Chapter 2

INTRUSION DETECTION: A SURVEY

Aleksandar Lazarevic, Vipin Kumar, Jaideep Srivastava
Computer Science Department, University of Minnesota

Abstract: This chapter provides the overview of the state of the art in intrusion detection research. Intrusion detection systems are software and/or hardware components that monitor computer systems and analyze events occurring in them for signs of intrusions. Due to widespread diversity and complexity of computer infrastructures, it is difficult to provide a completely secure computer system. Therefore, there are numerous security systems and intrusion detection systems that address different aspects of computer security. This chapter first provides taxonomy of computer intrusions, along with brief descriptions of major computer attack categories. Second, a common architecture of intrusion detection systems and their basic characteristics are presented. Third, taxonomy of intrusion detection systems based on five criteria (information source, analysis strategy, time aspects, architecture, response) is given. Finally, intrusion detection systems are classified according to each of these categories and the most representative research prototypes are briefly described.

Keywords: intrusion detection, taxonomy, intrusion detection systems, data mining.

1. INTRODUCTION

With rapidly growing adoption of the Internet, networked computer systems are playing an increasingly vital role in our society. Along with the tremendous benefits that the Internet brings, it also has its dark side. Specifically, new threats are created everyday by individuals and organizations that attack and misuse computer systems. As reported by the Computer Emergency Response Team/Coordination Center (CERT/CC) [37], the number of computer attacks has increased exponentially in the past few years (Figure 2-1). In addition, the severity and sophistication of the attacks is also growing (Figure 2-2). For example, Slammer/Sapphire Worm was the fastest computer worm in history. As it began spreading throughout the Internet, it doubled in size every 8.5 seconds and infected at least 75,000 hosts causing network outages and unforeseen consequences such as canceled airline flights, interference with elections, and ATM failures [153]. Earlier, the intruders needed profound understanding of computers and networks to launch attacks. However, today almost anyone can exploit the vulnerabilities in a computer system due to the wide availability of attack tools (Figure 2-2).

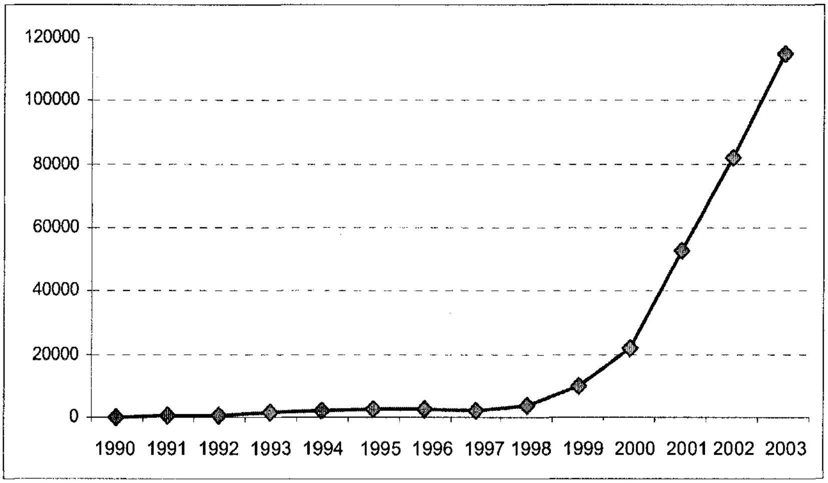

Figure 2-1. Growth rate of cyber incidents reported to Computer Emergency Response Team/Coordination Center (CERT/CC)

The conventional approach for securing computer systems is to design security mechanisms, such as firewalls, authentication mechanisms, Virtual Private Networks (VPN), that create a protective "*shield*" around them. However, such security mechanisms almost always have inevitable vulnerabilities and they are usually not sufficient to ensure complete security of the infrastructure and to ward off attacks that are continually being adapted to exploit the system's weaknesses often caused by careless design and implementation flaws. This has created the need for security technology that can monitor systems and identify computer attacks. This component is called intrusion detection and is a complementary to conventional security mechanisms.

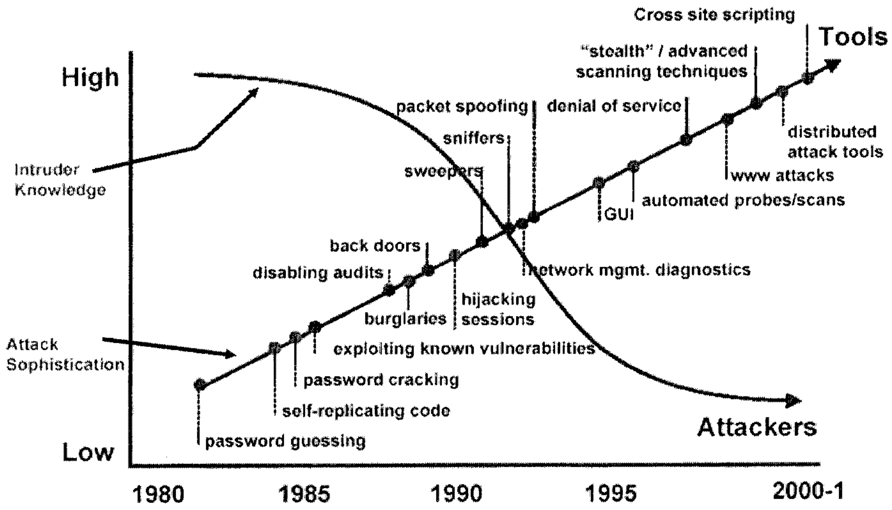

Figure 2-2. Attack sophistication vs. Intruder technical knowledge (source: http://www.cert.org/present/internet-security-trends)

The National Institute of Standards and Technology classifies intrusion detection [15] as "*the process of monitoring the events occurring in a computer system or network and analyzing them for signs of intrusions, defined as attempts to compromise the confidentiality, integrity, availability, or to bypass the security mechanisms of a computer or network*".

Intrusions in computer systems are usually caused by attackers accessing the systems from the Internet, or by authorized users of the systems who attempt to misuse the privileges given to them and/or to gain additional privileges for which they are not authorized. An Intrusion Detection System (IDS) can be defined as a combination of software and/or hardware components that monitors computer systems and raises an alarm when an intrusion happens.

This chapter provides an overview of the current status of research in intrusion detection. It first provides an overview of different types of computer intrusions, and then introduces a more detailed taxonomy of intrusion detection systems with an overview of important research in the field. Both taxonomies are illustrated and supported with several well known examples of computer attacks and intrusion detection techniques. Several surveys in the intrusion detection have been published in the past [4, 13, 31, 55, 92, 97, 110, 114, 136]. However, the growth of the field has been very rapid, and many new ideas have since emerged. The survey in this chapter attempts to build upon these earlier surveys, but is more focused on intrusion detection projects proposed in academic institutions and research organizations than on commercial intrusion detection systems, primarily due to the lack of detailed technical information available on commercial products. The reader interested in commercial IDSs is referred to a survey of IDS products [92] and to web sites that maintain lists of such systems [57, 76].

2. TAXONOMY OF COMPUTER ATTACKS AND INTRUSIONS

Research community in computer security has developed numerous definitions of computer attacks and intrusions. One of the most popular definitions for intrusion [181] is that it represents a "*malicious, externally induced, operational fault*". Computer intrusions and attacks are often considered synonymous. However, other definitions of the word "attack" that differentiate it from intrusion have also been proposed in the intrusion detection literature. For example, a system can be attacked (either from the outside or the inside), but the defensive "*shield*" around the system or resource targeted by the attack may be sufficiently effective to prevent intrusion. Therefore, we may say that an attack is an intrusion attempt, and an intrusion results from an attack that has been (at least partially) successful [181].

There have been numerous attempts to categorize and classify computer attacks and intrusions [11, 112, 115, 128, 135]. Some of these attempts have provided formally developed taxonomies and specified a certain set of properties that the taxonomy should satisfy, e.g., they should be: (i) logical and intuitive [84], (ii) based on solid technical details [23], (iii) comprehensible [128], (iv) complete [5], (v) exhaustive [84, 128], (vi) mutually exclusive [84, 128], (vii) objective [108], (viii) repeatable [84, 108], and (ix) useful [84, 128]. For more details on these characteristics, the reader is referred to the above publications, as well as to Lough's PhD thesis [135].

Initial work in categorizing different aspects of computer security focused on weaknesses in computer systems and design flaws in operating systems [12], as well as functional vulnerabilities and computer abuse methods [172]. Several taxonomies that were developed later mainly focused on two issues: (i) categorization of computer misuse (i.e. attacks) and (ii) categorization of the people trying to get unauthorized access to computers (perpetrators), and the objectives and results of these attempts.

In one of earlier attempts for describing types of computer attacks, Neumann and Parker developed the SRI Computer Abuse Methods Model [165, 166, 173], which outlines about 3000 attack cases and computer misuses collected over nearly twenty years and categorizes them into the nine-level tree of attack classes. Lindqvist and Jonsson [128] extended the Neumann and Parker model by expanding several attack categories (categories 5, 6 and 7 from original nine-level tree of attacks) and by introducing the concept of dimension, which represents a basis of the attack classification. They specified two interesting criteria for system owners to perform attack classification, namely "*intrusion techniques*" and "*intrusion results*", and they called these criteria dimensions. Jayaram and Morse [96] also developed a taxonomy of security threats to networks, in which they provide five "classes of security threats" and two "classes of security mechanisms". Another significant work in computer attack taxonomies is performed by the CERIAS group at Purdue University [11, 108, 112]. Their first attempt [112] provided a classification of computer intrusions on Unix systems using system logs and colored Petri nets. Aslam [11] extended this work by providing a taxonomy of security flaws in Unix systems. Finally, Krsul [108] reorganized both previous taxonomies and provided a more complex taxonomy of computer attacks that contains four main categories (design, environmental assumptions, coding faults and configuration errors). Richardson [189, 190] extended these taxonomies by developing a database of vulnerabilities to help study of the problem of Denial of Service (DoS) attacks. The database was populated with 630 attacks from popular sites that report computer incidents. These attacks were cataloged into the categories that correspond to extensions from Aslam's taxonomy of security flaws [11] and Krsul's taxonomy of computer attacks [108]. Within the DARPA intrusion detection project, Kendall [103] developed a similar database of computer attacks that exist in DARPA intrusion detection evaluation data sets [52]. An excellent overview of these techniques as well as their extensions is provided in Lough's PhD thesis [135].

Anderson presented one of the first categorizations of attack perpetrators according to their types. He used a 2x2 table to classify computer threats into three groups (external penetration, internal penetration and misfeasance), based on whether or not penetrators are authorized to use the computer system or to use particular resources in the system [7]. One of the most influential taxonomies in categorizing attack perpetrators is the classification of types of attackers, used tools, access information, attack consequences and the objectives of the attacks, performed by CERT [84]. Researchers at Sandia National Laboratories [45] proposed a very similar taxonomy, with a few added or merged categories.

The taxonomy we provide in this survey is more general, and is obtained by examining and combining existing categorizations and taxonomies of host and network attacks published in the intrusion detection literature, and by revealing common characteristics among them. In previously published taxonomies, categories used in classification of attacks were usually either a cause of a vulnerability or the result (i.e., effect) of a vulnerability. In the taxonomy proposed here, we use traditional cause of vulnerability to specify the following categories of attacks:

- Attack type
- Number of network connections involved in the attack
- Source of the attack
- Environment
- Automation level

Attack type. The most common criterion for classifying computer attacks and intrusions in the literature is according to the attack type [84, 103]. In this chapter, we categorize computer attacks into the following classes:

– **Denial of Service (DoS) attacks**. These attacks attempt to "*shut down a network, computer, or process; or otherwise deny the use of resources or services to authorized users*" [144]. There are two types of DoS attacks: (i) operating system attacks, which target bugs in specific operating systems and can be fixed with patches; and (ii) networking attacks, which exploit inherent limitations of networking protocols and infrastructures. An example of operating system attack is teardrop, in which an attacker exploits a vulnerability of the TCP/IP fragmentation re-assembly code that do not properly handle overlapping IP fragments by sending a series of overlapping packets that are fragmented. Typical example of networking DoS attack is a "SYN flood" attack, which takes advantage of three-way handshake for establishing a connection. In this attack, attacker establishes a large number of "half-open" connections using IP spoofing. The attacker first sends SYN packets with the spoofed (faked) IP address to the victim in order to establish a connection. The victim creates a record in a data structure and responds with SYN/ACK message to the spoofed IP address, but it never receives the final acknowledgment message ACK for establishing the connection, since the spoofed IP addresses are unreachable or unable to respond to the SYN/ACK messages. Although the record from the data structure is freed after a time out period, the attacker attempts to generate sufficiently large number of "half-open" connections to overflow the data structure that may lead to a segmentation fault or locking up the computer. Other examples of DoS attacks include disrupting connections between machines thus preventing access to a service, preventing particular individuals from accessing a service, disrupting service to a specific system or person, etc. In distributed DoS (DDoS) attack, which is an advanced variation of DoS attack, multiple machines are deployed to attain this goal. DoS and DDoS attacks have posed an increasing threat to the Internet, and techniques to thwart them have become an active research area [151, 152, 154, 169, 171, 176, 226]. Researchers that analyze DoS attacks have focused on two main problems: (i) early detection mechanisms and identification of ongoing DoS activities [41, 75, 218, 235]; and (ii) response mechanisms for alleviating the effect of DoS attacks (e.g. damage caused by the attack). Response mechanisms include identifying the origin of the attack using various traceback techniques [27, 91, 195, 206] and slowing down the attack and reducing its intensity [141, 151, 248] by blocking attack packets. In addition to these two main approaches, some systems use measures to suppress DoS attacks. For example, CenterTrack [218] is an overlay network that uses selective rerouting to trace the entrance points of large flooding attack, while SOS (Secure Overlay Services) [104] employs a combination of "secure overlay tunneling, routing via consistent hashing, and filtering" to proactively prevent large flooding DoS attacks.

- ***Probing (surveillance, scanning)***. These attacks scan the networks to identify valid IP addresses (Figure 2-3) and to collect information about them (e.g. what services they offer, operating system used). Very often, this information provides an attacker with the list of potential vulnerabilities that can later be used to perform an attack against selected machines and services. Examples of probing attacks include IPsweep (scanning the network computers for a service on a specific port of interest), portsweep (scanning through many ports to determine which services are supported on a single host), nmap (tool for network mapping), etc. These attacks are probably the most common ones, and are usually precursor to other attacks. The existing scan detection schemes essentially look for IP addresses that make more than N connections in T seconds. These schemes are very good at picking out fast and disperse noisy scans. Unfortunately, tools based on these techniques are quite inefficient at detecting slow/stealthy scans or scans targeted specifically at the monitored enterprise - the type of scans that analysts would really be interested in. Stealthy scans can be defined as scans that would normally not trigger typical scan alert technology. Due to these reasons, sophisticated adversaries typically attempt to adjust their scans by reducing the frequency of their transmissions in order to avoid detection. For detecting stealthy scans, there are a few recently proposed more sophisticated technique based on collecting various statistics [62, 102, 147, 191, 214, 222].

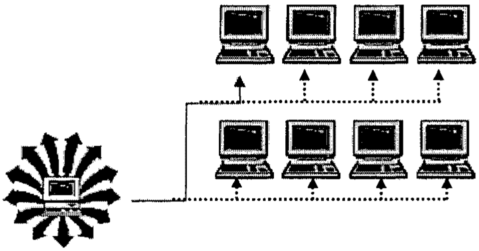

Figure 2-3. Typical scanning activity

- ***Compromises.*** These attacks use known vulnerabilities such as buffer overflows [38] and weak security points for breaking into the system and gaining privileged access to hosts. Depending upon the source of the attack (outside attack vs. inside attack), the compromises can be further split into the following two categories:
  - *R2L (Remote to Local) attacks*, where an attacker who has the ability to send packets to a machine over a network (but does not have an account on that machine), gains access (either as a user or as the root) to the machine. In most *R2L* attacks, the attacker breaks into the computer system via the Internet. Typical examples of R2L attacks include guessing passwords (e.g. guest and dictionary attacks) and gaining access to computers by exploiting software vulnerability (e.g. phf attack, which exploits the vulnerability of the phf program that allows remote users to run arbitrary commands on the server).
  - *U2R (User to Root) attacks*, where an attacker who has an account on a computer system is able to misuse/elevate her or his privileges by exploiting a vulnerability in computer mechanisms, a bug in the operating system or in a program that is installed on the system. Unlike *R2L* attacks, where the hacker breaks into the system from the outside, in *U2R* compromise, the local user/attacker is already in the system and typically becomes a root or a user with higher privileges. The most common U2R attack is buffer overflow, in which the attacker exploits the programming error and attempts to store more data into a buffer that is located on an execution stack. Since buffers are created to contain a specific amount of data, the additional information used by the attacker can overflow into adjacent buffers, corrupting or overwriting the valid data held in them. This data may contain codes designed to trigger specific actions, such as damaging user's files or providing the user with root access. Many approaches have recently been proposed for detection and prevention of buffer overflow attacks [49, 71], due to increased

interest in them. It is important to note that buffer overflow attacks can also belong to R2L attacks, where remote users attempts to compromise the integrity of target computer. For example, a vulnerability discovered in Microsoft Outlook and Outlook Express in July 2000 [35] allowed the attackers to simply send an e-mail message and to overflow the specific areas with superfluous data, which allowed them to execute whatever type of code they desired on the recipient's computers.

- ***Viruses/Worms/Trojan horses*** are programs that replicate on host machines and propagate through a network.
    - *Viruses* are programs that reproduce themselves by attaching them to other programs and infecting them. They can cause considerable damage (e.g. erase files on the hard disk) or they may only do some harmless but annoying tricks (e.g. display some funny messages on the computer screen). Viruses typically need human interaction (e.g. trading files on a floppy or opening e-mail attachments) for replication and spreading to other computers. One of the most well known virus examples is Michelangelo virus that infects the hard disk's master boot record and activates a destructive code on March 6, which is Michelangelo's birthday. There are various types of viruses, and classifying them is not easy as many viruses have multiple characteristics and may fall into multiple categories. The most common virus classification is according to the environment, operating system, different algorithms of work and destructive capabilities [150], although there are other categorizations based on what and how viruses infect [48, 87].
    - *Worms* are self-replicating programs that aggressively spread through a network, by taking advantage of automatic packet sending and receiving features found on many computers. Worms can be organized into several categories [105, 215, 236]:
        - *traditional worms* (e.g. Slammer [37]) usually use direct network connections to spread through the system and do not require any user interaction.
        - *e-mail (and other client application) worms*, (e.g. Melissa worm [34]) infect other hosts on the network (Internet) by exploiting user's e-mail capabilities or utilizing other client applications (e.g. ICQ – "I seek you").
        - *windows file sharing worms* (e.g. ExploreZip [221]) replicate themselves by utilizing MS Windows peer-to peer service, which is activated every time a networking device is detected in the system. This type of a worm very often occurs in

combination with other attacks, such as MS-DOS and Windows viruses.

- *hybrid worms* (e.g. Nimda [36]) typically exploit multiple vulnerabilities that fall into different categories specified above. For example, Nimda used many different propagation techniques to spread (e-mail, shared network drives and scanning for backdoors opened by the Code Red II and Sadmind worms). Success of Nimda demonstrated that e-mail and http traffic are effective ways to penetrate the network system, and that the file sharing is quite successful in replicating within the system [236].

It is important to note that some of the worms that appeared recently have also been used to launch DoS attacks [83]. For example, the erkms and li0n worms were used to deploy DDoS tools via BIND vulnerabilities [83], while Code Red was used to launch TCP SYN DoS attacks [83]. However, traditional DoS attacks typically target a single organization, while worms (e.g. SoBig.F worm) typically affect a broad range of organizations. Over the last few years, many DoS attacks have gradually mutated and merged with more advanced worms and viruses (e.g. Blaster worm in August 2003). Analysts also expect that in the future DoS attacks will be more often part of worm payloads [83].

- *Trojan horses* are defined as "malicious, security-breaking programs" that are disguised as something benign [134]. For example, the user may download a file that looks like a free game, but when the program is executed, it may erase all the files on the computer. Victims typically download Trojan horses from an archive on the Internet or receive them via peer-to-peer file exchange using IRC/instant messaging/Kazaa etc. Some actual examples include Silk Rope and Saran Wrap.

Many people use terms like Trojan horse, viruses and worms interchangeably since it is not easy to make clear distinction between them. For example, “Love Bug" is at the same time a virus, worm, and Trojan horse. It is a trojan horse since it pretends to be a love letter but it is a harmful program. It is a virus because it infects all the image files on the disk, turning them into new Trojan horses. Finally, it s also a worm since it propagates itself over the Internet by hiding in trojans that it sends out using peoples’ email address book, IRC client, etc.

Number of network connections involved in an attack. Attacks can be classified according to the number of network connections involved in the attack:

- Attacks that involve *multiple network connections*. Typical examples of such attacks are DoS, probing and worms (Figure 2-3).

- Attacks that involve *a single or very few network connections*. Typical attacks in this category usually cause compromises of the computer system (e.g. buffer overflow).

Source of the attack. Computer attacks may be launched from a *single location* (*single source attacks*) or from *several different locations* (*distributed/coordinated attacks*). Most of the attacks typically originate from a single location (e.g. simple scanning), but in the case of large distributed DoS attacks or other organized attacks, multiple source locations may participate in the attack. In addition, very often distributed/coordinated attacks are targeted not only to a *single* computer, but also to *multiple destinations*. Detecting such distributed attacks typically requires the analysis and correlation of network data from several sites.

Environment. Attacks may be categorized according to the environment where they occur:

- *Intrusions on the host machine* are intrusions that occur on a specific machine, which may not even be connected to the network. These attacks are usually detected by investigating the system information (e.g. system commands, system logs). The identity of the user that performs an attack in this case is typically associated with the username, and is therefore easier to discover.
- *Network intrusions* are intrusions that occur via computer networks usually from outside the organization. Detection of such intrusions is performed by analyzing network traffic data (e.g. network flows, tcpdump data). However, such analysis often cannot reveal the precise identity of the attackers, since there is typically no direct association between network connections and a real user.
- *Intrusions in a P2P environment* are intrusions that occur in a system where connected computers act as peers on the Internet. Unlike standard "client/server" network architectures, in P2P environment, the computers have equivalent capabilities and responsibilities and do not have fixed IP address. They are typically located at "*the edges of the Internet*" [240], and actually disconnected from the DNS systems. Although P2P file sharing applications can increase productivity of enterprise networks, they can also introduce vulnerabilities in them, since they enable users to download executable codes that can introduce rogue or untraceable "backdoor" applications on users' machines and jeopardize enterprise network security.
- *Intrusions in wireless networks* are intrusions that occur between computers connected through wireless network. Detection of attacks in wireless networks is based on analyzing information about the connections in wireless networks, which is typically collected at wireless

access points [126]. In general, security threats in wireless networks can be categorized into:

- *eavesdropping*, when intruder only listens for the data;
- *intrusions*, when intruder attempts to access or to modify the data;
- *communication hijacking*, when a rogue node captures the channel, poses as a rogue wireless access point and attracts mobile nodes to connect to it and then collects confidential data from them (e.g. passwords, secret keys, logon names);
- *Denial of Service (jamming) attacks*, when an attacker disturbs the communication channel with various frequency domains (cordless phones, microwave ovens), physical obstacles and disables all communication on the channel.

Automation level. Depending on the level of the attack automation, there are several categories of attacks as follows:

- *Automated attacks* use automated tools that are capable of probing and scanning a large part of the Internet in a short time period. Using these easily available tools, even inexperienced attackers may create highly sophisticated attacks (Figure 2-2). Such attacks are probably the most common method of attacking the computer systems today.
- *Semi-automated attacks* deploy automated scripts for scanning and compromise of network machines and installation of attack code, and then use the handler (master) machines to specify the attack type and victim's address.
- *Manual attacks* involve manual scanning of machines and typically require a lot of knowledge and work. Manual attacks are not very frequent, but they are usually more dangerous and harder to detect than semi-automated or automated attacks, since they give to attackers more control over the resources. Experts or organized groups of attackers generally use these attacks for attacking systems of critical importance.

3. INTRUSION DETECTION SYSTEMS

Since the first model for intrusion detection was developed by Dorothy Denning [56] at SRI International, many intrusion detection systems (IDSs) have been proposed both in the research and commercial world. For information about these research and commercial products, the reader is referred to Web sites that contain links to them [32, 76, 149, 198, 223]. Although these systems are extremely diverse in the techniques they employ to gather and analyze data, most of them rely on a relatively general architectural framework (Figure 2-4), which consists of the following components:

- *Data gathering device* (sensor) is responsible for collecting data from the monitored system.
- *Detector* (*Intrusion Detection* (*ID*) *analysis engine*) processes the data collected from sensors to identify intrusive activities.
- *Knowledge base* (*database*) contains information collected by the sensors, but in preprocessed format (e.g. knowledge base of attacks and their signatures, filtered data, data profiles, etc.). This information is usually provided by network and security experts.
- *Configuration device* provides information about the current state of the intrusion detection system (IDS).
- *Response component* initiates actions when an intrusion is detected. These responses can either be automated (active) or involve human interaction (inactive).

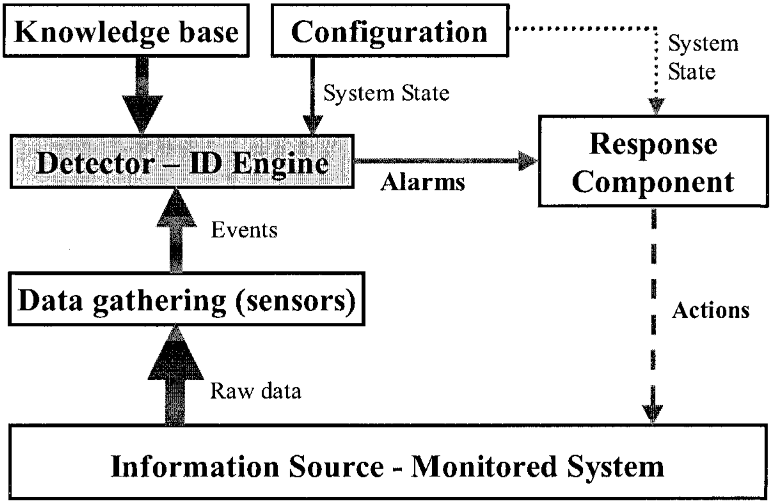

Figure 2-4. Basic architecture of intrusion detection system (IDS)

3.1 Characteristics of Intrusion Detection Systems

A number of desired characteristics for intrusion detection systems (IDSs) have been identified [55, 180], as follows:

- *Prediction performance.* In intrusion detection, simple performance measure such as prediction accuracy is not adequate. For example, the network intrusions typically represent a very small percentage (e.g. 1%) of the entire network traffic, and a trivial IDS that labels all network traffic as normal, can achieve 99% accuracy. In order to have good prediction performance, an IDS needs to satisfy two criteria: (i) it must be able to correctly identify intrusions and (ii) it must not identify legitimate action in a system environment as an intrusion. Typical measures for evaluating predictive performance of IDSs include detection rate and false alarm rate (Table 1). Detection rate is defined as the ratio of the number of correctly detected attacks and the total number of attacks, while the false alarm (false positive) rate is the ratio of the number of normal connections that are incorrectly misclassified as attacks and the total number of normal connections. In practice, it is very difficult to evaluate these two measures, since it is usually infeasible to have global knowledge of all attacks. Since detection rate and false alarm rate are often in contrast, evaluation of IDSs is also performed using ROC (Receiver Operating Characteristics) analysis [183]. ROC curve represents a trade-off between detection rate and false alarm rate as illustrated in Figure 2-5. The closer the ROC is to the left upper corner of the graph (point that corresponds to 0% false alarm and 100% detection rate), the more effective the IDS is.

Table 2-1. Evaluations of intrusions (attacks)

		Predicted connection label	
		Normal	Intrusions (Attacks)
Actual connection label	Normal connections	True Negative (TN)	False Alarm (FP)
	Intrusions (Attacks)	False Negative (FN)	Correctly detected intrusions – True Positive (TP)

- *Time Performance.* The time performance of an intrusion-detection system corresponds to the total time that the IDS needs to detect an intrusion. This time includes the *processing time* and the *propagation time.* The *processing time* depends upon the processing speed of the IDS, which is the rate at which the IDS processes audit events. If this rate is not sufficiently high, then the real time processing of security events may not be feasible. The *propagation time* is the time needed for processed information to propagate to the security analyst. Both times need to be as short as possible in order to allow the security analyst sufficient time to react to an attack before much damage has been done, as well as to stop an attacker from modifying audit information or altering the IDS itself.

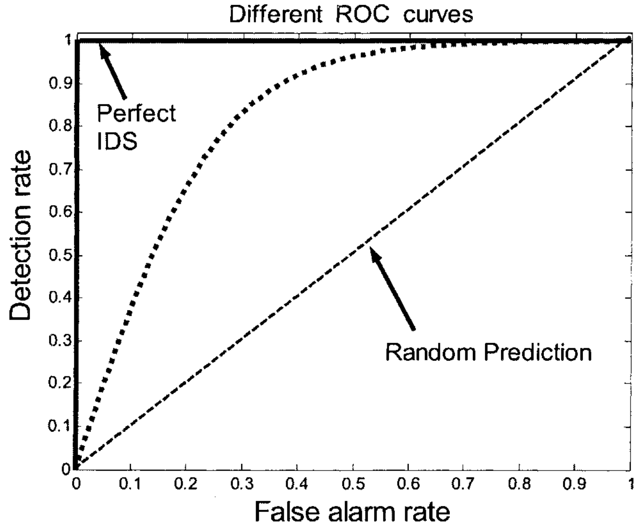

Figure 2-5. ROC Curves for different intrusion detection techniques

- *Fault tolerance.* An IDS should itself be dependable, robust and resistant to attacks, and should be able to recover quickly from successful attacks and to continue providing a secure service. This is especially true in the case of very large distributed DoS attacks, buffer overflow attacks and various deliberate attacks that can shut down the computer system and thus IDS too. This characteristic is very important for the proper functioning of IDSs, since most commercial IDSs run on operating systems and networks that are vulnerable to different types of attacks. In addition, IDS should also be resistant to scenarios when an adversary can cause the IDS to generate a large number of false or misleading alarms. Such alarms may easily have a negative impact on the availability of the system, and the IDS should be able to quickly overcome these obstacles.

3.2 Taxonomy of Intrusion Detection Systems (IDSs)

Several classifications of intrusion detection methods have been proposed in the past [4, 13, 55, 97, 110, 114, 136], but there is still no universally accepted taxonomy. In this chapter, we present a taxonomy that is based on the synthesis of a number of existing ones [13, 55]. We use five criteria to classify IDSs, as summarized in Figure 2-6.

The first criterion is *information (data) source*, which distinguishes IDSs based on the system that is monitored, i.e. source of input information (see Figure 2-4). The source information can be (i) audit trails (e.g. system logs) on a host, (ii) network connections/packets, (iii) application logs, (iv) wireless network traffic or (v) intrusion-detection and/or sensor alerts produced by other intrusion-detection systems.

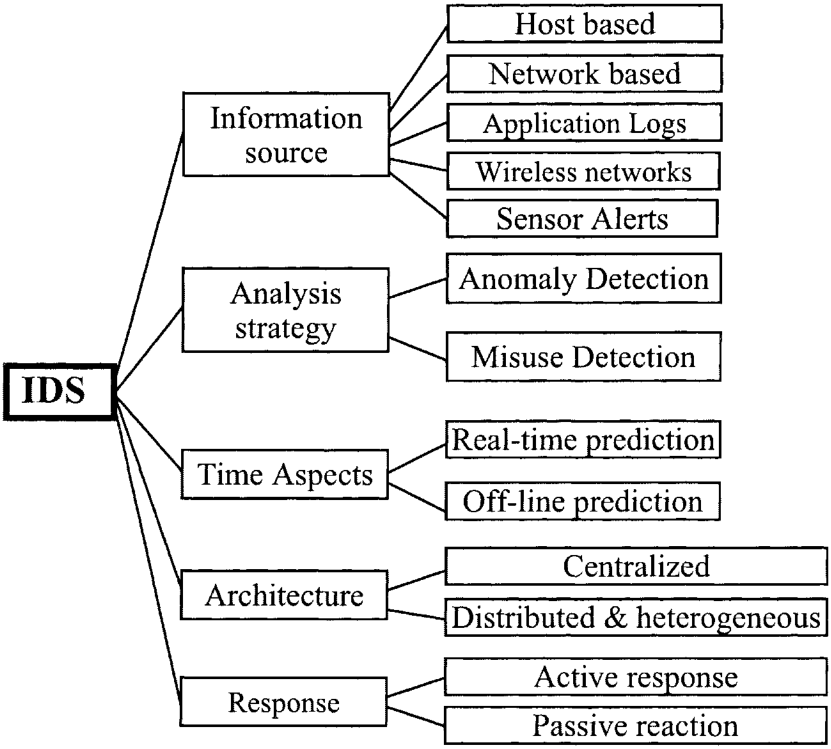

Figure 2-6. Taxonomy of intrusion detection systems according to proposed six criteria

The *analysis strategy* describes the characteristics of the detector (intrusion detection engine from Figure 2-4). When the IDS looks for events or sets of events that match a predefined pattern of a known attack, this analysis strategy is called misuse detection. When the IDS identifies intrusions as unusual behavior that differs from the normal behavior of the monitored system, this analysis strategy is called anomaly detection.

Time aspects are used to categorize the IDSs into on-line IDSs that detect intrusions in real time and off-line IDSs that usually first store the monitored data and then analyze it in batch mode for signs of intrusion.

The *architecture* of IDSs is used to differentiate between centralized IDSs that analyze the data collected only from a single monitored system and distributed IDSs that collect information from multiple monitored systems in order to investigate global, distributed and coordinated attacks.

Detection response describes the reaction of the IDS to an attack (intrusion). If the IDS reacts to the attack by taking corrective action (e.g. closing holes) or pro-active action (e.g. logging out possible attackers, closing down services), the response is called active. If the IDS only generates alarms (including paging security analysts) and does not take any actions, the response is called passive.

4. INFORMATION SOURCE

Early intrusion detection systems were largely host-based, since mainframe computers were common and all users were local to the system. In such an environment, intrusion-detection was focused only on insider threats, since interaction with outside world was quite rare. The audit information collected at the mainframe was analyzed either locally [137] or on a separate machine [204] and security-suspicious events were reported.

However, with the growth of computer networks, there has been an increasing focus on IDSs for the networked environment. Initial attempts of intrusion detection in a networked environment were focused on enabling communication among host-based intrusion-detection systems [93] and then exchanging information at several levels, either through a raw audit trail over the network [80, 204], or issuing alarms generated by local analysis [205].

In the late nineties, the intrusion detection research community debated the superiority of network-based vs. host-based approaches. However, today many systems attempt to provide an integrated tool by incorporating both variants. These IDSs are usually called hybrid IDSs. For example, in the distributed intrusion detection system (DIDS) developed by Snapp et al [205], Haystack [80, 204] is used on each host to detect local attacks, while network security monitor (NSM) [81] is employed to monitor the network. Both systems, Haystack and NSM, send information to the DIDS Director, where the final analysis is performed.

Network/host based IDSs typically analyze past network traffic and host OS activity, but they are unable to detect unauthorized use of specific applications. This caused the emergence of application-based IDSs that focus on monitoring interactions between a user and specific applications.

More recently, increasing popularity of wireless networks has caused intrusion detection researchers to focus on detecting attacks in wireless environment. Wireless network are highly sensitive and extremely insecure, as they are vulnerable to easy eavesdropping and jamming thus requiring additional security policies as well as specific intrusion detection techniques.

4.1 Host-based IDSs

Host based intrusion detection systems (IDSs) analyze users' activities and behavior on a given machine. Host-based IDSs have an advantage that they are able to work with high quality data that is typically very informative. However, depending upon the processing performed, host-based IDSs can significantly impact the performance of the machine they are running on. In addition, audit sources used in host-based intrusion analysis, can be easily modified by a successful attack, which represents another limitation of host-based IDSs. In order to alleviate these drawbacks, host-based IDSs have to process the audit trail sufficiently fast to be able to raise alarms before an attacker has an opportunity to observe and/or modify the audit trail or the intrusion-detection system itself.

There are several types of information that are typically used in host-based IDSs, e.g. (i) system commands, (ii) system accounting, (iii) syslog and (iv) security audit information.

4.1.1 System commands

System commands are a useful source of information that can be employed by host based IDSs for detecting malicious users [51, 116, 145, 193]. By analyzing system commands that users invoke in their sessions, it is possible to build user profiles, which describe users' characteristics and common behavior. Examples of such logged system commands in Unix are `ps, pstat, vmstat, getrlimit`. Information about different events provided by these commands can be very precise and informative. Since the audit information is collected as unstructured data, and has to be preprocessed before analysis.

4.1.2 System accounting

System accounting is present in both Windows and Unix operating systems. Although the interest for system accounting in Windows environment is increasing, there have not been many intrusion detection approaches that used this type of data for intrusion analysis. On the other hand, system accounting is commonly used in the Unix environment to collect information on system behavior, such as consumption of shared resources (e.g. processor time, memory, disk) by the users of the system. Data generated by system accounting can serve as a valuable and convenient source of information for IDSs [63].

There are two typical Unix accounting logs that are used for easy extraction of system behavioral information, without extensive kernel modifications often required for detailed auditing, namely: process accounting and login accounting. The standard file for storing the process accounting information is `pacct` or `acct`, while the standard file for the login accounting information is `wtmp`. Process accounting keeps track of information about a process at the time of process completion (e.g. user and group IDs of those that use the process, beginning and elapsed times of the process, CPU time for the process, amount of memory used). The login accounting (`wtmp`) system records information about users' login and logout from the system. When users successfully log in and log out or unsuccessfully attempt to login, the Unix kernel appends `utmp` structures to the log file.

Use of system accounting as a source of information for IDSs has several advantages. First, all Unix systems have the same format of the accounting records. Second, the time needed to store system accounting records is generally small, since information is compressed. Finally, system accounting is quite common in the modern operating systems, and it is easy to setup and use. However, using system accounting also has a few drawbacks that limit their use in security applications. First, in order to perform real time analysis of system accounting data, all historical profiles have to be compared to each currently active profile, which can be computationally intensive. This generally impacts the system load and therefore slows down potential statistical data analysis. Second, accounting is either enabled for all users or not enabled at all, and cannot be selectively activated only for particular individuals of interest. Third, system accounting logs require a large amount of disk storage, and hence, they must be periodically removed. Fourth, the accounting structures limit the length of recorded command name to only a fixed number of characters (typically eight), thus losing important information (e.g. common arguments are not recorded). Finally, the accounting data is recorded only when the application terminates, so continuously running executables such as system daemons (e.g. sendmail) are never audited (these applications have to be audited using syslogs). In such cases, it is only possible to perform off-line intrusion analysis.

Due to these drawbacks of system accounting, its use is not very common. Nevertheless, there are several systems that employ this information for intrusion detection [54, 63]. For example, the statistical and neural network modules in Hyperview [54] use system accounting only as additional information to security audit, but not as a substitute for it, while anomaly-based detection techniques in Eschrich's thesis [63] use accounting logs to identify imposters. Imposters are special class of intruders who are valid users in a system but gain illegal access to the account of other users.

4.1.3 System log information

System log data contains information that is not available at the network level, such as when users log in, when they send email, who they send email to, which ftp logs commands are issued, and which files are transferred. Capturing and collecting system log file information in a readable format is typically performed by the syslog daemon.

One of the major drawbacks of using syslog information for intrusion detection is that syslog information is not very secure, since several syslog daemons exhibit buffer overflow exploitation [33]. On the other hand, due to straightforward use of syslog, this information is widely employed by numerous network services and applications, such as `login`, `sendmail`, `nfs`, `http`, as well as security-related tools such as sudo, klaxon, or TCP wrappers [55]. For example, Swatch [78] and TkLogger [85] perform regular expression matching against system log files, search for certain patterns and take appropriate actions when they are found. These tools are especially useful for identifying things that may indicate very specific problems.

4.1.4 Security audit processing

The *security audit trails* represent records that contain all potentially important activities related to the security of the system. Since these activities are usually logged to a file in chronologically sorted order, their analysis could allow easier investigation of sequential intrusive patterns. One of the most popular security audit trails is BSM (Basic Security Module), auditing facility in Solaris operating system form Sun Microsystems Inc [219]. BSM monitors security related events and records the "*crossing of instructions executed by the processor in the user space and instructions executed in the kernel*" [219].

In general, the security audit trail can provide information about full system call traces, which includes detailed user and group identification, the parameters of system call execution, memory allocation, context switches, internal semaphores, and successive file reads that typically do not appear in the regular audit trail. In addition, advantages of using security audit data include strong user authentication, easier audit system configuration, and fine-grain parameterization of collected information [55]. On the other hand, drawbacks of using security audit trails include complex setup, intensive resource requirement and possible vulnerability to DoS attack due to filling audit file system [55].

Several research groups [77, 155, 180, 217] have been actively using security audit trails mainly for host-based intrusion detection systems. The focus of their research has been mainly to define what information the security audit trail should contain in order to increase the IDS prediction performance as well as to establish an acceptable common format for audit trail records.

4.2 Network-based information sources

With rapidly growing popularity of the Internet, there have been an increasing number of attacks aimed at the network itself (e.g. spoofing, TCP hijacking, port scanning, ping of death) that cannot be (at least not easily) detected by examining the host audit trail alone. These reasons have led to the development of specific tools that sniff network packets [161, 175, 224] in real time and facilitate searching for network attacks. In addition, by analyzing the payload of the packet, a number of typical attacks against servers can also be detected.

There are several advantages of using network based IDSs over host-based IDSs. First, network-based IDSs can be installed such that they do not have effect on existing computer systems or infrastructures. Second, they are usually more resistant than host-based IDSs, since they do not reside on the hosts that may be the targets of certain attacks. Third, the majority of network-based IDSs typically do not depend on the operating system that is used and can extract useful information at a network level (e.g. packet fragmentation). Finally, they can be installed at strategic points in a network (e.g. routers, borders) where they can be used to watch all traffic passing through these ports and therefore used to discover network attacks. However, their major drawbacks are their weak scalability, high possibility for dropping packets in fast networks under heavy load, and inability to perform intrusion detection when data is encrypted.

Network based intrusion detection systems analyze various kinds of information that are obtained by monitoring network infrastructures. Typical sources of such information are network connections/packets collected by network sniffers and management information between network devices collected due to use of Simple Network Management Protocol (SNMP).

4.2.1 Network connections and network packets

Network packet sniffers are commonly used for collecting information about events that occur on a network. Sniffers capture copies of network packets directly from the network interface and provide administrators with detailed information about the IP addresses of senders and receivers, the number of transferred packets/ bytes and other low-level information about those packets. Certain sniffers also provide protocol-level analysis of data flowing through network, packet by packet. This information is typically beneficial for administrators to diagnose and fix network related problems.

Some organizations also collect information about network events at the firewalls. There are several categories of firewalls (packet filters, circuit level gateways, application level gateways and stateful multilayer inspection firewalls [21, 231]) that all collect firewall logs and use them to detect suspicious activity and alert human analysts.

Use of network connections/packets as source of intrusion detection data has several advantages:

- There are numerous network-specific attacks (e.g. large distributed denial-of-service attacks) that cannot be detected using audit information on the host but only using information about network infrastructure.
- TCP/IP standardization of network traffic facilitates collecting, formatting and analyzing information from heterogeneous audit trail formats that come from different portions of large and complex networks.
- Using the payload information (content of the packets) can be very informative in detection of attacks against hosts.

However, using network connections/packets also has several drawbacks:

- When an intrusion has been detected, it is not straightforward to identify an attacker, since there is no direct association between network connections/packets and the identity of the user who actually performed the attack.
- If the packets are encrypted, it is practically impossible to analyze the payload of the packets, as important information may be hidden from network sniffers. In addition, if the attack signatures are not sufficiently comprehensive, it is possible to evade detection by making the contents of the packet more complex [184].

Packet sniffers can be placed at the gateways between the protected system and the outside world, or on switches within the network. Which of these is the most appropriate location, it is not always clear. Placing sniffers on switches gives better audit information but at a higher cost, due to a larger number of switches in the network. Nevertheless, networks that use switches are commonly used since they are less vulnerable to sniffer attacks [42, 184].

Network packets are the source of information used by most of the recent commercial products [8, 47, 89, 159, 160, 210, 222, 238], as well as by many projects in the research community [61, 120, 142, 174, 188, 192, 216]. Other network-based systems such as Bro [174] have been developed as network data-acquisition tools, but not as tools to directly support intrusion-detection task.

4.2.2 Simple Network Management Protocol (SNMP) information

The Simple Network Management Protocol (SNMP) is the Internet standard operations and maintenance protocol that facilitates the exchange of management information between network devices. SNMP was designed to help network administrators to manage network performance, to find and solve network problems, and to minimize resources necessary for supporting network management.

An SNMP-managed network typically consists of three components: managed devices, agents, and one or more network management systems (NMSs). A managed device corresponds to any SNMP-compliant equipment that resides on a managed network, collects management information and sends this information to NMSs using SNMP. Examples of managed devices include routers, switches, hubs, workstations, printers, etc. An agent is typically a "*network-management software*" module that resides on a managed device. The agent gathers management information from managed devices and converts that information into a format that can be passed over the network using SNMP. Finally, an NMS monitors and controls managed devices, issues requests and returns responses from devices. Information collected from NMSs can serve as a useful audit source.

One of the earliest projects that used SNMPv1 Management Information Base (MIB) for Ethernet and TCP/IP was SECURENET [212]. The SECURENET project showed that the counters maintained in the SNMPv1 MIBs could be potentially interesting as an audit source for anomaly detection techniques. SNMPv2 and SNMPv3 have also been used for security and intrusion detection [100], but the failure of SNMPv2 has lowered the interest of the intrusion-detection community in these information sources.

4.3 Application log files

Application based IDSs monitor only specific applications such as database management systems, content management systems, accounting systems, etc. An application based IDS has access to types of information that network based or host based IDSs do not have. For example, by analyzing application log files, application based IDSs can detect many types of computer attacks, suspicious activities that can be difficult to detect using host based or network based IDSs. In addition, they can be used to trace down unauthorized activities from individual users or to analyze encrypted data by employing application-based encryption/decryption services [20]. As application servers have recently become increasingly popular, application log files are used more often as an information source for intrusion detection.

In general, there are two approaches to implement application based IDSs [20]. In the first approach, IDS monitors an application and analyzes its audit log files. This post analysis allows suspicious activities in the application to be observed easily, but only after they happen. In a second, more complex approach, application based IDS is integrated into the application itself. This integration allows IDS to analyze the data at the same time the application interprets it, to detect attacks in real-time making it possible to take an immediate action.

The operation of an application based IDS in general is not impacted by the total amount of network traffic unless most of the traffic is due to the application (e.g., at a large commercial vendor sites such as Amazon.com, application-based IDSs highly depend on the network traffic).

In general, application based IDSs offer several advantages:

- *Unencrypted information.* Unlike the analyzed data at the network level, the data at the application level is not encrypted, thus giving more information for intrusion analysis to application based IDSs.
- *Prediction performance.* Since an application based IDS focuses on monitoring operations specific to the application, it is easier to define the normal and the anomalous behavior. There are certain types of information (e.g. query logs from database applications) that are available only to application based IDSs but not visible to the operating system. As a result, application based IDSs can detect intrusions that are not detectable by host-based IDSs. This results in a lower false alarm rate, as well as in higher detection rate.
- *Complete sessions.* Unlike network monitoring where network connection may be fragmented during recording, the application typically records complete transaction, and there is no inconsistency involved in the reconstruction of session records.
- *Prevention.* When an application based IDS is embedded in the application module itself, it can stop the intruder from proceeding with the attack by denying malicious operations.

However, application based IDSs have also certain limitations:

- *Performance penalty.* When an application based IDS is not a part of an application itself, it usually needs to be installed on the same host as the application. In such scenario, this installation could result in a decrease in the system performance.
- *Larger system overhead.* Since the application based IDSs have to be installed on every individual host machine, and the organization may have numerous hosts, there is a larger administration overhead.
- *Non-detectable attacks below the application layer.* Although analyzing the data at the application level allows application-based IDSs access to

encrypted information, they are not able to detect attacks that target protocols below the application layer.

- *Specific development.* Every application based IDS has to be developed for a specific application, since there is no general application-based IDS.

4.4 Wireless networks

Wireless network systems have become increasingly popular recently, mainly due to the ease of their installation and maintenance. However, this convenience comes at a price, since wireless networks pose a serious security risk. There are numerous, potentially devastating threats that have emerged in wireless networks that are more difficult to detect due to the following reasons [3, 88, 126]:

- Physical layer in wireless networks is essentially a broadcast medium and therefore less secure than in fixed computer networks. For example, an attacker that enters the wireless network, bypasses existing security mechanisms and can easily sniff sensitive and confidential information. In addition, the attacker also has access to all the ports that are regularly available only to the people within the network. In wired networks, attempts to access these ports from outside world through Internet are stopped at the firewalls. Finally, the attacker can also excessively load network resources thus causing denial of service to regular users.
- There are no specific traffic concentration points (e.g. routers) where packets can be monitored, so each mobile node needs to run an intrusion detection system.
- Separation between normal and anomalous traffic is often not clear in wireless ad-hoc networks, since the difference between compromised or false node and the node that is temporarily out of synchronization due to volatile physical movement can be hard to observe.

There are currently only a few commercial wireless IDS solutions [3, 88] in the market that try to detect a wide range of known attacks as well as identify abnormal network activities and policy violations for wireless networks. For Linux operating system, Lin et al have developed a homegrown wireless IDS [126] along with a freely available software. Other open source solutions include Snort-Wireless [208] and WIDZ [239].

4.5 Alerts from intrusion detection systems

Due to increase in a traffic volume, current commercial IDSs usually tend to produce a very large number of alarms [185]. These alarms are raised both for actual intrusions (attacks), but very often for regular behavior, thus increasing false alarm rate and overwhelming security administrator. In addition, a large distributed DoS or scanning attack may trigger multiple alarms since many network connections are involved in such attacks. This further increases the number of alarms that security analysts have to analyze. In order to decrease this number, the threshold for detecting intrusions is raised, but this can reduce the overall detection rate.

Due to these reasons, a number of researchers have attempted to develop a new generation of intrusion-detection systems that correlate information from several, "lower-level" IDSs to identify intrusions [50, 101, 168, 177, 186, 225, 229]. These IDSs employ different correlation and data-mining techniques in order to reduce both false alarm rate and the burden on the security analyst. In addition, some of these IDSs can typically provide security analysts with a summarized view of detected anomalous activities. Examples of such IDSs include distributed intrusion detection system (DIDS) [217] that correlates user identification by using information from sensors and GrIDS [225] that measures the traffic on hosts and network links and then correlates information from sensors on multiple networks. In general, there are three basic groups of alert correlation methods:

- *Methods based on similarities between alert attributes (features)* [101, 229] compare the degree to which alerts have similar features (e.g. source IP address, destination IP address, ports), and then correlate alerts with a high degree of feature similarity.
- *Correlation methods based on known attack scenarios* [50, 186, 225] utilize the fact that intrusions often require several actions to take place in order to succeed (e.g. to carry out a DoS attack on the DNS server, the attacker could first do an nslookup, ping, and scan port 139, and then a winnuke (sends out-of-Band data to an IP address of a windows machine)). Every attack scenario has corresponding steps required for the success of the attack. Low-level alerts from IDS(s) are compared against the predefined attack scenario before the alerts can be correlated. Major drawbacks of this method are (i) it requires that human users specify the attack scenarios and (ii) it is limited to detection of known attacks.
- *Correlation methods based on preconditions and consequences of individual attacks* [168] work at a higher level then correlation based on feature similarities, but at a lower level then correlation based on known scenarios. Preconditions are defined as conditions that must exist for the attack to occur, and the consequences of the attack are defined as conditions that may exist after a specific attack has occurred.

5. ANALYSIS STRATEGY: MISUSE DETECTION VS. ANOMALY DETECTION

There are two primary approaches for analyzing events to detect attacks; namely misuse detection and anomaly detection. Misuse detection is based on extensive knowledge of known attacks and system vulnerabilities provided by human experts. The misuse detection approaches look for hackers that attempt to perform these attacks and/or to exploit known vulnerabilities. Although the misuse detection can be very accurate in detecting known attacks, misuse detection approaches cannot detect unknown and emerging cyber threats.

Anomaly detection, on the other hand, is based on the analysis of profiles that represent normal behavior of users, hosts, or network connections. Anomaly detectors characterize normal "legitimate" computer activity using different techniques and then use a variety of measures to detect deviations from defined normal behavior as potential anomaly. The major benefit of anomaly detection algorithms is their ability to potentially recognize unforeseen attacks. However, the major limitation is potentially high false alarm rate. Note that deviations detected by anomaly detection algorithms may not necessarily represent actual attacks, as they may be new or unusual, but still legitimate, network behavior.

Many contemporary IDSs integrate both approaches to benefit from their respective advantages [164, 167, 200, 207].

5.1 Misuse Detection

Misuse detection is the most common approach used in the current generation of commercial intrusion detection systems (IDSs). The misuse detection approaches can be classified into the following four main categories: (i) signature-based methods, (ii) rule-based techniques, (iii) methods based on state-transition analysis, and (iv) data mining based techniques.

5.1.1 Signature-based techniques

Signature-based IDSs operate analogously to virus scanners, i.e. by searching a database of signatures for a known identity – or signature – for each specific intrusion event. In signature-based IDSs, monitored events are matched against a database of attack signatures to detect intrusions. Signature-based IDSs are unable to detect unknown and emerging attacks since signature database has to be manually revised for each new type of intrusion that is discovered. In addition, once a new attack is discovered and its signature is developed, often there is a substantial latency in its deployment across networks [130]. The most well known signature-based IDSs include SNORT [207], Network Flight Recorder [167], NetRanger [47], RealSecure [89], Computer Misuse Detection System (CMDS™) [230], NetProwler [14], Haystack [204] and MuSig (Misuse Signatures) [127].

SNORT [207] is a widely used open source signature-based network IDS, which is used for performing real-time traffic logging and analysis over IP networks. Currently, SNORT has an extensive database of over a thousand attack signatures. There are three main modes in which SNORT can be configured; namely sniffer, packet logger, and network IDS. In the sniffer mode, SNORT monitors the network packets and continuously displays them on the console. Packet logger mode is used to store (log) the packets to the disk. In the network intrusion detection mode, the system analyzes network traffic for matches against a database of user defined rules and performs one of five corresponding actions:

- *Alert* – raise an alarm using the selected alert method and then log the packet;
- *Log* – log the analyzed packet;
- *Pass* – ignore the analyzed packet;
- *Activate* – generate an alert and then turn on another *dynamic* rule;
- *Dynamic* – stay inactive until turned on by an *activate* rule.

Network Flight Recorder (NFR) is a network-based IDS that also creates alerts based on rules. These rules, called "backends" in NFR terminology, contain filters (hard-coded signatures) written to trigger in response to different computer attacks. NFR includes a complete programming language, called N, designed for packet analysis and creating filters.

NetRanger [47], an IDS developed at Cisco, was introduced to intrusion detection community in November 1998. Over the years NetRanger grew into a more complex Cisco IDS [46] that provides complete intrusion protection and is a component of a SAFE BluePrint Cisco security system. NetRanger is composed of three major components: sensors, director and post office. Sensors are network appliances that analyze the network traffic using a rule-based engine, which distills large volumes of network traffic into meaningful security events, which are then forwarded to a Director. Directors are responsible for the management of security across a distributed network of sensors and can be structured hierarchically to manage large networks. Finally, the post office provides communication between NetRanger services and hosts.

RealSecure, is an earlier version of the Proventia system developed at Internet Security Systems [182]. While Real Secure was principally a signature-based IDS composed of three modules: network engines, system agents, and managers, Proventia provides a more complete security solution including: inspection firewall, antivirus protection, intrusion detection and prevention, anti-spam filters and application protection.

CMDS [230] was a predecessor of Intrusion SecureHost [90], which represents a host-based IDS that monitors and protects applications at the kernel level of operating system by building a profile of the application's normal behavior based on the "*code paths of a running program*". NetProwler [14] is another host basd IDS that is based on "Stateful Dynamic Signature Inspection" virtual processor proposed by Anxent, which was acquired by Symantec recently. Today, NetProwler is a part of Symantec Intruder Alert IDS [220]. NetProwler collects various types of information "sniffed" from the network and then integrates them into more complex events that are matched against predefined signatures in real time. In addition, the system can install novel signatures without stopping the intrusion detection process.

The Haystack prototype [204] was one of the first signature based IDSs developed for the task of intrusion detection in a multi-user Air Force computer system. Haystack employs both misuse detection and anomaly detection strategy for detecting intrusions. The misuse detection module identifies intrusions according to behavioral constraints (rules) imposed by official security policies. On the other hand, the anomaly detection module is based on building profiles of users' behavior in the past and on constructing generic user group models that describe generic acceptable behavior for a particular group of users.

Adaptable real-time misuse detection system (ARMD) [127], developed at George Mason University, provides a high-level language for abstract misuse signatures, called MuSigs, and a mechanism to translate MuSigs into a monitoring program. With the notion of abstract events, the high-level language specifies a MuSig as a pattern over a sequence of abstract events, which is described as conditions that the abstract event attributes must satisfy. In addition, on the basis of MuSigs, the available audit trail, and the strategy costs, ARMD uses a strategy generator to automatically generate monitoring strategies to govern the misuse detection process.

Kumar and Spafford proposed a generalized framework for matching intrusion signatures based on Colored Petri Nets [113]. In this approach, every signature of an attack is represented as a Petri net, and start states and final state are used to perform signature matching.

5.1.2 Rule-based systems

Rule-based systems use a set of "if-then" implication rules to characterize computer attacks. At the early stage of intrusion detection era, rule based languages represented one of the regular methods for describing the expert's knowledge that is collected about numerous attacks and vulnerabilities. In rule-based IDSs, security events are usually monitored and then converted into the facts and rules that are later used by an inference engine to draw conclusions. Examples of such rule-based IDSs include Shadow [170], IDES [56, 95, 138, 139], NIDX [19], ComputerWatch [58], P-BEST [129], ISOA [241, 242] and AutoGuard that uses case-based reasoning [66, 67].

IDES [138] is a rule-based expert system trained to detect known intrusion scenarios, known system vulnerabilities, and site-specific security policies. IDES can also detect (i) outside attacks from unauthorized users; (ii) internal attacks from authorized users who masquerade as other users and (iii) attacks from authorized users who abuse their privileges by avoiding access controls. NIDX [19] extends the IDES model by including system dependent knowledge such as a description of file systems, and rules regarding system policies. It integrates (i) information obtained from the target computer system, (ii) user profiles built through history and (iii) intrusion detection heuristics into rules that are used to detect violations from the audit trail on the target system.

The ComputerWatch [58] data reduction tool was developed as an expert system IDS by the Secure Systems Department at AT&T. Computer Watch employs the host audit trail data to summarize system security activities and provides mechanisms for further investigation of suspicious security events by security analysts. The tool checks users' actions according to a set of rules that describe proper usage policy, and flags any suspicious action that does not match the acceptable patterns.

Production Based Expert System Toolset (P-BEST) [129] is a rule-based, forward-chaining expert system developed at SRI, and used in the EMERALD IDS [179]. The system was first deployed in the MIDAS ID system at the National Computer Security Center, and then used as the rule-based inference engine of NIDES, which is an IDES successor. P-BEST is a programmable expert system shell that consists of the definition of several fact types, and a set of inference rules on these facts. Inference rules are composed of two parts. The first part is a guard, which tests the existence of facts satisfying logical expressions; and the second part is composed of actions upon the fact base (adding, removing, modifying facts) and of calls to external functions.

ISOA (Information Security Officer's Assistant) [241, 242] is a real time IDS for monitoring security relevant behavior in computer networks. ISOA serves as the central point for real-time collection and analysis of audit information. It has two components; i.e. statistical analysis module and an expert system. These components cooperate in the automated analysis of various "concern levels". If a recognized set of indicators are matched, concern levels increase and the IDS starts to analyze the growing classes of audit events in more details to flag suspicious users or hosts.

5.1.3 State transition analysis

Intrusion detection using state transition analysis requires the construction of a finite state machine, in which states correspond to different IDS states, and transitions characterize certain events that cause IDS states to change. IDS states correspond to different states of the network protocol stacks or to the integrity and validity of current running processes or certain files. Every time when the automation reaches a state that is flagged as a security threat, the intrusion is reported as a sign of malicious attacker activity. This is the technique first proposed in USTAT (Unix State Transition Analysis Tool) [86, 178] and later in NetSTAT (Network-based State Transition Analysis Tool) [232].

USTAT, developed at UC Santa Barbara, is a real-time state transition analysis tool developed for the Unix system and based on STAT (State Transition Analysis Tool) [178]. STAT introduced the idea of representing computer attacks with high level descriptions and providing an expert system model to detect compromises. In STAT, attack scenarios are represented as states that describe security status of the system, and intrusions are detected by modeling the transition between states. The computer initially exists in a secure state, but as a result of a number of intrusions it may end up in a compromised target state. USTAT uses the C2 security audit trail data produced by the computer as the source of information about the system's state transitions. It records only those critical actions that have visible effect on the system state and must happen in order to successfully complete the penetration.

NetSTAT is a real-time network-based IDS that employs state transition analysis techniques from the STAT approach, for detecting intrusions that occur in a networked environment. The networked environment is represented by hypergraphs, where network interfaces are modeled as nodes, and hosts are modeled as edges of the hypergraph. By using state transition analysis for the states of network attacks, it is possible to automatically determine which network events have to be monitored in order to support intrusion analysis.

5.1.4 Data mining based techniques

In data mining methods for misuse detection, each instance in a data set is labeled as 'normal' or 'intrusive' and a learning algorithm is trained over the labeled data. These techniques are able to automatically retrain intrusion detection models on different input data that include new types of attacks, as long as they have been labeled appropriately. Research in misuse detection has focused mainly on classification of network intrusions using various standard data mining algorithms [16, 74, 121, 140, 202], rare class predictive models [40, 98, 99], cost sensitive modeling [99] and association rules [16, 122, 143]. Unlike signature-based intrusion detection systems, models of misuse are created automatically, and can be more sophisticated and precise than manually created signatures. The advantage of data mining based misuse detection techniques over signature-based intrusion detection systems is their high degree of accuracy in detecting known attacks and their variations.

MADAM ID [120, 122] at Columbia University was one of the first project that applied data mining techniques to the intrusion detection problem. Association rules and frequent episodes were extracted from network connection records to obtain additional features for data mining algorithms. Three groups of features are constructed, namely: content-based features that describe intrinsic characteristics of a network connection (e.g. number of packets, acknowledgments, data bytes from source to destination), time-based traffic features that compute the number of connections in some recent time interval (e.g. last few seconds) and connection based features that compute the number of connections from a specific source to a specific destination in the last *N* connections (e.g. N = 1000). In addition to the standard features that were available directly from the network traffic (e.g. duration, start time, service), these constructed features were also used by the RIPPER algorithm to learn intrusion detection rules from DARPA 1998 data set [132, 133].

Other classification algorithms for the intrusion detection problem include decision trees [24, 202], modified nearest neighbor algorithms [246], fuzzy association rules [26, 72, 140], neural networks [30, 51, 131, 247], naïve Bayes classifiers [196], genetic algorithms [26, 145], genetic programming [158], support vector machines [65, 156], and adaptive regression splines [157]. Most of these approaches attempt to directly apply specified standard techniques to some of publicly available intrusion detection data sets [132, 133], assuming that the labels for normal and intrusive behavior are already known.

Computer intrusions, however, are much rarer than normal behavior, and in such scenarios standard classification algorithms do not perform well. Thus, some researchers have developed specially designed algorithms for handling rare classes and applied them to the problem of intrusion detection [40, 98, 99].

Finally, association patterns, often expressed in the form of frequent itemsets or association rules, have also been found to be valuable for analyzing network traffic data [16, 121, 143]. In [121], association patterns generated at different times were used to study significant changes in the network traffic characteristics at different periods of time, while in [16, 121, 143] they were used to construct a profile of the normal network traffic behavior for anomaly detection systems.

5.2 Anomaly Detection

Increase in the number of computer attacks, in their severity and complexity has raised substantial interest in anomaly detection algorithms due to their potential for recognizing unforeseen and emerging cyber activities. There are many anomaly detection algorithms proposed in the literature that differ according to the information used for analysis and according to techniques that are employed to detect deviations from normal behavior. In this section, we provide classification of anomaly detection techniques based on employed techniques into the following five groups: (i) statistical methods; (ii) rule based methods; (iii) distance based methods (iv) profiling methods and (v) model based approaches. Although anomaly detection algorithms are quite diverse in nature, and thus may fit into more than one proposed category, our classification attempts to find the most suitable category for all described anomaly detection algorithms.

5.2.1 Statistical methods

Statistical methods monitor the user or system behavior by measuring certain variables over time (e.g. login and logout time of each session). The basic models keep averages of these variables and detect whether thresholds are exceeded based on the standard deviation of the variable. More advanced statistical models also compare profiles of long-term and short-term user activities. These statistical models are used in host-based IDSs, network-based IDSs, as well as in application-based IDSs for detecting malicious viruses. Some of the first proposed anomaly detection algorithms were integrated in well known IDSs such as IDES [56, 95, 138, 139], NIDES [6], EMERALD [164, 179] and SPADE [214].

IDES [138], whose misuse detection module is explained in section 4.1.2., also has an anomaly detection module. This module characterizes normal user activity using an audit data and detects deviations from described normal user behavior. Each new audit record is processed as it enters the system, and verified against the known profile. To further distinguish unusual but authorized behavior, the prototype was extended to handle two sets of profiles for monitored subjects depending on whether the activity took place on “normal” or “suspicious” days. The security analyst defines whether working days are "normal" or not. The NIDES system [6] extends IDES by integrating results from misuse detection component with the results produced by the anomaly detection module. NIDES monitors ports and addresses and builds a statistical model of long term behavior over a period of hours or days, which is assumed to contain few or no attacks. If short-term behavior (seconds, or a few packets) differs significantly from normal, then an alarm is raised.

EMERALD [164, 179] has statistical profile-based anomaly detection module that tracks subject activity through one of four types of statistical variables: categorical, continuous, traffic intensity (e.g., volume over time), and event distribution (e.g., a meta-measure of other measures). The eBayes system [228] is a recently developed module that extends earlier anomaly detection component from the EMERALD system [164, 179] by encoding probabilistic models of normal, attack, and anomalous behavior modes with hypotheses. The eBayes system first collects basic variables of network sessions as well as derives new ones (e.g. maximum number of open connections to any unique host), and then applies probabilistic Bayesian inference to them in order to obtain a belief for the session over the states of hypotheses. For example, the session hypotheses in the eBayes TCP tree may correspond to both normal traffic modes (MAIL, FTP, etc.) and to attack scenario modes PORTSWEEP, SYNFLOOD, etc.). The eBayes builds a table of conditional probabilities for all the hypotheses and variables, which is adjusted every time the current observation is made. The eBayes has an option of detecting novel attacks by dynamically generating new hypothesis, which is obtained by adding a fake state of hypothesis and a new conditional probability table row initialized by a uniform distribution.

Similarly to eBayes, many anomaly detection techniques have been proposed recently to overcome limitations of earlier statistical anomaly detection algorithms. For example, SPADE [214] is a statistical based system, that is available as a plug-in for SNORT as a plug-in, and used for automatic detecting stealthy port scans. Unlike traditional scan detectors that look for X events in Y seconds, SPADE takes a fundamentally different approach and looks at the amount of information gained by probing. It has four different methods of calculating the likelihood of packets, of which most successful method measures the direct joint probability P(dest IP, dest Port) between destination IP address and destination port. SPADE examines TCP-SYN packets and maintains the count of packets observed on (destIP, destPort) tuples. When a new packet is observed, SPADE checks the probability of observing that packet on the (dest IP, dest Port) tuple. The lower the probability of the packet, the higher the anomaly score. However, in a real life system, SPADE gives a high false alarm rate, since all unseen (dest IP, dest Port) tuples are detected as attacks regardless whether or not they correspond to actual intrusions.

Another recently proposed statistical method employs statistical traffic modeling [29] for detecting novel attacks against networks. In this approach, a network activity model is used to detect large classes of Denial of Service and scanning attacks by monitoring the network traffic volume. By applying the KolmogorovSmirnov test on the DARPA dataset [132], it was demonstrated that, for example, normal telnet connections are statistically different from the attacks that use telnet connections.

Chi-square (χ^2) statistics have also been successfully used to detect anomalies both in host-based and network based intrusion detection. For host-based IDSs, Ye [245] proposed approach where activities on a host machine are captured through a stream of events and then characterized by the event type. For each event type, the profiles of audit events from normal behavior are defined, and then used to compute χ^2 as a measure of difference between the test audit event and the normal audit event, whereas large deviations are detected as anomalies. In network based IDS, the chi-square statistic has also been used [111] to differentiate the payload distribution (distribution of characters in the content of the network packets) in normal network packets and anomalous ones.

Some researchers have used outlier detection algorithms for anomaly detection, since outliers are typically defined as data points that are very different from the rest of the data. The statistics community has studied the concept of outliers quite extensively [17]. In these techniques, the data points are modeled using a stochastic distribution, and points are determined to be outliers depending on their relationship with this model. For example, SmartSifter [244] uses a probabilistic model as a representation of underlying mechanism of data generation, and scores each data example by measuring how large the model has changed after the learning. Smart sifter extension [243] gives positive labels to higher scored data and negative to the lower scored data, and then constructs an outlier filtering rule by applying supervised learning. Eskin's approach [64] computes the likelihood of data distribution $L_t(D)$ at some specific time interval t, removes a data example at the interval t-1 and measures the likelihood of data distribution without removed data example $L_{t-1}(D)$. The probability that removed data example is an outlier is proportional to the difference between the new likelihood $L_{t-1}(D)$ and the original one $L_t(D)$. Information theoretic measures such as entropy, conditional entropy, relative conditional entropy, information gain, and information cost [123] were also proposed for anomaly detection task. These measures were used to characterize the characteristics of an audit data set by measuring their regularity, and to build appropriate anomaly detection models according to these regularity measures. The higher regularity of audit data, the better the anomaly detection module is.

Statistic based anomaly detection techniques have also been used in detecting malicious viruses through e-mail messages. For example, the MET (Malicious Email Tracking) [22] system keeps track of email attachments as they are exchanged between users through a set of collaborating email servers that forward a subset of their data to a central data warehouse and correlation server. Only attachments with a high frequency of appearance are deemed suspicious, while the email exchange patterns among users are used to create models of normal behavior. MET system contains MET server and MET clients. MET server is used to collect data on malicious activity, store them in a database, and calculate derived statistics, while MET clients analyze email attachments across all mail domains and then detect email-based attacks.

5.2.2 Distance based methods

Most statistical approaches have limitation when detecting outliers in higher dimensional spaces, since it becomes increasingly difficult and inaccurate to estimate the multidimensional distributions of the data points [2]. Distance based approaches attempt to overcome limitations of statistical outlier detection approaches and they detect outliers by computing distances among points. Several distance based outlier detection algorithms have been recently proposed for detecting anomalies in network traffic [117]. These techniques are based on computing the full dimensional distances of points from one another [107, 187] using all the available features, and on computing the densities of local neighborhoods [25, 117]. MINDS (Minnesota Intrusion Detection System) [61] uses net-flow data to extract useful set of features to be used in anomaly detection. MINDS anomaly detection module employs an outlier detection algorithm to assign an anomaly score to each network connection. A human analyst then has to look at only the most anomalous connections to determine if they are actual attacks or other interesting behavior. MINDS anomaly detection module is used at the University of Minnesota and is also incorporated into the Interrogator architecture at the ARL Center for Intrusion Monitoring and Protection (CIMP), where network data from multiple sensors are collected and analyzed by human analysts to detect intrusions and attacks. Experiments on live network traffic at the University of Minnesota and at the ARL-CIMP have shown that MINDS is able to routinely detect various suspicious behavior (e.g. policy violations), worms, as well as various scanning activities.

In addition, in several clustering based techniques (fixed-width and canopy clustering [65]), network intrusions in DARPA 1998 evaluation data sets have been detected as small clusters when compared to the large ones that corresponded to the normal behavior.

In another interesting approach [68], artificial anomalies in the network intrusion detection data are generated around the edges of the sparsely populated data regions, thus forcing the learning algorithm to discover the specific boundaries that distinguish these regions from the rest of the data.

5.2.3 Rule based systems

Rule based systems used in anomaly detection characterize normal behavior of users, networks and/or computer systems by a set of rules. Examples of rule based IDSs include ComputerWatch [58] and Wisdom & Sense [124, 125].

ComputerWatch system [58] employs a typical rule based system that summarizes "normal" security events and then detects anomalous behavior as deviations from them. The rule system creates rules to describe proper usage policy, to check users' actions according to these rules, and to flag any action that does not match the described rule patterns. Wisdom & Sense [124, 125] employs historic audit data to produce a set of rules describing normal behavior, forming the "wisdom" of the title. These rules are then fed to an expert system that evaluates recent audit data for violations of the rules, and alerts the security analyst when the rules indicate ("sense") anomalous behavior.

Recently, Valdes [227] proposed an unsupervised technique that does not require attack free training data and detects novel scans through pattern-based anomaly detection. The model assigns network connections into one of a number of modes discovered by competitive learning. The technique is applied to port patterns in TCP sessions in simulated and real network traffic.

5.2.4 Profiling methods

In profiling methods, profiles of normal behavior are built for different types of network traffic, users, programs etc., and deviations from them are considered as intrusions. Profiling methods vary greatly ranging from different data mining techniques to various heuristic-based approaches. In this section, we provide an overview of several distinguished profiling methods for anomaly detection.

ADAM (Audit Data and Mining) [16] is a hybrid anomaly detector trained on both attack-free traffic and traffic with labeled attacks. The system uses a combination of association rule mining and classification to discover attacks in *tcpdump* data. One of the advantages of ADAM is its ability to detect novel attacks, without depending on attack training data, through a novel application of the pseudo-Bayes estimator [16]. Recently reported IDDM system [1] represents an off-line IDS, where the intrusions are detected only when sufficient amounts of data are collected and analyzed. The IDDM system describes profiles of network data at different times, identifies any large deviations between these data descriptions and produces alarms in such cases.

Human immune system has gained a lot of attention among researchers in intrusion detection community, especially when analyzing attacks at the host level [73, 119, 209]. These techniques first collect data patterns representing the appropriate behavior of the service and extract a reference table containing all the known good sequences of system calls. These patterns are then used for live monitoring to check whether the sequences generated are listed in the table or not. If they are not listed, an alarm is generated. Wespi [237] also proposed a novel technique for modeling process behavior by building a table of variable length patterns, which is based on the Teiresias algorithm. Experimental results show that the variable length pattern model is significantly better than a fixed length approach, both in reducing the number of patterns to describe the normal process behavior and in achieving better detection rates. Although the immune system approach is interesting and intuitively appealing, so far it has proven to be difficult to apply [60].

The temporal sequence learning [116] has been shown successful in profiling Unix user command line data, where user shell commands are used to build user profiles for activities during an intrusion and for activities during normal use. By comparing these profiles, it is possible to detect new types of anomalous user behavior.

Association pattern analysis has been shown to be beneficial in constructing a profile of normal network traffic behavior [61, 118, 143]. For example, Manganaris [143] used association rules to characterize the normal stream of IDS alerts from a sensor and later to distinguish between false alarms and real ones. On the other hand, MINDS [61] uses association patterns to provide high-level summary of network connections that are ranked highly anomalous in the anomaly detection module. These summaries allow a human analyst to examine a large number of anomalous connections quickly and to provide templates from which signatures of novel attacks can be built for augmenting the database of signature-based intrusion detection systems.

PHAD (packet header anomaly detection) [142] monitors network packet headers and builds profiles for 33 different fields from these headers by observing attack free traffic and building contiguous clusters for the values observed for each field. The number of clusters is pre-specified and if a new value that is observed does not fit into any of the clusters, it is treated as a new cluster and the closest two clusters are merged. The number of updates, *r*, is maintained for each field as well as the number of observations, *n*. When a new packet is being tested for anomaly, the values of all fields are checked to see if they fit into the clusters formed in the training phase. If the values for some fields do not fit into any clusters, then each of them contributes to the anomaly score value of the packet proportional to the *n/r* ratio for the field. ALAD (application layer anomaly detection) [142] uses the same method for calculating the anomaly scores as PHAD, but it monitors TCP data and builds TCP streams when the destination port is smaller than 1024. It constructs five features from these streams as opposed to 33 fields used in PHAD.

ADMIT (Anomaly-based Data Mining for InTrusions) [201] attempts to discriminate between masqueraders and true users on computer terminals. This task is performed by augmenting conventional password authentication measures and by continuously running a terminal-resident IDS program, which monitors the terminal usage by each user, creates an appropriate profile and verifies user data against it.

Call stack information [71] was also effectively used to detect various exploits on computer systems. The anomaly detection approach, called *VtPath*, first extracts return addresses information from the call stack and generates "abstract execution paths" between two execution points in the program. These "abstract execution paths" are then compared to the "abstract execution paths" learned during normal runs of the program.

Finally, there have also been several recently proposed commercial products that use profiling based anomaly detection techniques. For example, Antura from System Detection [222] use data mining based user profiling, while Mazu Profiler form Mazu Networks [147] and Peakflow X from Arbor networks [8] use rate-based and connection profiling anomaly detection schemes.

5.2.5 Model based approaches

Many researchers have used different types of models to characterize the normal behavior of the monitored system. In the model-based approaches, anomalies are detected as deviations for the model that represents the normal behavior.

Very often, researchers have used data mining based predictive models such as replicator neural networks [79] or unsupervised support vector machines [65, 117]. Replicator four-layer feed-forward neural network (RNN) [79] have the same number of input and output nodes. During the training phase, RNNs reconstruct input variables at the output layer, and then use the reconstruction error of individual data points as a measure of outlyingness. Unsupervised support vector machines [65, 117] attempt to separate the entire training data set from the origin, i.e. to find a small region where most of the data lies and label data points in this region as a normal behavior. In the test phase they detect deviations from learned models as potential intrusions. In addition, standard neural networks (NN) were also used in intrusion detection problems to learn a normal profile. For example NNs were often used to model the normal behavior of individual users [193], to build profiles of software behavior [74] or to profile network packets and queue statistics [122].

User Intention Identification [213] is a technique developed within the SECURENET project [212]. The goal of this technique is to model the normal behavior of users using a set of high-level tasks they have to perform on the system. These tasks are then refined into actions, which in turn are related to the audit events observed on the system. The analyzer keeps a set of tasks that each user can perform. Whenever an action occurs that does not fit the task pattern, an alarm is issued. User intention identification was also successfully used in several recently proposed approaches [43, 44].

Wagner [234] proposed to statically generate a non-deterministic finite automaton (NDFA) or a non deterministic pushdown automaton (NDPDA) from the global control flow graph of the program. The approach first computes a model of expected application behavior, built statically from program source code, then monitors program execution online at run time, and finally checks its system call trace for compliance to the model.

Specification based intrusion detection techniques have been recently proposed to produce a low rate of false alarms [199], but they have not been as effective as anomaly detection in detecting novel attacks. Hence, specification based anomaly detection [199] was designed to mitigate the weaknesses of both specification based IDSs and anomaly detection techniques and complement their strengths. The approach begins with state-machine specifications of network protocols, and augments these state machines with information about statistics that need to be maintained to detect anomalies.

Finally, anomaly detection has also been used in embedded systems [146], where Markov models were employed to determine whether the states (events) in a sequential data streams, taken from a monitored process, are normal or anomalous. It computes the probabilities of transitions between events in a training set, and uses these probabilities to assess the transitions between events in a test set.

6. TIME ASPECTS

When considering time aspects of IDSs, we distinguish two main groups: real-time (on-line) IDSs and off-line IDSs. Real-time (on-line) IDSs attempt to detect intrusions in real-time or near real-time. They operate on continuous data streams from information sources and analyze the data while the sessions are in progress (e.g. network sessions for network intrusion detection, login sessions for host based intrusion detection). Real-time IDSs should raise an alarm as soon as an attack is detected, so that action that affects the progress of the detected attack can be taken. Most commercial IDSs claim continuous processing capability [8, 147].

Off-line IDSs perform post-analysis of audit data. This method of audit data analysis is common among security analysts who often examine network behavior, as well as behavior of different attackers, in an off-line (batch) mode. Many early host-based IDSs used this timing scheme, since they used operating system audit trails that were recorded as files [77, 155].

Off-line analysis is also often performed using static tools that analyze the snapshot of the environment (e.g. host vs. network environment), look for vulnerabilities and configuration errors and assess the security level of the current environment configuration. Examples of these tools include COPS [69] and Tiger [194] for host environments, and Satan [70] and CyberCop Scanner [163, 197] for networks. Virus detectors belong to static tools too and they scan the disks searching for patterns matching known viruses. Although static tools are very popular and broadly used by system administrators, they are typically not sufficient to ensure high security [55].

Static tools can be also specifically designed for active investigation of vulnerabilities over the Internet. For example, Tripwire [106] or ATP [233] can be used to monitor a designated set of files and to detect computer intrusions that exploited older vulnerable applications. These intrusions should also be identified and reported to the system administrator as potential security holes using other tools like COPS [69] or Tiger [194].

7. ARCHITECTURE

There are two principal architectures that are used in IDSs, namely centralized and distributed IDSs. Most IDSs employ centralized architecture and detect intrusions that occur in a single monitored system. However, there is a recent increasing trend towards distributed and coordinated attacks, where multiple machines are involved, either as attackers (e.g. distributed denial-of-service) or as victims (e.g. large volume worms). Analysis that uses data from a single site and that is often employed by many existing intrusion detection schemes is often unable to detect such attacks. To effectively combat them, there is a need for distributed IDS and cooperation among security analysts across multiple network sites.

Unlike a centralized IDS, where the analysis of data is performed on a fixed number of locations (independent of how many hosts are being monitored), in a distributed IDS the analysis of data is performed on a number of locations that is proportional to the number of hosts that are being monitored [211]. An excellent comparison of centralized and distributed IDSs, with their advantages and drawbacks, is provided in a paper by Spafford and Zamboni [211]. Despite several drawbacks of distributed IDSs, many commercial vendors have realized the need for detecting coordinated cyber attacks from distributed locations, and adapted their systems to address these challenges [9, 162].

Starting from the first proposed distributed IDS [205], the most typical architectures of distributed IDSs assume employment of intelligent agents. There are several advantages of using mobile agent based intrusion detection systems over other approaches for distributed intrusion detection [94]. First, agents are independently running entities and can be added, removed and reconfigured without altering other components, and without restarting local IDSs. Second, agents can be tested on their own before introducing them into a more complex environment. Finally, agents can exchange information to derive more complex results than any one of them may be able to obtain on their own. Although IDSs based on mobile agents are still in their infancy and fully implemented systems are still emerging, there are many agent-based distributed IDSs [39, 109]. The typical examples include DIDS [59], AAFID [211], Argus [203], IDA [10], Micael [53].

DIDS [59] and distributed autonomous-agent NID [18] use a similar architecture that consists of a central analysis server and multiple IDS agents that communicate with each other. AAFID (autonomous agents for intrusion detection) [211] has a hierarchical design with three levels. At the lowest level, agents perform host security monitoring and data analysis. The information gathered by agents is forwarded to transceivers that distribute the information either to other agents or monitors, and control and configure agents at the second level. At the highest level, each monitor collects data from transceivers and evaluates their input. Intelligent agents in [82] employ classifier algorithms and travel among collection points, referred to as data cleaners, and uncover suspicious activities. The architecture is hierarchical, with a data warehouse at the root, data cleaners at the leaves, and classifier agents in between. A classifier agent specializes in a specific category of intrusion and is capable of collaborating with agents of another category to determine the severity level of an activity deemed suspicious. Moving the computational analysis to each collection point avoids the costly movement of information to an aggregation unit. Argus [203] employs a similar architecture with low-level agents that serve as data cleansers, and data mining agents that generate not only rules for matching a normal profile but also generate feedback for knowledge-based components. These rules can be used then to update the rule database of the NFR knowledge component [167]. Bayesian multiple hypothesis tracking was also used to more effectively analyze information provided by existing IDSs from multiple networks [28]. Hypotheses that explain the measured intrusion events are generated and stored, and then evaluated against the understanding of the sensor behavior in order to determine the likelihood of the hypotheses. The hypothesis with the greatest likelihood is assumed correct, while other hypotheses are treated as intrusions.

The Intrusion Detection Agent (IDA) system [10] is a multi-host based IDS that relies on mobile agents to trace intruders among the various hosts involved in an intrusion. IDA watches specific events that are related to various intrusions. These events are called "Marks Left by Suspected Intruder" (MLSI). If a specific MLSI is identified, IDA collects all the information related to this MLSI, analyzes this information and determines whether the MLSI is related to a real attack or not. The IDA system has a hierarchical tree structure, in which the central manager is placed at the root of the tree, while numerous agents are located at the leaves.

Micael [53] is a distributed IDS that uses autonomous mobile intelligent agents able to make various decisions in the process of intrusion detection (e.g. investigating intrusions and initiating countermeasures against them). The Micael architecture contains the following agents: (i) *headquarters*, i.e. specialized centralized agents that are responsible for creating other agents and maintaining their executable codes. They receive information about potential intrusions from *sentinel* agents and can create new *detachment* agents that will be sent to hosts when needed; (ii) *sentinels*, i.e. immobile agents that collect data about the activities on the host machines and inform *headquarter* agents about detected anomalies; and (iii) *detachments*, i.e. mobile agents that are used to face possible intrusions (hazards) by starting a detailed analysis of log files.

Applying intrusion detection techniques on a system-wide basis allows the system to be protected against general misuse, but may require significant resources. By optimizing the placement and configuration of these tools, it is possible to offer both increased protection for sensitive systems, and more context-sensitive detection, at the cost of general protection. For example, distributed IDS deployment often concentrates monitors in high-risk areas, such as network ingress points (e.g. adjacent to firewalls), or in the presence of valuable resources (such as network server farms) [148].

8. RESPONSE

The response of IDSs to identified attacks may be either passive or active. In the most common scenario, IDSs have passive response and simply inform responsible personnel of an event, but no countermeasure is actively applied to thwart the attack. The most common method for such notifications is through pop-up windows or on-screen alerts or through recording alerts into a file. These alerts may vary from notification of alarms only to detailed information about computer attacks such as source IP address, target of the attack, specific port of interest, the tools used to perform the attack, the outcome of the attack, etc. Some products also offer remote notification through sending alarms or alerts to cellular phones and pagers carried by system security personnel. In addition, notification is often sent through e-mail messages, but this may be unsafe, as attackers may monitor email and might even block the message. Certain IDSs (e.g. Cisco IDS [46]) use SNMP traps and messages to report generated alarms to a network management system, where network operations personnel can investigate them. Passive response is often used for off-line analysis.

Alternatively, IDSs can also provide an active response to critical events, such as "patching" a system vulnerability, logging off a user, re-configuring routers and firewalls, or disconnecting a port.

Given the speed and frequency at which attacks can occur, an ideal IDS would automatically respond to computer attacks at machine speed without requiring any operator intervention. However, this is an unrealistic expectation, largely due to the difficulty in eliminating false alarms. Nevertheless, IDS products can still provide a variety of active response mechanisms that may be used at the discretion of the system administrator.

One of the most harmless, but often most productive, active responses is to collect additional information about a suspected attack and to perform damage control. This might involve increasing the sensitivity level of information sources (e.g., increasing the number of events logged by an operating system audit trail, or increasing the sensitivity of a network monitor that captures all packets). Such additional information collected can help resolve the detection of the attack (assisting the system in diagnosing whether an attack did or did not take place) thus allowing the IDS to gather information that can be used to support investigation of the attacker.

In more recent IDS tools, active responses that include countermeasure against the attacker have become increasingly popular. An example of such a tool with early countermeasure capability is NetProbe [192], which monitors a network for undesired connections and immediately terminates them. There are also other tools with similar capabilities, such as RealSecure [89], NetRanger [47], and WebStalker [204] that have options to interrupt suspicious network connections that carry attacks, to block network traffic from the hosts that are originating attacks, or to reconfigure routers and firewalls.

9. CONCLUSIONS

Intrusion detection techniques have improved dramatically over time, especially in the past few years. Initially developed to automate tedious and difficult log parsing activity, IDSs have developed into sophisticated, real-time applications with the ability to have a detailed look at traffic and to sniff out malicious activity. They can handle high-speed networks and complex traffic, and deliver detailed insight – previously unavailable – into active threats against critical online information resources. IDS technology is developing rapidly and its near-term future is very promising. It is increasingly becoming an indispensable and integral component of any comprehensive enterprise security program, since it complements traditional security mechanisms.

This chapter provides an overview of the current state of the art of both computer attacks and intrusion detection techniques. The overview is based on presented taxonomies exemplified with the most illustrative paradigms. The taxonomy of computer attacks and intrusions provides the current status and trends in techniques that attackers employ today. The taxonomy of IDSs highlights their properties and provides an overview of the past and current developments. Although a variety of techniques have been developed for detecting different types of computer attacks in different computer systems, there are still a number of research issues concerning the prediction performance, efficiency and fault tolerance of IDSs that need to be addressed. Signature analysis, the most common strategy in the commercial domain until recently, is increasingly integrated with different anomaly detection and alert correlation techniques in order to detect emerging and coordinated computer attacks.

We hope this survey provides actionable information and advice on the topics, as well as serves to acquaint newcomers with the world of IDSs and computer attacks. The information provided herein is by no means complete and we recommend further reading to the interested reader.

ACKNOWLEDGEMENTS

This work was partially supported by Army High Performance Computing Research Center contract number DAAD19-01-2-0014, NSF grant IIS-0308264, and ARDA contract number F30602-03-C-0243. The content of the work does not necessarily reflect the position or policy of the government and no official endorsement should be inferred. Access to computing facilities was provided by the AHPCRC and the Minnesota Supercomputing Institute.

REFERENCES

[1] T. Abraham, IDDM: Intrusion Detection Using Data Mining Techniques, DSTO Electronics and Surveillance Research Laboratory, Department of Defense, Australia Technical Report DSTO-GD-0286, 2001.

[2] C.C. Aggarwal and P. Yu, Outlier Detection for High Dimensional Data, In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Santa BArbara, CA, May 2001.

[3] A. AirDefense, http://www.airdefense.net/products/index.html, 2004.

[4] J. Allen, A. Christie, W. Fithen, J. McHugh, J. Pickel, E. Stoner, J. Ellis, E. Hayes, J. Marella and B. Willke, State of the Practice of Intrusion Detection Technologies., Carnegie Mellon University, Pittsburgh, PA Technical Report CMU/SEI-99-TR-028, 1999.

[5] E. Amoroso, *Fundamentals of Computer Security Technology*, Prentice-Hall PTR, 1994.

[6] D. Anderson, T. Lunt, H. Javitz, A. Tamaru and A. Valdes, Detecting Unusual Program Behavior Using the Statistical Component of the Next-Generation Intrusion Detection Expert System (NIDES), Computer Science Laboratory, SRI International, Menlo Park, CA Technical Report SRI-CSL-95-06.

[7] J.P. Anderson, Computer Security Threat Monitoring and Surveillance, James P. Anderson Co., Box 42, Fort Washington, PA 19034 Technical Report Contract 79F296400, April 1980.

[8] Arbor Networks, Intelligent Network Management with Peakflow Traffic, http://www.arbornetworks.com/products_sp.php, 2003.

[9] ArcSight, Enterprise Security Management Software, http://www.arcsight.com/.

[10] M. Asaka, S. Okazawa, A. Taguchi and S. Goto, A Method of Tracing Intruders by Use of Mobile Agents, In *Proceedings of the 9th Annual Conference of the Internet Society (INET'99)*, San Jose, CA, June 1999.

[11] T. Aslam, A Taxonomy of Security Faults in the UNIX Operating System, Purdue University Master's thesis, August 1995.

[12] C.R. Attanasio, P.W. Markstein and R.J. Phillips, Penetrating an Operating System: A Study of VM/370 Integrity, *IBM System Journal*, vol. 15, 1, pp. 102-116, 1976.

[13] S. Axelsson, Intrusion Detection Systems: A Survey and Taxonomy, Dept. of Computer Engineering, Chalmers University Technical Report 99-15, March 2000.

[14] AXENT Technologies, Inc, NetProwler-Advanced Network Intrusion Detection, available online at:, http://www.axent.com/iti/netprowler/idtk_ds_word_1.html, 1999.

[15] R. Bace and P. Mell, NIST Special Publication on Intrusion Detection Systems, 2001.

[16] D. Barbara, N. Wu and S. Jajodia, Detecting Novel Network Intrusions Using Bayes Estimators, In *Proceedings of the First SIAM Conference on Data Mining*, Chicago, IL, April 2001.

[17] V. Barnett and T. Lewis, *Outliers in Statistical Data.* New York, NY, John Wiley and Sons, 1994.

[18] J. Barrus and N. Rowe, A Distributed Autonomous-Agent Network-Intrusion Detection And Response System, In *Proceedings of the Command and Control Research and Technology Symposium*, Monterey, CA, 577-586, June 1998.

[19] D.S. Bauer and M.E. Koblentz, NIDX - An Expert System For Real-Time, *Computer Networking Symposium,* 1988.

[20] T. Baving, Network vs. Application-Based Intrusion Detection, *Network and Internet Nettwork Security, Computer Science Honours*, 2003.

[21] S.M. Bellovin and W.R. Cheswick, Network Firewalls., *IEEE Communications Magazine*, vol. 32, 9, pp. 50-57, September 1994.

[22] M. Bhattacharyya, M. Schultz, E. Eskin, S. Hershkop and S. Stolfo, MET: An Experimental System for Malicious Email Tracking, In *Proceedings of the New Security Paradigms Workshop (NSPW)*, Hampton, VA, September 2002.

[23] M. Bishop, How Attackers Break Programs, and How To Write Programs More Securely, In *Proceedings of the 8th USENIX Security Symposium*, University of California, Davis, August 1999.

[24] E. Bloedorn, A. Christiansen, W. Hill, C. Skorupka, L. Talbot and J. Tivel, Data Mining for Network Intrusion Detection: How to Get Started, MITRE Technical Report, http://www.mitre.org/work/tech_papers/tech_papers_01/bloedorn_datamining, August 2001.

[25] M.M. Breunig, H.P. Kriegel, R.T. Ng and J. Sander, LOF: Identifying Density Based Local Outliers, *ACM SIGMOD Conference*, vol. Dallas, TX, May 2000.

[26] S. Bridges and R. Vaughn, Fuzzy Data Mining and Genetic Algorithms Applied to Intrusion Detection, In *Proceedings of the Twenty-third National Information Systems Security Conference*, Baltimore, MD, October 2000.

[27] H. Burch and B. Cheswick, Tracing Anonymous Packets to Their Approximate Source, In *Proceedings of the USENIX Large Installation Systems Administration Conference*, New Orleans, LA, 319-327, December 2000.

[28] D. Burroughs, L. Wilson and G. Cybenko, Analysis of Distributed Intrusion Detection Systems Using Bayesian Methods, www.ists.dartmouth.edu/IRIA/projects/ipccc.final.pdf, 2002.

[29] J. Cabrera, B. Ravichandran and R. Mehra, Statistical Traffic Modeling For Network Intrusion Detection, In *Proceedings of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, San Francisco, CA, August 2000.

[30] J. Cannady, Artificial Neural Networks For Misuse Detection, In *Proceedings of the National Information Systems Security Conference (NISSC'98)*, Arlington, VA, 443-456, October, 1998.

[31] J. Cannady and J. Harrell, A Comparative Analysis of Current Intrusion Detection Technologies, In *Proceedings of the Fourth Technology for Information Security Conference'96 (TIS'96)*, Houston, TX, May 1996.

[32] CERIAS Intrusion Detection Resources, http://www.cerias.purdue.edu/coast/ids/ids-body.html, 2004.

[33] CERT® Advisory CA-1995-13 Syslog Vulnerability - A Workaround for Sendmail, http://www.cert.org/advisories/CA-1995-13.html, September, 1997.

[34] CERT® Advisory CA-1999-04 Melissa Worm and Macro Virus, http://www.cert.org/advisories/CA-1999-04.html, March 1999.

[35] CERT® Advisory CA-2000-14 Microsoft Outlook and Outlook Express Cache Bypass Vulnerability, http://www.cert.org/advisories/CA-2000-14.html, July 2000.

[36] CERT® Advisory CA-2001-26 Nimda Worm, http://www.cert.org/advisories/CA-2001-26.html, September 2001.

[37] CERT® Advisory CA-2003-04 MS-SQL Server Worm, http://www.cert.org/advisories/CA-2003-04.html, 2003.

[38] CERT® Advisory CA-2003-25 Buffer Overflow in Sendmail, http://www.cert.org/advisories/CA-2003-25.html, September, 2003.

[39] P.C. Chan and V.K. Wei, Preemptive Distributed Intrusion Detection Using Mobile Agents, In *Proceedings of the Eleventh IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE 2002)*, Pittsburgh, PA, June 2002.

[40] N. Chawla, A. Lazarevic, L. Hall and K. Bowyer, SMOTEBoost: Improving the Prediction of Minority Class in Boosting, In *Proceedings of the Principles of Knowledge Discovery in Databases, PKDD-2003*, Cavtat, Croatia, September 2003.

[41] C. Cheng, H.T. Kung and K. Tan, Use of Spectral Analysis in Defense Against DoS Attacks, In *Proceedings of the IEEE GLOBECOM*, Taipei, Taiwan, 2002.

[42] W.R. Cheswick and S.M. Bellovin, *Firewalls and Internet Security - Repelling the Wily Hacker*, Addison-Wesley, ISBN 0-201-63357-4, 1994.

[43] R. Chinchani, S. Upadhyaya and K. Kwiat, A Tamper-Resistant Framework for Unambiguous Detection of Attacks in User Space Using Process Monitors, In *Proceedings of the IEEE International Workshop on Information Assurance*, Darmstadt, Germany, March 2003.

[44] R. Chinchani, S. Upadhyaya and K. Kwiat, Towards the Scalable Implementation of a User Level Anomaly Detection System, In *Proceedings of the IEEE Conference on Military Communications Conference (MILCOM)*, Anaheim, CA, October 2002.

[45] J. Christy, Cyber Threat & Legal Issues, In *Proceedings of the ShadowCon'99*, Dahlgren, VA, October 26, 1999.

[46] Cisco Intrusion Detection, www.cisco.com/warp/public/cc/pd/sqsw/sqidsz, May 2004.

[47] Cisco Systems, Inc., NetRanger-Enterprise-scale, Real-time, Network Intrusion Detection System, http://www.cisco.com/univercd/cc/td/doc/product/iaabu/netrangr/, 1998.

[48] cknow.com Virus Tutorial, http://www.cknow.com/vtutor/vtmap.htm, 2001.

[49] C. Cowan, C. Pu, D. Maier, H. Hinton, J. Walpole, P. Bakke, S. Beattie, A. Grier and P. Zhang, StackGuard: Automatic Adaptive Detection and Prevention of Buffer-Overflow Attacks, In *Proceedings of the 7th USENIX Security Symposium*, San Antonio, TX, 63-77.

[50] O. Dain and R. Cunningham, Fusing a Heterogeneous Alert Stream Into Scenarios, In *Proceedings of the ACM Workshops on Data Mining for Security Applications*, Philadelphia, PA, November 2001.

[51] V. Dao and R. Vemuri, Computer Network Intrusion Detection: A Comparison of Neural Networks Methods, Differential Equations and Dynamical Systems, *Special Issue on Neural Networks*, 2002.

[52] DARPA, DARPA Intrusion Detection Evaluation, http://www.ll.mit.edu/IST/ideval/pubs/pubs_index.html, 2004.

[53] J. De Queiroz and Carmo L., MICHAEL: An Autonomous Mobile Agent System to Protect New Generation Networked Applications, In *Proceedings of the 2nd Annual Workshop n Recent Advances in Intrusion Detection*, Rio de Janeiro, Brasil, 1999.

[54] H. Debar, M. Becker and D. Siboni, A Neural Network Component for an Intrusion-Detection System, In *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, Oakland, CA, 240-250, May 1992.

[55] H. Debar, M. Dacier and A. Wespi, Towards a Taxonomy of Intrusion Detection Systems, *Computer Networks*, vol. 31, 8, pp. 805-822, 1999.

[56] D. Denning, An Intrusion-Detection Model, *IEEE Transactions on Software Engineering*, vol. 13, 2, pp. 222-232, 1987.

[57] dmoz Open Security Project, Intrusion Detection Systems, http://dmoz.org/Computers/Security/Intrusion_Detection_Systems/,

[58] C. Dowell and P. Ramstedt, The Computerwatch Data Reduction Tool, In *Proceedings of the 13th National Computer Security Conference*, Washington, DC, 1990.

[59] N. Einwechter, An Introduction To Distributed Intrusion Detection Systems, *Security Focus*, January 2002.

[60] D. Engelhardt, Directions for Intrusion Detection and Response: A survey, DSTO Electronics and Surveillance Research Laboratory, Department of Defense, Australia Technical Report DSTO-GD-0155, 1997.

[61] L Ertoz, E. Eilertson, A. Lazarevic, P. Tan, J. Srivastava, V. Kumar and P. Dokas, The MINDS - Minnesota Intrusion Detection System, in *Data Mining: Next Generation Challenges and Future Directions*, A. Joshi H. Kargupta, K. Sivakumar, and Y. Yesha, Ed., 2004.

[62] L. Ertoz, E. Eilertson, P. Dokas, V. Kumar and K. Long, Scan Detection - Revisited, Army High Performance Computing Research Center Technical Report, 2004.

[63] S. Eschrich, Real-Time User Identification Employing Standard Unix Accounting, Florida State University PhD Thesis, Fall 1995.

[64] E. Eskin, Anomaly Detection over Noisy Data using Learned Probability Distributions, In *Proceedings of the International Conference on Machine Learning*, Stanford University, CA, June 2000.

[65] E. Eskin, A. Arnold, M. Prerau, L. Portnoy and S. Stolfo, A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data, in *Applications of Data Mining in Computer Security*, *Advances In Information Security*, S. Jajodia D. Barbara, Ed. Boston: Kluwer Academic Publishers, 2002.

[66] M. Esmaili, B. Balachandran, R. Safavi-Naini and J. Pieprzyk, Case-Based Reasoning For Intrusion Detection, In *Proceedings of the 12th Annual Computer Security Applications Conference*, San Diego, CA, December 1996.

[67] M. Esmaili, R. Safavi-Naini and B.M. Balachandran, Autoguard: A Continuous Case-Based Intrusion Detection System, In *Proceedings of the Australian Computer Science*

Conference, Australian Computer Science Communications, Sydney, Australia, 392-401, February 1997.

[68] W. Fan, W. Lee, M. Miller, S.J. Stolfo and P.K. Chan, Using Artificial Anomalies to Detect Unknown and Known Network Intrusions, In *Proceedings of the First IEEE International conference on Data Mining*, vol. San Jose, CA, December 2001.

[69] D. Farmer, Cops Overview, http://www.trouble.org/cops/overview.html, May 1993.

[70] D. Farmer and W. Venema, Improving The Security Of Your Site By Breaking Into It, http://www.trouble.org/security/admin-guide-to-cracking.html,

[71] H. Feng, O. Kolesnikov, P. Fogla, W. Lee and W. Gong, Anomaly Detection Using Call Stack Information, In *Proceedings of the IEEE Symposium Security and Privacy*, Oakland, CA, May 2003.

[72] G. Florez, S. Bridges and R. Vaughn, An Improved Algorithm for Fuzzy Data Mining for Intrusion Detection, In *Proceedings of the North American Fuzzy Information Processing Society Conference (NAFIPS 2002)*, New Orleans, LA, June, 2002.

[73] S. Forrest, S. Hofmeyr, A. Somayaji and T. Longstaff, A Sense of Self for Unix Processes, In *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, 120-128, May 1996.

[74] A. Ghosh and A. Schwartzbard, A Study in Using Neural Networks for Anomaly and Misuse Detection, In *Proceedings of the Eighth USENIX Security Symposium*, Washington, D.C., 141-151, August , 1999.

[75] T.M Gil and M. Poletto, MULTOPS: A Data-Structure for Bandwidth Attack Detection, In *Proceedings of the USENIX Security Symposium*, Washington, D.C., 23-28, July 2001.

[76] Google directory, http://directory.google.com/Top/Computers/Security/Intrusion_Detection_Systems,

[77] N. Habra, B. LeCharlier, A. Mounji and I. Mathieu, ASAX: Software Architecture and Rule-Based Language for Universal Audit Trail Analysis, In *Proceedings of the Second European Symposium on Research in Computer Security (ESORICS), Vol. 648, Lecture Notes in Computer Science, Springer-Verlag*, Toulouse, France, November 1992.

[78] S.E. Hansen and E.T. Atkins, Automated System Monitoring and Notification With Swatch., In *Proceedings of the Seventh Systems Administration Conference (LISA'93)*, Monterey, CA, November 1993.

[79] S. Hawkins, H. He, G. Williams and R. Baxter, Outlier Detection Using Replicator Neural Networks, In *Proceedings of the 4th International Conference on Data Warehousing and Knowledge Discovery (DaWaK02), Lecture Notes in Computer Science 2454*, Aix-en-Provence, France, 170-180, September 2002.

[80] Haystack Labs, Inc., Stalker, http://www.haystack.com/stalk.htm, 1997.

[81] L.T. Heberlein, G.V. Dias, K.N. Levitt, B. Mukherjee, J. Wood and D. Wolber, A Network Security Monitor, In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Oakland, CA, 296-304, May 1990.

[82] G. Helmer, J.S.K Wong, V. Honavar and L. Miller, Intelligent Agents for Intrusion Detection, In *Proceedings of the IEEE Information Technology Conference*, Syracuse, NY, 121-124, September 1998.

[83] K. Houle, G. Weaver, N. Long and R. Thomas, Trends in Denial of Service Attack Technology, CERT® Coordination Center, Pittsburgh, PA October 2001.

[84] J.D. Howard, An Analysis of Security Incidents on the Internet, Carnegie Mellon University, Pittsburgh, PA 15213 Ph.D. dissertation, April 1997.

[85] D. Hughes, TkLogger, ftp://coast.cs.purdue.edu/pub/tools/unix/tklogger.tar.Z,

[86] K. Ilgun, USTAT A Real-time Intrusion Detection System for UNIX, University of California Santa Barbara Master Thesis, 1992.

[87] Internet Guide, Computer Viruses / Virus Guide, http://www.internet-guide.co.uk/viruses.html, 2002.

[88] Internet Security Systems Wireless Products, Active Wireless Protection, An X-Force's white paper, available at: documents.iss.net/whitepapers/ActiveWirelessProtection.pdf, September 2002.

[89] Internet Security Systems, Inc., RealSecure, http://www.iss.net/prod/rsds.html, 1997.

[90] Intrusion.com, Intrusion SecureHost, white paper available at: www.intrusion.com/products/hids.asp, 2003.

[91] J. Ioannidis and S. Bellovin, Implementing Pushback: Router-Based Defense Against DDoS Attacks, In *Proceedings of the Network and Distributed System Security Symposium*, San Diego, CA, February 2002.

[92] K. Jackson, Intrusion Detection System Product Survey, Los Alamos National Laboratory Research Report, LA-UR-99-3883, June 1999.

[93] R. Jagannathan, T. Lunt, D. Anderson, C. Dodd, F. Gilham, C. Jalali, H. Javitz, P. Neumann, A. Tamaru and A. Valdes, System Design Document: Next-Generation Intrusion Detection Expert System (NIDES). SRI International Technical Report A007/A008/A009/A011/A012/A014, March 1993.

[94] W. Jansen and P. Mell, Mobile Agents in Intrusion Detection and Response, In *Proceedings of the 12th Annual Canadian Information Technology Security Symposium*, Ottawa, Canada, 2000.

[95] H.S. Javitz and A. Valdes, The SRI IDES Statistical Anomaly Detector, In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Oakland, CA, 1991.

[96] N.D. Jayaram and P.L.R. Morse, Network Security - A Taxonomic View, In *Proceedings of the European Conference on Security and Detection*, School of Computer Science, University of Westminster, UK, Publication No. 437, 28-30, April 1997.

[97] A. Jones and R. Sielken, Computer System Intrusion Detection, University of Virginia Technical Report, 1999.

[98] M. Joshi, R. Agarwal and V. Kumar, PNrule, Mining Needles in a Haystack: Classifying Rare Classes via Two-Phase Rule Induction, In *Proceedings of the ACM SIGMOD Conference on Management of Data*, Santa Barbara, CA, May 2001.

[99] M. Joshi, R. Agarwal and V. Kumar, Predicting Rare Classes: Can Boosting Make Any Weak Learner Strong?, In *Proceedings of the Eight ACM Conference ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Edmonton, Canada, July 2002.

[100] Y.F. Jou, F. Gong, C. Sargor, S.F. Wu and W.R. Cleaveland, Architecture Design of a Scalable Intrusion Detection System For The Emerging Network Infrastructure, MCNC Information Technologies Division, Research Triangle Park, NC 27709 Technical Report CDRL A005, April 1997.

[101] K. Julisch, Mining Alarm Clusters to Improve Alarm Handling Efficiency, In *Proceedings of the 17th Annual Conference on Computer Security Applications*, New Orleans, LA, December 2001.

[102] J. Jung, V. Paxson, A. W. Berger and H. Balakrishnan, Fast Portscan Detection Using Sequential Hypothesis Testing, In *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, May, 2004.

[103] K. Kendall, A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems, Massachusetts Institute of Technology Master's Thesis, 1998.

[104] A.D. Keromytis, V. Misra and D. Rubenstein, SoS: Secure Overlay Services, In *Proceedings of the ACM SIGCOMM Conference*, Pittsburgh, PA, 61-72, August 2002.

[105] D. Kienzle and M. Elder, Recent Worms. A Survey and Trends, In *Proceedings of the The Workshop on Rapid Malcode (WORM 2003), held in conjunction with the 10th ACM Conference on Computer and Communications Security*, Washington, DC, October 27, 2003.

[106] G. Kim and E. Spafford, The Design and Implementation of Tripwire: A File System Integrity Checker, In *Proceedings of the ACM Conference on Computer and Communications Security, COAST*, Purdue University, IN, 18-29, November 1994.

[107] E. Knorr and R. Ng, Algorithms for Mining Distance based Outliers in Large Data Sets, In *Proceedings of the Very Large Databases (VLDB) Conference*, New York City, NY, August 1998.

[108] I.V. Krsul, Software Vulnerability Analysis, Purdue University Ph.D. dissertation, May 1998.

[109] C. Kruegel and T. Toth, Distributed Pattern Detection For Intrusion Detection, In *Proceedings of the Network and Distributed System Security Symposium Conference Proceedings, Internet Society*, Los Angeles, CA, February 2002.

[110] C. Krugel and T. Toth, A Survey on Intrusion Detection Systems, Technical University of Vienna Technical report, TUV-1841-00-11, 2000.

[111] C. Krugel, T. Toth and E. Kirda, Service Specific Anomaly Detection for Network Intrusion Detection, In *Proceedings of the ACM Symposium on Applied Computing*, Madrid, Spain, March 2002.

[112] S. Kumar, Classification and Detection of Computer Intrusion, Computer Science Department, Purdue University Ph.D. dissertation, August 1995.

[113] S. Kumar and E. Spafford, An Application of Pattern Matching in Intrusion Detection, Purdue University Technical Report, 1994.

[114] H. Kvarnstrom, A Survey of Commercial Tools for Intrusion Detection, Chalmers University of Technology, Göteborg, Sweden Technical Report, 1999.

[115] C. Landwehr, A. Bull, J. McDermott and W. Choi, A Taxonomy of Computer Program Security Flaws, *ACM Computing Surveys*, vol. 26, 3, pp. 211-254, September 1994.

[116] T. Lane and C. Brodley, Temporal Sequence Learning and Data Reduction for Anomaly Detection, *ACM Transactions on Information and System Security*, vol. 2, 3, pp. 295-331, 1999.

[117] A. Lazarevic, L. Ertoz, A. Ozgur, J. Srivastava and V. Kumar, A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection, In *Proceedings of the Third SIAM International Conference on Data Mining*, San Francisco, CA, May 2003.

[118] A. Lazarevic, J. Srivastava and V. Kumar, Cyber Threat Analysis - A Key Enabling Technology for the Objective Force (A Case Study in Network Intrusion Detection), In *Proceedings of the IT/C4ISR, 23rd Army Science Conference*, Orlando, FL, December 2002.

[119] W. Lee, S. Stolfo and P. Chan, Patterns from Unix Process Execution Traces for Intrusion Detection, In *Proceedings of the AAAI Workshop: AI Approaches to Fraud Detection and Risk Management*, Providence, RI, July 1997.

[120] W. Lee, S. Stolfo and K. Mok, Adaptive Intrusion Detection: A Data Mining Approach., *Artificial Intelligence Review*, vol. 14, pp. 533-567, 2001.

[121] W. Lee and S.J. Stolfo, Data Mining Approaches for Intrusion Detection, In *Proceedings of the USENIX Security Symposium*, San Antonio, TX, January, 1998.

[122] W. Lee and S.J. Stolfo, A Framework for Constructing Features and Models for Intrusion Detection Systems., *ACM Transactions on Information and System Security*, vol. 3, 4, pp. 227-261, 2000.

[123] W. Lee and D. Xiang, Information-Theoretic Measures for Anomaly Detection, In *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, May 2001.

[124] G. Liepins and H. Vaccaro, Anomaly Detection Purpose and Framework, In *Proceedings of the 12th National Computer Security Conference*, Baltimore, MD, 495-504, October 1989.

[125] G. Liepins and H. Vaccaro, Intrusion Detection: It's Role and Validation, *Computers and Security*, pp. 347-355, 1992.

[126] Y.X. Lim, T. Schmoyer, J. Levine and H.L. Owen, Wireless Intrusion Detection and Response, In *Proceedings of the IEEE Workshop on Information Assurance*, United States Military Academy, West Point, NY, June 2003.

[127] J.L Lin, X.S. Wang and S. Jajodia, Abstraction-Based Misuse Detection: High-Level Specifications and Adaptable Strategies, In *Proceedings of the 11th IEEE Computer Security Foundations Workshop*, Rockport, MA, June 1998.

[128] U. Lindqvist and E. Jonsson, How to Systematically Classify Computer Security Intrusions, *IEEE Security and Privacy*, pp. 154-163, 1997.

[129] U. Lindqvist and P.A. Porras, Detecting Computer and Network Misuse Through the Production-Based Expert System Toolset (P-BEST), In *Proceedings of the IEEE Symposium on Security and Privacy*, Berkeley, CA, May 1999.

[130] R. Lippmann, The Role of Network Intrusion Detection, In *Proceedings of the Workshop on Network Intrusion Detection*, H.E.A.T. Center, Aberdeen, MD, March 19-20, 2002.

[131] R. Lippmann and R. Cunningham, Improving Intrusion Detection Performance Using Keyword Selection and Neural Networks, *Computer Networks*, vol. 34, 4, pp. 597-603, 2000.

[132] R. Lippmann, J.W. Haines, D.J. Fried, J. Korba and K. Das, The 1999 DARPA Off-Line Intrusion Detection Evaluation, *Computer Networks*, 2000.

[133] R.P. Lippmann, R.K. Cunningham, D.J. Fried, I. Graf, K.R. Kendall, S.E. Webster and M.A. Zissman, Results of the DARPA 1998 Offline Intrusion Detection Evaluation, In *Proceedings of the Workshop on Recent Advances in Intrusion Detection, (RAID-1999)*, West Lafayette, IN, September, 1999.

[134] J. Lo, Trojan Horse Attacks, www.irchelp.org/irchelp/security/trojan.html, April 2004.

[135] D. Lough, A Taxonomy of Computer Attacks with Applications to Wireless Networks, Virginia Polytechnic Institute PhD Thesis, April 2001.

[136] T. Lunt, A Survey of Intrusion Detection techniques, *Computers & Security*, vol. 12, 4, pp. 405-418, June 1993.

[137] T. Lunt, R. Jagannathan, R. Lee, S. Listgarten, D.L. Edwards, P.G. Neumann, H.S. Javitz and A. Valdes, IDES: The Enhanced Prototype - A Real-Time Intrusion-Detection Expert System, SRI International Technical Report SRI-CSL-88-12.

[138] T. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, C. Jalali, P.G. Neumann, H.S. Javitz, A. Valdes and T.D. Garvey, A Real Time Intrusion Detection Expert System (IDES), SRI Technical report, 1992.

[139] T.F. Lunt, Real-Time Intrusion Detection, In *Proceedings of the Thirty Fourth IEEE Computer Society International Conference (COMPCON), Intellectual Leverage*, San Francisco, CA, February 1989.

[140] J. Luo, Integrating Fuzzy Logic With Data Mining Methods for Intrusion Detection, Department of Computer Science, Mississippi State University Master's thesis, 1999.

[141] R. Mahajan, S. Bellovin, S. Floyd, J. Ioannidis, V. Paxson and S. Shenker, Controlling High Bandwidth Aggregates in The Network, *ACM Computer Communication Review*, July 2001.

[142] M. Mahoney and P. Chan, Learning Nonstationary Models of Normal Network Traffic for Detecting Novel Attacks, In *Proceedings of the Eight ACM International Conference on Knowledge Discovery and Data Mining*, Edmonton, Canada, 376-385, July 2002.

[143] S. Manganaris, M. Christensen, D. Serkle and K. Hermiz, A Data Mining Analysis of RTID Alarms, *Computer Networks*, vol. 34, 4, October 2000.

[144] D. Marchette, *Computer Intrusion Detection and Network Monitoring, A Statistical Viewpoint*. New York, Springer, 2001.

[145] J. Marin, D. Ragsdale and J. Surdu, A Hybrid Approach to Profile Creation and Intrusion Detection, In *Proceedings of the DARPA Information Survivability Conference and Exposition*, Anaheim, CA, June, 2001.

[146] R. Maxion and K. Tan, Anomaly Detection in Embedded Systems, *IEEE Transactions on Computers*, vol. 51, 2, pp. 108-120, 2002.

[147] Mazu Profiler™, An Overview, http://www.mazunetworks.com/solutions/white_papers/download/Mazu_Profiler.pdf, December 2003.

[148] M. Medina, A Layered Framework for Placement of Distributed Intrusion Detection Devices, In *Proceedings of the 21st National Information Systems Security Conference (NISSC'98)*, Crystal City, VA, October 1998.

[149] Meier. M. and M. Sobirey, Intrusion Detection Systems List and Bibliography, http://www-rnks.informatik.tu-cottbus.de/en/security/ids.html,

[150] Metropolitan, Metropolitan Network BBS, Inc., Kaspersky.ch, Computer Virus Classification, http://www.avp.ch/avpve/classes/classes.stm, 2003.

[151] J. Mirkovic, G. Prier and P. Reiher, Attacking DDoS at the Source, *10th IEEE International Conference on Network Protocols*, November 2002.

[152] J. Mirkovic and P. Reiher, A Taxonomy of DDoS Attacks and Defense Mechanisms, *ACM Computer Communication Review*, April 2004.

[153] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford and N. Weaver, The Spread of the Sapphire/Slammer Worm, http://www.cs.berkeley.edu/~nweaver/sapphire/, 2003.

[154] D. Moore, G. M. Voeker and S. Savage, Inferring Internet Denial-of-Service Activity, *USENIX Security Symposium*, pp. 9-22, August 2001.

[155] A. Mounji, Languages and Tools for Rule-Based Distributed Intrusion Detection, Facult es Universitaires Notre-Dame de la Paix, Namur, Belgium Doctor of Science Thesis, September 1997.

[156] S. Mukkamala, G. Janoski and A. Sung, Intrusion Detection Using Neural Networks and Support Vector Machines, In *Proceedings of the IEEE International Joint Conference on Neural Networks*, Honolulu, HI, May 2002.

[157] S. Mukkamala, A. Sung and A. Abraham, Intrusion Detection Systems Using Adaptive Regression Splines, In *Proceedings of the 1st Indian International Conference on Artificial Intelligence (IICAI-03)*, Hyderabad, India, December 2003.

[158] S. Mukkamala, A. Sung and A. Abraham, A Linear Genetic Programming Approach for Modeling Intrusion, In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC2003)*, Perth, Australia, December, 2003.

[159] NAGIOS Network Monitoring Tool, www.nagios.org, February 2004.

[160] Nessus Network Security Scanner, http://www.nessus.org/, 2004.

[161] Netflow Tools, www.netflow.com,

[162] NetForensics®, Security Information Management, http://www.netforensics.com/,

[163] Network Associates, Inc., Cybercop server, http://www.nai.com/products/security/cybercopsvr/index.asp, 1998.

[164] P. Neumann and P. Porras, Experience with Emerald to Date, In *Proceedings of the First Usenix Workshop on Intrusion Detection and Network Monitoring*, Santa Clara, CA, 1999.

[165] P.G. Neumann, *Computer Related Risks*, The ACM Press, a division of the Association for Computing Machinery, Inc. (ACM), 1995.

[166] P.G. Neumann and D.B. Parker, A Summary of Computer Misuse Techniques, In *Proceedings of the 12th National Computer Security Conference*, 396-407, 1989.

[167] NFR Network Intrusion Detection, http://www.nfr.com/products/NID/, 2001.

[168] P. Ning, Y. Cui and D. Reeves, Constructing Attack Scenarios through Correlation of Intrusion Alerts, In *Proceedings of the 9th ACM Conference on Computer & Communications Security*, Washington D.C., 245-254, November 2002.

[169] S. Nomad, Distributed Denial of Service Defense Tactics, http://razor.bindview.com/publish/papers/strategies.html, 2/14/2000.

[170] S. Northcutt, SHADOW, http://www.nswc.navy.mil/ISSEC/CID/, 1998.

[171] K. P. Park and H. Lee, On the Effectiveness of Router-Based Packet Filtering for Distributed Dos Attack Prevention in Power-Law Internets, In *Proceedings of the ACM SIGCOMM Conference*, San Diego, CA, August 2001.

[172] D.B. Parker, Computer Abuse Perpetrators and Vulnerabilities of Computer Systems, Stanford Research Institute, Menlo Park, CA 94025 Technical Report, December 1975.

[173] D.B. Parker, COMPUTER CRIME Criminal Justice Resource Manual, U.S. Department of Justice National Institute of Justice Office of Justice Programs,

Prepared by SRI International under contract to Abt Associates for National Institute of Justice, U.S. Department of Justice, contract #OJP-86-C-002., 1989.

[174] V. Paxson, Bro: A System for Detecting Network Intruders in Real-Time, In *Proceedings of the 7th USENIX Security Symposium*, San Antonio, TX, January 1998.

[175] Pcap, libpcap, winpcap, libdnet, and libnet Applications and Resources, http://www.steams.org/doc/pcap-apps.html, 2004.

[176] T. Peng, C. Leckie and K. Ramamohanarao, Defending Against Distributed Denial of Service Attack Using Selective Pushback, In *Proceedings of the Ninth IEEE International Conference on Telecommunications (ICT 2002)*, Beijing, China, June 2002.

[177] P. Porras, D. Schanckernberg, S. Staniford-Chen, M. Stillman and F. Wu, Common Intrusion Detection Framework Architecture, http://www.gidos.org/drafts/ architecture.txt, 2001.

[178] P.A. Porras and R.A. Kemmerer, Penetration State Transition Analysis: A Rule-Based Intrusion Detection Approach, In *Proceedings of the Eighth Annual Computer Security Applications Conference*, San Antonio, TX, December, 1992.

[179] P.A. Porras and P.G. Neumann, EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances, In *Proceedings of the 20th National Information Systems Security Conference*, Baltimore, MD., 353-365, October, 1997.

[180] P.A. Porras and A. Valdes, Live Traffic Analysis of TCP/IP Gateways, In *Proceedings of the ISOC Symposium on Network and Distributed System Security (NDSS'98)*, San Diego, CA, March 1998.

[181] D. Powell and R. Stroud, Conceptual Model and Architecture, Deliverable D2, Project MAFTIA IST-1999-11583, IBM Zurich Research Laboratory Research Report RZ 3377, Nov. 2001.

[182] Proventia™, Security's Silver Bullet? An Internet Security Systems White Paper, available at:, http://documents.iss.net/whitepapers/ProventiaVision.pdf, 2003.

[183] F. Provost and T. Fawcett, Robust Classification for Imprecise Environments, *Machine Learning*, vol. 42, 3, pp. 203-231, 2001.

[184] T.H. Ptacek and T.N. Newsham, Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection, Secure Networks, Inc Technical Report, January 1998.

[185] Michael Puldy, Lessons Learned in the Implementation of a Multi-Location Network Based Real Time Intrusion Detection System, In *Proceedings of the Workshop on Recent Advances in Intrusion Detection (RAID 98)*, Louvain-la-Neuve, Belgium, September 1998.

[186] X. Qin and W. Lee, Statistical Causality Analysis of INFOSEC Alert Data, In *Proceedings of the 6th International Symposium on Recent Advances in Intrusion Detection (RAID 2003)*, Pittsburgh, PA, September 2003.

[187] S. Ramaswamy, R. Rastogi and K. Shim, Efficient Algorithms for Mining Outliers from Large Data Sets, In *Proceedings of the ACM SIGMOD Conference*, Dallas, TX, May 2000.

[188] M.J. Ranum, K. Landfield, M. Stolarchuk, M. Sienkiewicz, A. Lambeth and Wall E., Implementing a Generalized Tool for Network Monitoring, In *Proceedings of the Eleventh Systems Administration Conference (LISA'97)*, San Diego, CA, October 1997.

[189] T. Richardson, The Development of a Database Taxonomy of Vulnerabilities to Support the Study of Denial of Service Attacks., Iowa State University PhD Thesis, 2001.

[190] T. Richardson, J. Davis, D. Jacobson, J. Dickerson and L. Elkin, Developing a Database of Vulnerabilities to Support the Study of Denial of Service Attacks, *IEEE Symposium on Security and Privacy*, May 1999.

[191] S. Robertson, E. Siegel, M. Miller and S. Stolfo, Surveillance Detection in High Bandwidth Environments, In *Proceedings of the 3rd DARPA Information Survivability Conference and Exposition (DISCEX 2003)*, Washington DC, April 2003.

[192] P. Rolin, L. Toutain and S. Gombault, Network Security Probe, In *Proceedings of the 2nd ACM Conference on Computer and Communication Security (ACM CCS'94)*, Fairfax, VA, 229-240, November 1994.

[193] J. Ryan, M-J. Lin and R. Miikkulainen, Intrusion Detection with Neural Networks, In *Proceedings of the AAAI Workshop on AI Approaches to Fraud Detection and Risk Management*, Providence, RI, 72-77, July 1997.

[194] D. Safford, D. Schales and D. Hess, The Tamu Security Package: An Ongoing Response to Internet Intruders in an Academic Environment, In *Proceedings of the Fourth USENIX Security Symposium*, Santa Clara, CA, 91-118, October 1993.

[195] S. Savage, D. Wetherall, A. Karlin and T. Anderson, Practical Network Support for IP Traceback, In *Proceedings of the ACM SIGCOMM Conference*, Stockholm, Sweden, 295-306, August 2000.

[196] M. Schultz, E. Eskin, E. Zadok and S. Stolfo, Data Mining Methods for Detection of New Malicious Executables, In *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, 38-49, May 2001.

[197] Secure Networks, Inc., Ballista Security Auditing System, http:// www.securenetworks.com/ballista/ballista.html, 1997.

[198] SecurityTechNet.com Intrusion Detection Links, http://cnscenter.future.co.kr/security/ ids.html, 2004.

[199] R. Sekar, A. Gupta, J. Frullo, T. Shanbhag, A. Tiwari, H. Yang and S. Zhou, Specification Based Anomaly Detection: A New Approach for Detecting Network Intrusions, In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, Washington, D.C., November 2002.

[200] A. Seleznyov and S. Puuronen, HIDSUR: A Hybrid Intrusion Detection System Based on Real-Time User Recognition, In *Proceedings of the 11th International Workshop on Database and Expert Systems Applications (DEXA'00)*, Greenwich, London, UK, September, 2000.

[201] K. Sequeira and M. Zaki, ADMIT: Anomaly-base Data Mining for Intrusions, In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Edmonton, Canada, July 2002.

[202] C. Sinclair, L. Pierce and S. Matzner, An Application of Machine Learning to Network Intrusion Detection, In *Proceedings of the 15th Annual Computer Security Applications Conference*, Phoenix, AZ, 371-377, December 1999.

[203] S. Singh and Kandula S., Argus: A Distributed Network Intrusion Detection System, Indian Institute of Technology Kanpur, Department of Computer Science &

Engineering, available at: http://www.cse.iitk.ac.in/research/btp2001/Argus.html Technical Report, 2001.

[204] S. Smaha, Haystack: An Intrusion Detection System, In *Proceedings of the Fourth Aerospace Computer Security Applications Conference*, 37-44, October 1988.

[205] S.R. Snapp, J. Brentano, G.V. Dias, T.L. Goan, T. Heberlein, C. Ho, K.N. Levitt, B. Mukherjee, S.E. Smaha, T. Grance, D.M. Teal and D. Mansur, DIDS (Distributed Intrusion Detection System) Motivation, Architecture, and an Early Prototype, In *Proceedings of the 14th National Computer Security Conference*, Washington, DC, 167-176, October 1991.

[206] A.C. Snoeren, C. Partridge, L.A. Sanchez, C.E Jones, F. Tchakountio, S.T. Kent and W.T. Strayer, Hash-Based IP Traceback, In *Proceedings of the ACM SIGCOMM Conference*, San Diego, CA, 3-14, August 2001.

[207] SNORT Intrusion Detection System, www.snort.org, 2004.

[208] Snort-Wireless Intrusion Detection, http://snort-wireless.org, 2003.

[209] A. Somayaji, S. Hofmeyr and S. Forrest, Principles of a computer immune system, In *Proceedings of the New Security Paradigms Workshop*, Langdale, Cumbria UK, 1997.

[210] Sourcefire, Sourcefire Real-time Network Awareness™ (RNA), http://www.sourcefire.com/products/rna.html, 2004.

[211] E. Spafford and D. Zamboni, Intrusion Detection Using Autonomous Agents, *Computer Networks*, vol. 34, pp. 547-570, 2000.

[212] P. Spirakis, S. Katsikas, D. Gritzalis, F. Allegre, J. Darzentas, C. Gigante, D. Karagiannis, P. Kess, H. Putkonen and T. Spyrou, SECURENET: A Network-Oriented Intelligent Intrusion Prevention And Detection System., *Network Security Journal*, vol. 1, 1, November 1994.

[213] T. Spyrou and J. Darzentas, Intention Modelling: Approximating Computer User Intentions for Detection and Prediction of Intrusions, In *Proceedings of the Information Systems Security*, Samos, Greece, 319-335, May 1996.

[214] S. Staniford, J. Hoagland and J. McAlerney, Practical Automated Detection of Stealthy Portscans, *Journal of Computer Security*, vol. 10, 1-2, pp. 105-136, 2002.

[215] S. Staniford, V. Paxson and N. Weaver, How to Own the Internet in Your Spare Time, In *Proceedings of the USENIX Security Symposium*, San Francisco, CA, 149-167, August 2002.

[216] S. Staniford-Chen, C.R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip and D. Zerkle, GrIDS - A Graph Based Intrusion Detection System for Large Networks, In *Proceedings of the 19th National Information Systems Security Conference*, Baltimore, MD.

[217] S. Staniford-Chen, B. Tung, P. Porras, C. Kahn, D. Schnackenberg, R. Feiertag and M. Stillman, The Common Intrusion Detection Framework - Data Formats, Internet Draft Draft-ietf-cidf-data-formats-00.txt, March 1998.

[218] R. Stone, Centertrack: An IP Overlay Network for Tracking DoS Floods, In *Proceedings of the USENIX Security Symposium*, Denver, CO, 199-212, July 2000.

[219] SunSHIELD Basic Security Module Guide, http://docs.sun.com/db/doc/802-1965?q=BSM, 1995.

[220] Symantec Intruder Alert, http://enterprisesecurity.symantec.com/products/products.cfm?ProductID=171&EID=0, May 2004.

[221] Symantec Security Response, W32.ExploreZip.L.Worm, http:// securityresponse.symantec.com/avcenter/venc/data/w32.explorezip.l.worm.html, January 2003.

[222] System Detection, Anomaly Detection: The Antura Difference, http:// www.sysd.com/library/anomaly.pdf, 2003.

[223] Talisker's Network Security Resource, http://www.networkintrusion.co.uk/ids.htm,

[224] TCPDUMP public repository, www.tcpdump.org,

[225] S. Templeton and K. Levit, A Requires/Provides Model for Computer Attacks, In *Proceedings of the Workshop on New Security Paradigms*, Ballycotton, Ireland, 2000.

[226] B. Tod, Distributed Denial of Service Attacks, OVEN Digital, http://www.linuxsecurity.com/resource_files/intrusion_detection/ddos-faq.html, 2000.

[227] A. Valdes, Detecting Novel Scans Through Pattern Anomaly Detection, In *Proceedings of the Third DARPA Information Survivability Conference and Exposition (DISCEX-III 2003)*, Washington, D.C., April 2003.

[228] A. Valdes and K. Skinner, Adaptive, Model-based Monitoring for Cyber Attack Detection, In *Proceedings of the Recent Advances in Intrusion Detection (RAID 2000)*, Toulouse, France, 80-92, October 2000.

[229] A. Valdes and K. Skinner, Probabilistic Alert Correlation, In *Proceedings of the Recent Advances in Intrusion Detection (RAID 2001)*, Davis, CA, October 2001.

[230] J. Van Ryan, SAIC's Center for Information Security, Technology Releases CMDS Version 3.5, http://www.saic.com/news/may98/news05-15-98.html, 1998.

[231] Vicomsoft White Paper, Firewall White Paper - What Different Types of Firewalls are There?, available at:, http://www.firewall-software.com/firewall_faqs/ types_of_firewall.html, 2003.

[232] G. Vigna and R.A. Kemmerer, Netstat: A Network-Based Intrusion Detection Approach, *Journal of Computer Security*, vol. 7, 1, pp. 37-71, 1999.

[233] D. Vincenzetti and M. Cotrozzi, ATP - Anti Tampering Program, In *Proceedings of the Fourth USENIX Security Symposium*, Santa Clara, CA, 79-89, October 1993.

[234] D. Wagner and D. Dean, Intrusion Detection via Static Analysis, In *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, May 2001.

[235] H. Wang, D. Zhang and K. Shin, Detecting SYN Flooding Attacks, In *Proceedings of the IEEE Infocom*, New York, NY, 000-001, June 2002.

[236] N. Weaver, V. Paxson, S. Staniford and R. Cunningham, A Taxonomy of Computer Worms, In *Proceedings of the The Workshop on Rapid Malcode (WORM 2003), held in conjunction with the 10th ACM Conference on Computer and Communications Security*, Washington, DC, October 27, 2003.

[237] A. Wespi, M. Dacier and H. Debar, Intrusion Detection Using Variable-Length Audit Trail Patterns, In *Proceedings of the Recent Advances in Intrusion Detection (RAID-2000)*, Toulouse, FR, 110-129, October 2000.

[238] WheelGroup Corporation, Cisco Secure Intrusion Detection System, http://www.cisco.com/univercd/cc/td/doc/product/iaabu/csids/index.htm, 2004.

[239] WIDZ Wireless Intrusion Detection System, www.loud-fat-bloke.co.uk/articles/ widz_design.pdf.

[240] D. Winer, Clay Shirky on P2P, davenet.scripting.com/2000/11/15/clayShirkyOnP2p, November 2000.

[241] J.R. Winkler, A Unix Prototype for Intrusion and Anomaly Detection in Secure Networks, In *Proceedings of the 13th National Computer Security Conference*, Baltimore, MD, October 1990.

[242] J.R. Winkler and L.C. Landry, Intrusion and Anomaly Detection, ISOA Update, In *Proceedings of the 15th National Computer Security Conference*, Baltimore, MD, October 1992.

[243] K. Yamanishi and J. Takeuchi, Discovering Outlier Filtering Rules from Unlabeled Data, In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, August 2001.

[244] K. Yamanishi, J. Takeuchi, G. Williams and P. Milne, On-line Unsupervised Outlier Detection Using Finite Mixtures with Discounting Learning Algorithms, In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Boston, MA, 320-324, August 2000.

[245] N. Ye and Q. Chen, An Anomaly Detection Technique Based on a Chi-Square Statistic for Detecting Intrusions Into Information Systems, *Quality and Reliability Engineering International*, vol. 17, 2, pp. 105-112, 2001.

[246] N. Ye and X. Li, A Scalable Clustering Technique for Intrusion Signature Recognition, In *Proceedings of the 2001 IEEE Workshop on Information Assurance and Security*, United States Military Academy, West Point, NY, June, 2001.

[247] Z. Zhang, J. Li, C.N. Manikopoulos, J. Jorgenson and J. Ucles, HIDE: A Hierarchical Network Intrusion Detection System Using Statistical Preprocessing and Neural Network Classification, In *Proceedings of the IEEE Workshop on Information Assurance and Security*, United States Military Academy, West Point, NY, June 2001.

[248] E. Zwicky, S. Cooper, D. Chapman and D. Ru, *Building Internet Firewalls*, 2nd Edition ed, O'Reilly and Associates, 2000.

PART II

DATA MINING BASED ANALYSIS OF COMPUTER ATTACKS