrusion detection systems, how to make the system easily configurable, and how to make it secure as well as perform efficiently. We discuss these issues in the following.

1.2.1 Avoiding false negatives and false positives

Each individual approach is subject to false negatives (an intrusion not detected) and false positives (an allowed action determined as an intrusion). In a system using ano maly detection, an intrusion action may not be regarded as anomalous, resulting in a false negative; an allowed action may be treated as an anomaly, and thus treated as an intrusion, leading to a false positive. In a system using misuse detection, an unkn own attack will not be detected, also leading to a false negative. To effectively avoid false determination, we believe that every aspect of the system must be monitored and the results should be aggregated to make a decision.

1.2.2 Cooperating with other intrusion detection systems

An IDS should cooperate with its peer systems to be most effective. As exemplified in distributed denial of service attacks [19], when an attacker controls many slave machines, a victim can be simultaneously attacked from many locations. Merely being able to monitor one machine does not help address the whole attack. Ideally, all machines should be monitored effectively. This requires an IDS to gain up-to-date and comprehensive information related to a ll possible intrusions. Cooperation among different IDSs help exchange information, such as critical intrusion events or new intrusion signatures, and even take common actions.

Given the desirability of cooperation among various intrusion detection system s, more thought must be given to the design of the cooperation mechanism. First, the cooperation itself must be secure. A peer IDS must be authenticated before its information can be accepted and utilized. Second, a good mechanism is needed to facilitat e inter-communication between systems. Third, cooperation must foster better decision making.

1.2.3 Making it easily configurable and extensible

Intrusion detection has been evolving over time. While the number and complexities of intrusions are changing all the time, the detection methods also tend to improve. To accommodate a large variety of different detection methods, an effective intrusion detection system must be easily configurable and extensible to add new intrusion signatures, employ new detection methods, or dismiss obsolete ones.

1.3 Designing a multi-tier integrated IDS

We propose in this paper to build an integrated intrusion detection system that contains a number of detectors organized in a multi -tier structure. In this structure, the primary info rmation sources such as network traffic or log information is at the bottom tier, and system administrators are at the top tier. A detector has three main components: a consumer, a filter and a producer, where the consumer accepts signaling events from ot her detectors or primary sources, the producer exports signal events to other detectors or system administrators, and the filter amplifies intrusion signals and reduces noises. As a result, the information from the primary source will be enlarged to signi ficant signals that, if intrusion does happen, are communicated to the system administrators at the top. The system administrators thus can determine whether or not the signals are caused by an intrusion.

1.4 Overview of the paper

This paper is organized as f ollowing. In Section 2 we discuss the architecture of this multi -tier intrusion detection system. Section 3 explores the dynamic formation of the architecture. Advanced issues are outlined in Section 4, where policy enforcem ent, detection efficiency and detector authenticity will be addressed. Section 5 shows an example. Related works will be reviewed in section 6. Section 7 is about future work and we conclude in Section 8.

2 ARCHITECTURE

2.1 Description

The intrusion detectio n structure we propose is a multi -tier coordinated system that consists of a number of detectors [Figure 1]. Each detector resides at a specific tier and connects with other detectors below and above. Together, all detectors form a multi -tier structure t hrough which they cooperate to monitor a system and detect intrusions. Although each individual detector that focuses on some specific aspects could have high false negatives or high false positives, a system that seamlessly combines all those detectors can decide more accurately whether an intrusion has occurred or not.

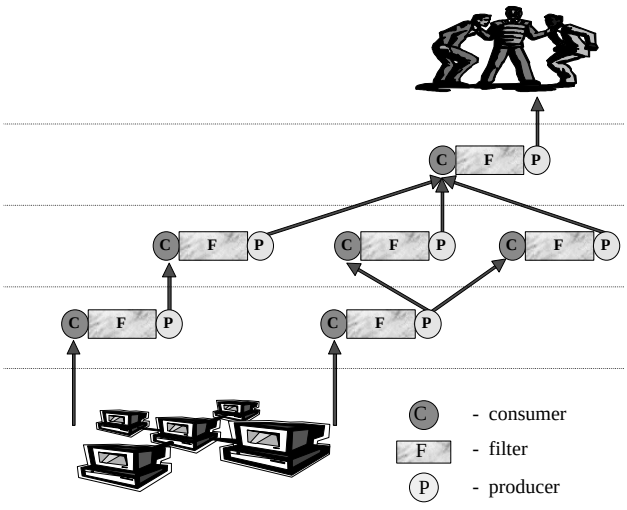

Figure 1: Multi-tier intrusion detection

Each individual detector has a consumer, a filter and a producer. The consumer is connected with the producers of other detectors at lower tiers, whereas the producer connects to the consumers of detectors at higher tiers. In particular, a producer can connect to multiple consumers at upper tiers, and a consumer can consume information from multiple producers at lower tiers.

Detectors at the lowest tier and the highest tier are treated differently. The lowest tier actually contains the primary sources, such as network traffic in the form of a bit stream, system log information, or other low-level information available through calling the operating system services. At the highest tier are system administrators—they are presented with more readable and accurate information, and thus able to make best decisions on whether an intrusion has happened or not in the final analysis.

2.2 Information Flow

2.2.1 Refine information from the bottom up

The raw data output from the lowest tier is actually the input to the whole integrated intrusion detection system. Chaotic, it usually hides useful intrusion detection information. The whole system's goal is to uncover any intrusion-relevant information from such chaotic data. To do so, the data are forced to flow strictly from lower-tier detectors to higher-tier directors and finally reach the system administrators, whereas each involved detector is allowed to transform the data for easier decision making. As a result, the system administrators will be notified with clear intrusion signals in case an intrusion did happen. The type of action that is deemed an intrusion can be regarded differently by different system administrators depending on their context.

In the above data transformation procedure, each detector amplifies intrusion-relevant signals and reduces intrusion-irrelevant noises. Consequently, the input to a detector is transformed to a different format and exported to the following detectors for further processing, perhaps even generating new intrusion signals meanwhile.

2.2.2 Restrictions on information flow

In the multi-tier structure, which disallows information flowing from a higher tier to a lower tier, there can be several different options for information flow as follow:

i. Information flows from a detector in tier i to another detector in tier $i+1$;
ii. Information flows from a detector in tier i to another detector in tier i;
iii. Information flows from a detector in tier i to another detector in tier $i+\boldsymbol{n}$, $\boldsymbol{n}>1$;

If only option *i* is allowed, strict information flow is enforced; option *ii* adds some flexibility; the maximum flexibility is achieved if all three options are allowed. In general, the maximum flexibility in information flow is preferred (although it entails complexity).

2.3 Detector

Each detector specializes in the types of incoming events it processes. For instance, a *networking* detector monitors only primitive network traffic, transmitting network-application-specific information to other detectors that specialize in corresponding network applications. In order to detect an *ftp* relevant intrusion, the networking detector can connect to an *ftp* detector that inspects ftp-specific packets; thus, all ftp-related events can be captured by the *networking* detector and forwarded to the *ftp* detector. The *ftp* detector can further connect to a *statistics* detector that statistically analyzes networking application behavior, and reports any anomalous behavior to system administrators.

2.3.1 Consumer

A consumer is an inter facing component in a detector that accepts incoming information from other detectors or primary sources. It has to maintain one queue or multiple queues to buffer the incoming information from various lower-tier detectors. Here, the consumer can run dif ferent queuing algorithms. FIFO (First-in-first-out), for example, requires the first incoming event also to be the first event processed. In case of multiple queues, the incoming information is put into different queues based on category, priority, or previous detector, and then forwarded to the filter in round-robin or another fashion.

Besides incoming information queuing, another related issue is the limitation of a consumer's buffer space. Since a filter may not be able to process information from a consumer promptly while new input continues to arrive, the consumer's buffer will overflow when the amount of the input is larger than the buffer can accommodate. One solution to buffer overflow is for the consumer simply to tell the sending producer to postpone sending new information and request a retransmission when buffer space becomes available again. However, this solution means that the buffer may be utilized unequally or unfairly. For instance, low-priority but high-frequency events may preempt high-priority but low-frequency events if buffer space is tight.

Another approach to managing the buffer space is via token passing. Knowing which detectors provide incoming information, a consumer can generate tokens based on its available buffer space, and allocate tokens to those source detectors. As a result, a detector, in fact its producer, only exports information to that consumer if it holds a token from that consumer. Tokens are allocated by considering factors such as size, category, priority or information source. On the other hand, a token is reclaimed when the associated event information is received.

As shown in the above analysis, the information forwarding from a producer to a consumer has many similarities to packet routing done by routers.

2.3.2 Filter

A filter is central to a detector since it is an engine that processes all incoming information. It can be either stateless or stateful. A stateless filter processes only incoming information and passes along the results to its producer. In contrast, a stateful filter needs to buffer the information and record the context of a system while waiting for more input.

Generally, a filter can implement any conventional intrusion technique. For example, the Generic Intrusion Detection Model [4] can act simply as a filter in the whole intrusion detection system: it can collect audit records, define the normal profile based on these audit records, and compare each particular action to the normal profile. Information exported from this filter thus only concerns any anomalous behavior that deviates from that expected, which can be further utilized by other detectors.

In addition, a filter can be application-specific. As in our previous example, the *ftp* detector can have a filter that processes only ftp packets. For instance, if an ftp client tries to retrieve the `/etc/passwd` file from a UNIX machine, an alarm signal can be produced.

2.3.3 Producer

The role of a producer is very simple —to forward the results from a filter to the consumers of some upper-tier detectors. For each producer, a local event-forwarding table (**EFT**) is defined that describes (1) what information to forward; (2) where to forward it; and (3) how to forward it.

One issue here is the connection characteristics from a producer to a consumer. Considering a lossy connection that crosses several machines, a reliability mechanism such as TCP may be needed. However,

a UDP-style connection may also be eligible if a small amount of data loss is acceptable; for instance, when data sample are to be transmitted to a detector for statistical analysis.

In general, a producer needs to negotiate on all relevant communication parameters with those consumers to which it is going to send information. Sometimes a consumer may have already suggested a means of reciprocal communication with the producer. In addition, the token mechanism discussed above also necessitates a producer to correctly handle tokens.

2.3.4 Internal detector interface (**IDI**)

Inside a detector, there are mainly two choices for different components to interact with each other. One is through shared memory. The other is message passing with specific message types defined. Although it seems natural to employ shared memory with all components belonging to the same detector, message passing has more advantages. With shared memory, changes in one component can easily influence the design of other components that access some shared memory. Message passing, on the contrary, is more modular. As long as each component follows the message type specification, the interaction between components will still be interoperable even if some changes occurred to a particular component. This message type specification, used for internal interaction between components, is called internal detector interface (**IDI**).

3 DYNAMIC INTEGRATION OF DETECTORS

3.1 Overview

The multi-tier structure can dynamically configure detectors, allowing a new detector loaded or an existing unloaded. In this section we mainly examine how a new detector joins the structure. Since we can assume that originally there are only primary sources and system administrators, investigating the joining procedure can demonstrate the formation of the whole multi-tier structure. (We omit the discussion on unloading a detector.)

A new detector must locate itself at a specific tier through the joining procedure. Most importantly, it must smoothly interact with those existing detectors, in order to import necessary information from other detectors and export useful information to them. Although a new detector normally knows the specific information that it will import and export, it is unfamiliar with what other detectors can import and export.

A new detector will be in pending state before it is completely settled down in the multi-tier structure. During the joining procedure, it must apply for the admission to join the system, determine which existing detectors that it will interact after it is added, and contact those detectors to set up the channels.

New detector admission is controlled by a registration daemon. This is particularly important when the multi-tier structure is distributed over several machines and the new detector will be running on a different machine from the registration daemon. As a separate issue, we omit the admission control details in this paper.

Furthermore, for every detector, the registration daemon records what information the detector imports and exports. Thus, based on the specific information that a new detector imports and exports, the registration daemon can determine which tier that this new detector should reside and also return a list of existing detectors that the new detector will interact. Meanwhile, the registration daemon will also record what this new detector imports and exports.

After a new detector is notified which tier it should locate itself in the multi-tier structure, its consumer and producer will then try to establish the channel with relevant detectors.

3.2 Connecting with relevant detectors

Knowing which existing detectors a new detector will import information from, the new detector's consumer will request these detectors to build an incoming pipe with each of them. The consumer can describe the specific information that it needs from an existing detector and the format that it prefers. The producer of the detector, upon hearing such a request, will decide whether it will accept the request; if so, it will acknowledge the new detector. In the case that this new detector imports information from primitive sources, such request is automatically agreed, and the new detector will actually fetch necessary information, instead of passively listening.

On the other hand, a new detector should also produce information to other existing detectors. For this purpose, the new detector's producer will notify other relevant existing detectors what it can export. At the same time, an existing detector's consumer is listening to the notification. If the existing detector decides it needs the exported information from the new detector, it sends back an acknowledgement. Upon receipt of each acknowledgement, the connection from the new detector toward another detector is established. In case there are no existing detectors willing to accept information from this new detector, the new detector directly exposes its output to system administrators.

3.3 Protocol specification

Communication among detectors requires a protocol to be defined. We call it external detector interface, or **EDI**. Two types of information are exchanged between detectors: control and data. Control information includes those that are used to manage buffer space, request input information, notify output information, and so on. Data information, on the other hand, is the real information that a detector imports from or export to another detector and that contains real intrusion detection information.

3.4 Extensibility

The simplest composition of this multi-tier intrusion detection system can consist of only the primitive source, one detector, and the system administrators. (In extreme case, you can even think that the primitive source and the system administrators can simply form an intrusion detection system, but either the system administrators have to be very smart to understand the primitive information, or the primitive information has to be clearly simple to tell whether an intrusion has happened or not.)

The extensibility lies in the fact that a detector can be easily added or removed from the structure. When the intrusion detection capability needs to be strengthened, a detector can be added. As discussed before, via its consumer, the new detector can subscribe itself to other detectors on what specific information that it accepts, and via its producer, it can publish to other detectors on what specific information that it exports.

Each detector of the integrated intrusion detection system functions independently of other detectors, with the exception of information exchange. Even within a detector, each individual component can be treated separately since it is only loosely coupled together via IDI. This clear-cut boundary makes the system modular. Each detector or each component inside a detector can be designed totally differently and is easy, thus, to replace.

It is apparent as long as a detector follows the external detector interface (**EDI**), it then can smoothly communicate with each other, no matter whatever implementation is inside each detector. Also, it is not mandatory to run the whole system on a single machine. In other words, each detector can run on different machines as long as the output from one detector can be transmitted to another, resulting in a distributed intrusion detection system. Instead of supervising a single machine, it can monitor a wider area where these detectors are distributed across the area, perhaps with some replication. The multi-tier

structure will still be maintained, except that the final analysis outcome can be the status of more than one machine, such as all machines on a local area network. Such a distributed system also has the potential of improving the performance or load balancing of the whole system.

4 DISCUSSION

4.1 Policy enforcement

Each detector may have a policy to determine in detail how a detector processes information. For instance, a detector that monitors login attempts may have a policy "send a warning signal if three continuous logins fail," or a system administrator that is regarded as a special detector may have a human-level policy. In particular, each detector's policy is allowed to be different, and all detectors' policies combined will collectively determine the system behavior and produce the intrusion signals that the system administrators finally receive.

Whereas a detector typically has a default policy, a detector's policy should be flexible to change. As an example, for a detector that searches for known patterns from input, being able to easily add new patterns is very important in order to detect new intrusions. However, changing a policy is not simply configuration parameter modification; due to policy change, the information to import or export will also be different, potentially causing further policy changes of other detectors.

4.2 Efficiency

Another important consideration is the efficiency of such a multi-tier intrusion detection system. Typically in this system all detectors will be running concurrently, each as an independent execution unit, such as a process or a thread. Whereas the efficiency of each individual detector is determined by the specific method that is employed by the detector, the efficiency when all detectors run concurrently relies on how the operating system schedules multiple execution units. In addition, a detector can be dynamically loaded or unloaded to avoid extra resource usage by detectors that are currently not needed. Finally, to balance resource usage between different machines, multiple detectors can be evenly distributed among them, as long as each individual detector that runs on a specific machine obeys the EDI specification and is able to authenticate itself to other detectors. For instance, a detector that needs lots of resources can be put in a specialized machine, while a detector that does not particularly care about performance can be assigned to a slower machine.

4.3 Security considerations

When the detectors in a multi-tier structure are distributed over multiple machines, the communication between the detectors on different machines becomes crucial. First, a detector must authenticate itself first in order to establish connection with another detector on a different machine. Second, the messages between detectors, after the connection is established, must also be protected.

One approach is to use public key cryptography, where each detector has a public key and a private key and a detector's authenticity can be certified by a certificate server. To protect the messages crossing machines, a detector can sign its message using its private key, while other detectors, who know the public key of this detector, can verify the signature. The messages can also be encrypted. A detector must then make its public key known to those detectors that it interacts. This can be done when the detector subscribes to the multi-tier structure by piggybacking a certificate of its public key, either on the information importation request to other detectors or on exportation capability.

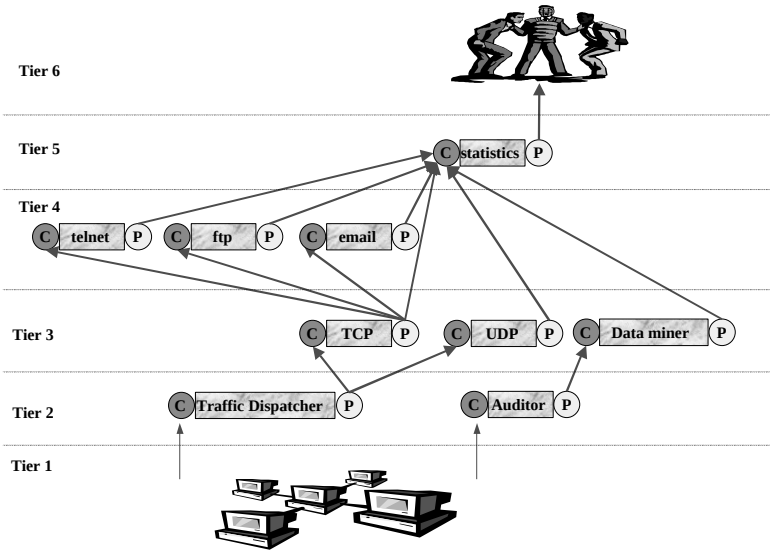

Figure 2. A multi-tier IDS example

5 EXAMPLE

In this section we show an example for a multi-tier intrusion detection system, as in Figure 2. On tier 1 is a primary source and on tier 6 are system administrators. On tier 2, there are two detectors. One is an Internet traffic dispatcher that inspects traffic at MAC (media access control) and IP (Internet Protocol) levels and dispatches TCP traffic and UDP traffic to detectors on tier 3, the TCP detector and the UDP detector, respectively. Another detector is an auditor that reads the system audit file and exports related information to the data miner detector on tier 3. On tier 3, a TCP detector checks TCP sessions and further asks some application specific detectors on tier 4 to inspect the traffic to check application-specific intrusions. Here in the example a statistics detector on tier 5 is employed to run statistics on events happening in the system; it also presents system administrators intrusion signals.

6 RELATED WORK

Intrusion detection is critical to computer security. As an example of the use of anomaly detection, data mining approaches [8] are proposed to detect intrusions in which learning agents continuously compute and provide the updated detection models to the detection agents. SRI developed NIDES [9] using an expert system approach, combining both misuse detection and anomaly detection techniques. At Purdue, Color Petri Nets is used to build an intrusion signature-matching model to detect misuse intrusion detection[7]; based on this, a mainly host-based intrusion detection system called IDIOT [3] has been built.

Networking intrusion detection has been particularly important, since more and more attacks are involving network applications. Bro [10] is a system developed by Lawrence Berkeley National Laboratory. The system is a stand-alone system for detecting network intruders by passively monitoring a network. It has an event engine to reduce network traffic into events, which are further analyzed by event handlers written in a so-called Bro language. NetProwler [13] from Axent is another sniffer-based network intrusion detection system that inspects traffic to see if any intrusion pattern can be matched. There are also many other commercial intrusion detection systems doing similar work, such as the

BlackICE[14] from Network ICE, NetRanger from Cisco [15], RealSecure from ISS [16], Network Flight Recorder from NFR [17], Dragon from NSW [18], etc. However, as a sniffer -based intrusion detection system, there is not enough information "on the wire" for a sniffer to reconstruct reliably what any two machines are talking, and it is subject to denial of service attacks.

There are also some specific intrusion related problems, such as how to identify the attacker of a network-based intrusion, how to detect intrusions over a network infrastructure, particularly the routers, how to make IP -spoofing difficult, and so on. DecIdUouS [1] identifies the source of network -based intrusions in a decentralized way. Ji -Nao [5] aims to stop intruders from breaking into network routers, switches and network management channels based on logical and statistical analysis of network routing and management protocols. Routing based on both source and destination address is also proposed to prevent packets with spoofed source addresses to be routed.

Recognizing there are so many different types of intrusions and different approaches, GrIDS (Graph-Based Intrusion Detection System) [2] developed at University of California at Davis features a hierarchical decomposition of the protected domains and investigates the characteristics with large -scale intrusion. CIDF (Common Intrusion Detection Framework) [6] proposes different intrusion detection systems to cooperate with each other and deal with diverse attacks across network and time, basically a peer-to-peer model trying to incorporate existing techniques together.

7 FUTURE WORK

Further research on the multi-tier intrusion detection structure is still needed. One significant issue is EDI and IDI specification. The second one is on more efficient passing of information from one detector to another, since copying data from a detector to another will require considerable effort if there are large amounts of data (network traffic handling from tier to tier as an example). There are also issues on how to make the system more scalable.

8 CONCLUSION

In this paper we have discussed designing a multi -tier intrusion detection system in which a detector is the basic building unit. A detector has a consumer, a filter and a producer. The filter specializes in amplifying intrusion signals and reducing the noise information, obtaining imported information from other detectors via the consumer and forwarding the result to other detectors via the producer.

This structure can be dynamically formed through the joining procedure of new detectors. External detector interface and internal detector interface has to be followed. Such a structure is very flexible, incorporating and combining different detection policies and methods, so that intrusion detection can be analyzed comprehensively. Our discussion also shows that such a design is extensible and can easily cross network to make aggregate analysis over a large number of machines in a distributed fashion.

REFERENCES

[1] H.Y.Chang, etc. DecIDUouS: Decentralized Source Identification for Network-Based Intrusion. July, 1998.

[2] Steven Cheung, Rick Crawford, Mark Dilger, Jeremy Frank, Jim Hoagland, Karl Levitt, Jeff Rowe, Stuart Staniford -Chen, Raymond Yip, Dan Zerkle, **The Design of GrIDS: A Graph -Based Intrusion Detection System.** Department of Computer Science, University of California at Davis. January 26, 1999.

[3] Mark Crosbie, Bryn Dole, Todd Ellis, Ivan Krsul, Eugene Spafford. **IDIOT – Users Guide.** Technical Report TR -96-050m, Purdue University. September, 1996.

[4] Dorothy E Denning. **An Intrusion Detection Model**. In IEEE Transactions on Software Engineering, No 2, page 222, February 1987.

[5] Y. Frank Jou, Fengmin Gong, Chandru Sargor, Shyhtsun Felix Wu, **Architecture Design of a Scalable Intrusion Detection System for the Emerging Network Infrastructure**. Technical Report CDRL A005, MCNC, Information technologies Division, Research Triangle Park, N.C. 27709.

[6] Clifford Kahn, Phillip A. Porras, Stuart Staniford-Chen, Brian Tung, **A Common Intrusion Detection Framework**. July 1998.

[7] Sandeep Kumar and Eugene H. Spafford, **An Application of Pattern Matching in Intrusion Detection** , Technical Report CSD-TR-94-013, The COAST Project, Department of Computer Sciences, Purdue University, June 1994.

[8] Wenke Lee and Salvatore J. Stolfo. **Data Mining Appro aches for Intrusion Detection** . Computer Science Department, Columbia University. http://www.cs.columbia.edu/~sal/hpapers/USENIX/usenix.html/

[9] Teresa F Lunt. **A survey of intrusion detection techniques**. In Computers and Security, 12(1993), pages 405-418.

[10] Vern Paxson. **Bro: A System for Detecting Network Intruders in Real-Time**. Proceedings of the 7th USENIX Security Symposium, San Antonio, Texas, January 26-29, 1998.

[11] Aurobindo Sundaram, **An Introduction to Intrusion Detection**.

[12] **FAQ: Network Intrusion Detection Systems**. http://www.ticm.com/kb/faq/idsfaq.html.

[13] **NetProwler**. Axent Technologies, Inc. http://www.axent.com/product/netprowler/default.htm

[14] **BlackIce.** Network ICE Corporation. http://networkice.com/Products/BlackICE/default.htm

[15] **NetRanger**. Cisco. http://www.cisco.com/univercd/cc/td/doc/pcat/211.htm

[16] **RealSecure**. Internet Security Systems, Inc. http://www.iss.net/prod/rs.php3

[17] **Network Flight Recorder**. Network Flight Recorder, Inc. http://www.nfr.net/products/ida-facts.html

[18] **Dragon**. http://www.network-defense.com/

[19] Computer Emergency Response Team. "CERT Advisory CA -2000-01 Denial -of-Service Developments," http://www.cert.org/advisories/CA-2000-01.html, January 2000.