Crossroads
The ACM Student Magazine

Computer Security and Intrusion Detection

by ***Khaled Labib***

Introduction

Computer attacks are now commonplace. By connecting your computer to the Internet, you increase the risk of having someone break in, install malicious programs and tools on it, and possibly use it to attack other machines on the Internet by controlling it remotely.

Several major banks have been subject to attacks, in which attackers gained access into customers' accounts and viewed detailed information about the activities on these accounts. In some instances the attackers stole credit card information to blackmail e-commerce companies by threatening to sell this information to unauthorized entities. Several online trading companies and e-commerce sites were shut down temporarily due to major packet flood attacks, also known as Denial-of-Service (DoS) attacks, causing these companies to lose revenue, customer satisfaction, and trust [**10**]. A major software development company discovered that attackers had broken into its network and stolen the source code for future releases of its popular products. Just recently, the source code of the future flagship product belonging to a major software development company was stolen and made publicly available on the Internet.

In order to combat this growing trend of computer attacks, both academic and industry groups have been developing systems to monitor networks and systems and raise alarms of suspicious activities. These systems are called Intrusion Detection Systems (IDS).

Anatomy of An Attack

Before discussing the recent work in IDS development, a brief introduction to the nature and methodology of a typical computer attack is provided in this section.

Computer attacks generally follow a five step approach as described below.

1. **Reconnaissance**: Before launching an attack, attackers conduct detailed reconnaissance to collect information about their prey. This process typically involves using low-technology reconnaissance, general web searches, the "whois" database, and the Domain Name System (DNS).
2. **Scanning**: The attacker, equipped with information about the infrastructure of the victim's network, begins scanning the victim's systems looking for vulnerabilities and openings. At the end of this phase the attacker will have gained valuable information about the victim's network, including lists of phone numbers with modems, addresses of live hosts, network topology, open ports, and firewall rule sets. There are a number of powerful freely-available network scanners on the web for that purpose.
3. **Gaining access**: If the attacker is a legitimate user of the system, then most likely he/she will attempt to gain access using operating system and application attacks. If the attacker is an outsider, then the attack is most likely to be through the network.
    - **Operating System and Application attacks**: This approach depends on the skill of the attacker with simple inexperienced attackers, usually referred to as **"script kiddies**," utilizing prepackaged exploits to more advanced attackers using highly systematic approaches. Generally, variations of the operating system buffer overflow attacks are used to gain root access to the target. In addition, password guessing is used as an entry point to log in to the target.
    - **Network attacks**: Network sniffers are usually utilized by attackers to collect Data Link Layer (DLL) information from all computers on the same subnet. A **sniffer** is a program that gathers traffic data from the network. In addition, other techniques like spoofing and session hijacking are used typically with a freely available tool called Netcat. In some cases, attackers are not interested in gaining access to the network, but would just like to prevent legitimate users from accessing its resources. In this case, the attackers would launch a DoS attack to consume the resources of the network and computers, especially web servers.
4. **Maintaining Access**: Now that the attackers have gained access to the target system, they need to maintain this access. Many techniques are utilized here, based on malicious software such as Trojan horses, backdoors, and RootKits. A **Trojan horse** is a program that looks like it has a benign or beneficial purpose, but is actually implementing some malicious function. A **RootKit** is a tool that allows an

attacker to maintain super-user access on a machine by modifying system software.

5. **Covering Tracks**: Some of the main techniques used by hackers to hide their tracks are backdoor programs and RootKits. Beyond that, attackers will also attempt to modify system logs and create covert channels, which are hidden communication paths used to transmit data so that the victim will not see the data.

What is an IDS and Why Do We Need One?

Intrusion is a set of actions that attempt to compromise the integrity, confidentiality, or availability of any resource on a computing platform. An IDS is a software tool that attempts to detect an intruder hacking into a system or a genuine user exploiting the resources of the system.

An IDS is a piece of software that runs on a host, which monitors the activities of users and programs on the host and monitors the network traffic on networks to which the host is attached. The objective of an IDS is to alarm the system's administrator of any suspicious and possibly intrusive event and possibly taking action to circumvent the intrusion. These actions can be as simple as writing the activities to a log file or as complex as controlling the system's and network's resources automatically by closing network ports or killing suspicious processes.

The objective of an IDS is to detect all intrusive actions for both successful and unsuccessful attempts with 100% confidence on the network under consideration.

Taxonomy of IDS

In the infancy of IDS research, two major approaches known as **anomaly detection** and **signature detection** were developed. The former relies on flagging behaviors that are abnormal and the latter flagging behaviors that are close to some previously defined pattern signature of a known intrusion [**2**]. A third approach that was introduced more recently by the University of California, Davis, referred to as **specification-based intrusion detection**, relies on manually setting program behavioral specifications that are used as a basis to detect attacks. This technique has been proposed as a promising alternative that combines the strengths of signature-based and anomaly-based detection [**3**].

IDSs may also be characterized by scope, as either *network-based* or *host-based*. The key difference between network-based and host-based IDSs is that a network-based IDS, although run on a single host, is responsible for an entire network, or some network segment, while a host-based IDS is only responsible for the host on which it resides [**11**].

The detection of intrusions or system abuses presupposes the existence of a model [4]. In **signature detection**, also referred to as **misuse detection**, the known attack patterns are modeled through the construction of a library of attack signatures. Incoming patterns that match an element of the library are labeled as attacks. If only exact matching is allowed, misuse detectors operate with no false alarms. By allowing some tolerance in attack matching, there is a risk of false alarms, but the detector is expected to be able to detect certain classes of unknown attacks that do not deviate much from the attacks listed in the library. Such attacks are called **neighboring attacks**. In anomaly detection, the normal behavior of the system is modeled. Incoming patterns that deviate substantially from normal behavior are labeled as attacks. The premise that malicious activity is a subset of anomalous activity implies that the abnormal patterns can be utilized to indicate attacks. The presence of false alarms is expected in this case in exchange for the hope of detecting unknown attacks, which may be substantially different from neighboring attacks. These are called **novel attacks**. Detecting novel attacks, while keeping acceptably low rates of false alarm, is possibly the most challenging and important problem in intrusion detection.

Which Algorithms Have Been Used to Implement IDSs?

Below is a typical flow of how IDSs work. The flow can be divided into the following tasks:

1. **Data Collection**: For network-based intrusion detection, this step involves collecting the network traffic using a sniffer software, such as `tcpdump`. These programs have filters that allow us to select the types of packets, hosts, and protocols we are interested in analyzing and can collect data from the network interface in real-time and write them to a file. For host-based intrusion detection a similar process takes place, but this time data, such as process activity, disk usage, memory usage, and system calls are collected and logged in a file for later analysis. Many tools are used for this purpose including UNIX commands such as `netstat`, `ps`, and `strace`.
2. **Feature Selection**: The raw data collected is usually substantially large, so it is desired that we only select a subset of this data by creating feature vectors that represent most of the information we need from the data. In network-based intrusion detection, typically the Internet Protocol (IP) packet header information is selected as the basis of the feature vector, which includes source and destination IP addresses, packet length, layer four protocol type, and other flags. Some research work utilizes more data (beyond layer four) from the packet for analysis. Each technique has its merits and demerits. Other work does not look at raw packet data directly from the network interface but rather defines a **connection** to be a set of related packets that make up a connection, typically a Transport Control Protocol (TCP) connection, and uses the connection information for the analysis phase. For host-based intrusion

detection, feature vectors may include user name, login time and date, duration of the session and number of opened files.

3. **Analysis**: The collected data is analyzed to determine if there is an attack signature or whether the data is anomalous as compared to normal traffic. This is the main research area where a large number of methods have been used and analyzed for detecting intrusions.
4. **Action**: The IDS alerts the system administrator for a possible attack and provides information to the administrator as to nature of the attacks using several methods including e-mail, alarm icons, and visualization techniques. The IDS can also have an active role of stopping or controlling the attack by closing network ports or killing processes.

Several implementations of IDSs use statistical methods such as k-means and clustering algorithms to find clusters in the data sets [**7**]. Most of the algorithms use Euclidean distance measures. Other implementations use neural networks to learn the behavior of attacks [**5**]. Variations of the back-propagation algorithms, support-vector machines, and radial basis functions are often employed. Several unsupervised learning methods have been applied to this problem, including Self-Organizing Maps [**8**] and Principal Component Analysis [**6**]. These algorithms lend themselves well into dimensionality reduction, thereby providing the system administrator with a summary of the multidimensional data on a two dimensional scale. From the field of Artificial Intelligence, rule-based algorithms are used especially in the misuse model, where the attacks are modeled as a set of rules and the data is checked against the models. From statistics, Chi-square and hypothesis testing are used along with other methods [**1**].

Testing an IDS: An Analysis of Available Data Sets

One of the first questions asked by those beginning their research in intrusion detection is how to create a data set suitable for developing and testing new intrusion detection algorithms.

In general, there are two ways to build and test an IDS. The first is to create your own simulation network, and collect relevant data. This method has several shortcomings. First, it is often difficult to create a simulated network environment with hundreds of computers, as may be required. For example, to simulate a Distributed Denial-of-Service (DDoS) attack may require thousands of computer nodes. Second, it is not possible to compare the results obtained through a simulated network environment with other results obtained using a different environment. Third, simulated network environments may yield traffic that does not mimic that of a real network due to the inherent nature of simulated traffic.

Another way to build and test an IDS is to use previously collected data sets. Although many organizations and corporations have been subject to computer attacks and have stored the data sets containing these intrusion incidents, in practice, there are several factors that stop these organizations from making this data available to the public for research purposes. Issues like privacy, security, and completeness greatly restrict organizations from providing this data to the public. Consequently, research organizations such as DARPA have created data sets that include real attacks on real networks for research purposes. The beauty of using previously collected data sets is that the results can be compared with others in the literature. Two of the more popular data sets are described in the following sections, namely, the DARPA Evaluation data sets [**14**] and the Knowledge Discovery and Data Mining (KDD) CUP 1999 competition data set [**17**].

DARPA 1998 Data Set

This data set contains seven weeks of training data and two weeks of testing data. Information about when a specific attack started and ended is provided for the training data only such that the performance of a given algorithm is evaluated during training phase. The simulated network represents thousands of UNIX hosts and hundreds of users. There are three UNIX machines designated as victim machines running three different operating systems: SunOS, Solaris OS, and Linux. A critical review of the DARPA data sets is presented by McHugh [**9**]. **Figure 1** shows an overview of the network used to create the data sets.

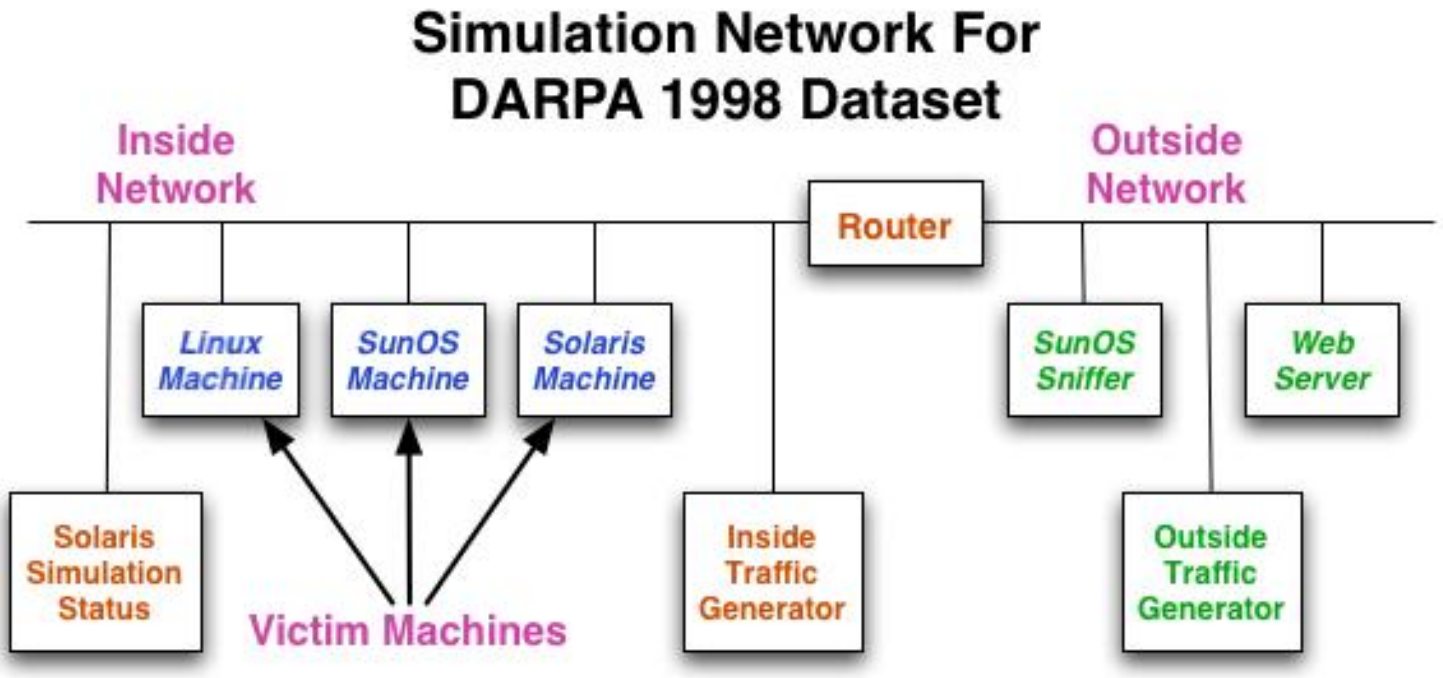

Figure 1: Simulation Network for the DAPRA 1998 dataset.

The data set contains more than three hundred instances of thirty eight different attacks from four attack categories. The four attack categories are shown in **Table 1**. Out of the thirty eight attacks, only fourteen were present in test data.

Four kinds of data were collected:

- TCP packets: Network level information
- BSM audit data: Host level information for Solaris OS
- File system dumps: Copies of system directories (generated every day)
- PS output: Lists of running processes (generated every minute)

With the above information, attacks could be detected at both the host and network levels.

Attack Category	Probing Attacks	Denial of Service (DoS)	User to Root (U2R)	Remote to Local (R2L)
Attack Description	Attacks that can automatically scan a network of computers to gather information or find known vulnerabilities.	Excessive consumption of resources that denies legitimate requests from legal users on the system.	Successful execution of attacks results in normal user getting root privileges.	Attacker having no account gains a legal user account on the victim machine by sending packets over the network.
Attack Example	IPsweep, Saint, and Satan.	DDoS, Pingflood, SYN flood, Mailbomb, and Process Table.	Eject, Fdformat, Loadmodule, and Perl.	Dictionary, FTP-write, Sendmail, and Xlock.

Table 1: Four attack categories in the DARPA 1998 data set.

KDD Data Set

The KDD CUP'99 data set was built using TCP packets collected during DARPA 1998 intrusion detection evaluation program. Data packets that form a complete session are

gathered in a single feature vector or connection record. Data mining techniques and frequently occurring patterns were used to identify features to detect various attack categories. KDD records have forty one features. There are four kinds of features present in KDD data set, namely, basic, content, time-based, and host-based features.

Basic or **intrinsic features** are common to every network connection, like duration of connection, service requested, and bytes transferred between source and destination machine address.

Content features were collected using domain knowledge of U2R and R2L attacks because these attack categories do not contain any frequently occurring patterns, for example, logged in flag, number of failed logins, number of root commands, number of compromised conditions, and hot indicators. Basic and content features could be used to detect U2R and R2L attacks.

Time-based features were collected by observing various connections within a two-second time window with respect to current connection. Features include SYN error rates, rejection rates, and number of different services requested. These features, combined with basic features, could be used to detect DoS and fast probing attacks with the two second time window.

Host-based features include the many slow probing attacks that require several minutes to execute. For such attacks, host-based features were collected that were based on the past one hundred connections similar to the one under consideration. These features were similar to time-based features.

DARPA 1999 Data Set

The DARPA 1999 data set is an extension to the DARPA 1998 intrusion detection evaluation program. In this data set, two new victim machines running the Windows NT operating system were added to the network. In addition, many new attacks were introduced, including a new attack category called data compromised. Both inside and outside attacks were logged in this data set. Instead of file system dumps, this data set contained specific file system information such as changed files, inode information, and important log files. Windows NT audit data was also provided as part of the data set. Out of three weeks of training data, only one week had attacks embedded. Similar to its predecessor, this data set includes two weeks of testing data.

What Metrics are Used to Compare the Performance of an IDS?

Several metrics are used to evaluate and compare the performance of IDSs. The most basic metrics are the detection and false alarm rates. The detection rate is equal tothe number of intrusions detected divided by the total number of intrusions in a data set, while the false alarm rate is equal to the number of normal instances detected as intrusions divided by the number of normal instances in a data set. False alarms are also referred to as **false positives**. Another method commonly used to evaluate and compare the performance of IDSs is the Receiver Operator Characteristic (ROC) curve. This is a graph with the false alarm rate on the x-axis and the detection rate on the y-axis. The optimal performance point is the one that achieves highest detection rate value with the lowest false positive rate for a specific system parameter setting.

Selecting a Platform for Developing and Evaluating an IDS

In in the early stages of developing an IDS, researchers are typically confronted by the question of which platform should be used to develop and evaluate their code. Knowing that the development of an IDS can involve statistical methods, especially multivariate statistics, machine learning methods, mathematical modeling, visualization, and a programming environment, it is best if they could find a single platform to work within. For this purpose, a freely available statistical analysis and visualization software package called R [**18**] could be used. R is an open-source implementation of the S language developed initially by Bell Labs. A student version of its commercial counterpart product S-Plus, developed by Insightful Corporation [**16**], is freely available for students and has a powerful graphical user interface (GUI), in addition to the full capabilities of command line and script processing.

The S Language is a powerful tool for the statistical and graphical analysis of data. It provides tools to implement many statistical ideas that have been made possible by the widespread availability of workstations with advanced graphics and computational capabilities. The tools available in S allow for the implementation of many known algorithms from statistics, artificial intelligence, and mathematics with advanced visualization and graphics.

In addition, the S Language provides a flexible and powerful environment in which to implement new statistical ideas. This integrated suite of software facilities for data analysis and graphical display allows for the creation of an extensive and coherent collection of tools. S has a language for expressing statistical models and provides graphical facilities for data analysis. Being an object-oriented programming language, S allows the user community to extend the capabilities of their models [**12**]. In addition to R, other modeling languages such as Matlab may be used to implement algorithms that rely primarily on

visualization.

Further Reading: Institutions Actively Researching IDSs

Several organizations are active in the research in Intrusion Detection. The Computer Emergency Response Team (CERT) [**13**] at Carnegie Mellon University collects information about computer attacks and releases public advisories describing the nature of the attacks and how to protect yourself against such attacks. The advisories are typically practical advice in applying system patches and better configuring systems to be more secure. A mailing list called Bugtraq is possibly one of the most valuable sources of information covering security vulnerabilities and defenses against them. The mailing list discusses details of computer security vulnerabilities, including what these vulnerabilities are, how to exploit them, and how to fix them. There are several conferences that primarily address issues in computer security. Some of these conferences are targeted towards individuals who are involved directly with analyzing and finding solutions for known vulnerabilities and are hands-on. A notable example of these conferences is DefCon [**15**]. There are other conferences that have a more corporate nature, but still provide valuable information about computer security. The System Administration, Networking, and Security (SANS) [**19**] organization holds several conferences each year that offer detailed training to systems administrators on how to build and deploy secure systems and networks. In addition, there are several web sites that provide valuable news and technical discussions covering security issues. Examples of these include the Security Focus and the Packetstorm web sites.

Conclusion

Computer attacks happen every day. The operating systems, software, and networking stacks we use have many vulnerabilities, some of which are intrinsic to them. Attackers scan our systems and networks for these vulnerabilities and break into the system, eventually getting access to private data, such as bank accounts and credit card information. Attackers also use the machines they break into to launch new attacks on other machines. IDSs were designed to detect and respond to intrusions and alert security officers to any possible attack on networks and systems. There are several models for intrusion detection, including signature-based, anomaly-based, and specification-based intrusion detection systems, where each model has its own set of merits and demerits. These intrusion detection models are implemented through a wide variety of algorithms and methods spanning several disciplines such as statistics, computer science, artificial intelligence, soft computing, and visualization. There are only a few data sets that can be used for developing and testing IDSs. There are primarily two methods to evaluate and compare the performance of an IDS. The first are the false-alarm rates and detection-rates

and the second is the ROC curves. A platform for developing and evaluating an IDS is generally sought for the initial work in intrusion detection. Using the free open-source R language, an implementation of the S Language developed by Bell Laboratories, is proposed for developing new methods for intrusion detection. This platform includes libraries for most algorithms and methods used in statistics and soft computing, in addition to having advanced visualization capabilities. There are several conferences, organizations, mailing lists, and web sites that discuss many aspects of computer security and provide the user community with up-to-date news on the latest vulnerabilities and how to protect against them.

References

1

Allen, J. et al. "State of The Practice: Intrusion Detection Technologies". Carnegie Mellon, SEI, Tech. Report CMU/SEI-99-TR-028, ESC-99-028, January 2000.

2

Axelsson, S. "Intrusion Detection Systems: A Survey and Taxonomy." Technical report 99-15, Dept. of Computer Engineering, Chalmers University of Technology, Goteborg, Sweden, March 2000.

3

Balepin, I., Maltsev, S., Rowe, J., and Levitt, K. "Using Specification-Based Intrusion Detection for Automated Response." *Proceeding of the 6th International Symposium*, RAID 2003, Recent Advances in Intrusion Detection, Pittsburgh, PA, September 8-10, 2003.

4

Cabrera, J., Ravichandran, B., and Mehra, R. "Statistical Traffic Modeling for Network Intrusion Detection." *Proceedings of the Eighth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, Aug. 2000.

5

Haykin, S. "Neural Networks: A Comprehensive Foundation." Second Edition. Prentice Hall Inc., 1999.

6

Hotelling, H. "Analysis of a Complex of Statistical Variables into Principal Components." *Journal of Educational Psychology*, 24:417-441, 1933.

7

Kaufman, L. and Rousseeuw, P. J. "Finding Groups in Data: An Introduction to Cluster Analysis." New York: 1990 John Wiley & Sons, Inc.

8

Kohonen, T. "Self-Organizing Maps." New York, Springer-Verlag, 1995.

9

McHugh, J. "Testing Intrusion Detection Systems: Critique of the 1998 DARPA

Intrusion Detection System Evaluations as Performed by Lincoln Laboratory." *ACM Transactions on Information and System Security*, Vol. 3, No. 4, November 2000, Pages 262-294.

10 Skoudis, E. "Counter Hack: A Step-by-Step Guide to Computer Attacks and Effective Defenses." Prentice Hall Inc., 2002.

11 Shah, H., Undercoffer J., and Joshi, A. "Fuzzy Clustering for Intrusion Detection." *FUZZ-IEEE*, 2003.

12 Venables, W. N., Ripley, B. D. "Modern Applied Statistics with S." Fourth Edition, Springer-Verlag, 2002.

13 The Computer Emergency Response Team: **http://www.cert.org/**

14 DARPA Intrusion Detection Evaluation Project: **http://www.ll.mit.edu/IST/ideval/**

15 The DefCon Conference web site: **http://www.defcon.com/**

16 Insightful Corporation: **http://www.insightful.com/**

17 Knowledge Discovery and Data mining (KDD) CUP 1999 competition: **http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html**

18 The R Project web site: **http://www.r-project.org/**

19 The System Administration, Networking and Security Organization: **http://www.sans.org/**

Biography

Khaled Labib (**kmlabib@ucdavis.edu**) is a Ph.D. candidate at the Department of Applied Science, University of California, Davis. Khaled's main research interests are in the fields of Network Security, Intrusion Detection, and Soft Computing. In his spare time, Khaled enjoys playing soccer, reading, and spending time with his family.