STATISTICAL METHODS USED FOR INTRUSION DETECTION

A Thesis Submitted to
The Graduate School of Engineering and Sciences of
İzmir Institute of Technology
in Partial Fulfillment of the Requrements for the Degree of

MASTER OF SCIENCE

in Computer Software

by
Onur ÖZARDIÇ

July 2006
İZMİR

We approve the thesis of **Onur ÖZARDIÇ**

Date of Signature

..... **11 July 2006**
Prof. Dr. Halis PÜSKÜLCÜ
Supervisor
Department of Computer Engineering
İzmir Institute of Technology

..... **11 July 2006**
Asst. Prof. Dr. Tuğkan TUĞLULAR
Co-Supervisor
Department of Computer Engineering
İzmir Institute of Technology

..... **11 July 2006**
Prof. Dr. Sıtkı AYTAÇ
Department of Computer Engineering
İzmir Institute of Technology

..... **11 July 2006**
Assoc. Prof. Dr. Ahmet KOLTUKSUZ
Department of Computer Engineering
İzmir Institute of Technology

..... **11 July 2006**
Prof. Dr. Şaban EREN
Department of Computer Engineering
Ege University

..... **11 July 2006**
Prof. Dr. Kayhan ERCİYEŞ
Head of Department
Department of Computer Engineering
İzmir Institute of Technology

.....
Assoc. Prof. Dr. Semahat ÖZDEMİR
Head of the Graduate School

ACKNOWLEDGEMENTS

I would like to express sincere gratitude to my advisors Prof. Dr. Halis Püskülcü and Asst. Prof. Dr. Tuğkan Tuğlular for their encouragement, guidance and support through the development of this thesis.

I also would like thank members of my thesis committee, Prof. Dr. Sıtkı Aytaç, Prof. Dr. Şaban Eren, Assoc. Prof. Dr. Ahmet Koltuksuz and Prof. Dr. Kayhan Erciyeş, Head of Computer Engineering Department.

I also would like to thank friends Hüseyin Hışıl, Burak Galip Aslan, Selma Tekir and Oğuzhan Arslan for their help and support.

Finally, I owe my special thanks to my family for their support, encouragements and patience

ABSTRACT

STATISTICAL METHODS USED FOR INTRUSION DETECTION

Computer networks are being attacked everyday. Intrusion detection systems are used to detect and reduce effects of these attacks. Signature based intrusion detection systems can only identify known attacks and are ineffective against novel and unknown attacks. Intrusion detection using anomaly detection aims to detect unknown attacks and there exist algorithms developed for this goal. In this study, performance of five anomaly detection algorithms and a signature based intrusion detection system is demonstrated on synthetic and real data sets. A portion of attacks are detected using Snort and SPADE algorithms. PHAD and other algorithms could not detect considerable portion of the attacks in tests due to lack of sufficiently long enough training data .

ÖZET

SALDIRI TESPİTİ İÇİN İSTATİSTİKSEL YÖNTEMLERİN KULLANIMI

Her gün bilgisayar ağlarına yönelik saldırılar gerçekleşmektedir. Saldırı tespit sistemleri bu saldırıları tespit edip etkilerini azaltmak için kullanılmaktadır. İmza temelli saldırı tespit sistemleri, sadece bilinen saldırıları tanımlayabilmekte, bilinmeyen ve yeni saldırılar karşısında etkisiz kalmaktadır. Anormallik tespiti ile saldırı tespiti yöntemleri bilinmeyen saldırıları tespit etmeyi hedeflemektedir ve bu amaca yönelik geliştirilmiş algoritmalar mevcuttur. Bu çalışmada beş anormallik tespiti algoritması ve imza tabanlı bir saldırı tespit sistemi olan Snort'un, sentetik ve gerçek veri kümeleri üzerinde test edilip başarımlarının gösterilmesi hedeflenmiştir. Snort ve SPADE algoritmaları kullanılarak saldırıların bir bölümü tespit edilebilmiştir. PHAD ve diğer algoritmalarda ise testlerde yeteri kadar uzun eğitim verisi olmaması sebebiyle saldırıların önemli bir bölümü tespit edilememiştir.

TABLE OF CONTENTS

LIST OF FIGURES

Figure **Page**

LIST OF TABLES

Table **Page**

CHAPTER 1

INTRODUCTION

Computers and Internet has become an ordinary and indispensable reality of life for many people. This trend makes people use facilities on-line with an increasing rate. This widespread usage has made Internet a new market for enterprises, a place to share and exchange information for researchers, source of entertainment and recreation. Widespread use and benefits of online resources have also attracted people who would like to benefit more than others with use of illegal methods. These people have exploited vulnerabilities in systems sometimes for benefit, sometimes for satisfying their curiosity only. In 1988, Morris Worm (WEB_1 2006) epidemic have caused to stop 10% of servers connected to Internet. Even though size and use of Internet was small at the time, economic and social impact was greater. Morris worm may be thought as a starting point for a new era. Attacks increased by time and this increase introduction of security mechanisms, precautions and development of software patches to remove vulnerability. Increased security precautions introduced new attack methods and exploitation of new vulnerabilities. Loop of improved security precautions and attacks Figure 1.1 shows number of reported incidents to CERT/CC between 1994 and 2003 (WEB_2 2006).

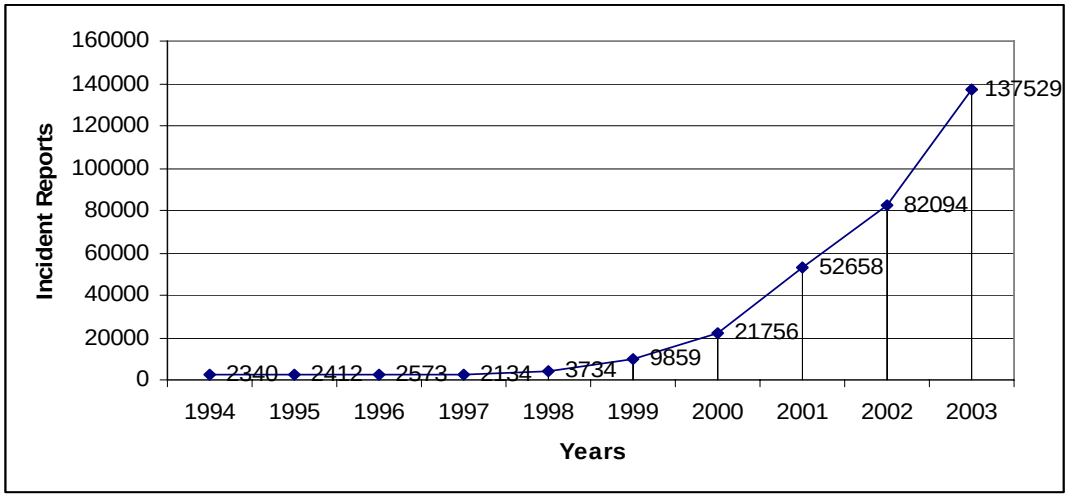

Fig.1.1. CERT/CC reported incidents by year

According to an annual survey (CSI 2005), total financial loss due to security incidents have been around $130M in 2005. 95% of the participants have experienced more than 10 web site incidents last year. Moreover, a majority of 73% has not reported the incident.

On the other hand, use of some of basic security tools has become very common, such as firewalls (97%), anti virus software (96%) and intrusion detection systems (72%) according to the report. Firewalls and anti virus software are applied as first line of defense against external attacks. Deploying and running intrusion detection systems may be considered as second line of defense, but it is obvious that they are also a popular solution, but not widely accepted.

Attacks which exploit recently discovered vulnerabilities have more impact than older attacks, since a software patch is released by software vendors to remove for a known attack and is applied by security-aware administrators. Unknown attacks have more chance on defeating deployed security solutions. Traditional systems usually fail on detecting unknown attacks, since their success rates heavily rely on description of attacks. One way to describe or detect them is using anomalies happened on the systems. These undiscovered attacks can create anomalies in systems. If the anomalies caused by attacks can be discovered when they occurred; precautions may be taken much earlier before attack becomes widely known. Detecting, or at least being informed about unknown attacks provide a significant increase in security. Anomaly detection algorithms may help discovering attacks using anomalies due to the attacks on the system, without describing them and increasing overall security.

Uses of anomaly detection for detecting hostile activity have been studied for a long time. There are anomaly detection algorithms and tools developed for detecting intrusions, both in commercial and academic research sites. However, commercial products are not tested in laboratory environment with scientific metrics. Even though result of anomaly detection algorithms sounds promising environment, most of them are not tried in harsh environment of real networks. Lack of being tested in real environments may be misleading for detection performance. For commercial products, even though there exists evaluations, but almost none of these make sense in a scientific way. This is a handicap for revealing real performance of commercial systems. In this study a set of anomaly detection algorithms and commercial products were tested using

synthetic and genuine real time datasets and results are discussed with comparison of their performance.

Remainder of this dissertation is as follows, starting from Chapter 2:

In Chapter 2, concepts of security, intrusion, intrusion detection and anomaly detection are described with their goals. In addition, tools and evaluation material are described.

In Chapter 3, algorithms that are to be evaluated in this study are introduced with their background.

In Chapter 4, introduces process of data collection operation from Izmir Institute of Technology campus servers.

In Chapter 5, processes of evaluation of algorithms are explained with sufficient detail of evaluation environment, configuration and modifications and operation.

In Chapter 6, findings on collected data are introduced. Similarities and significant differences between available datasets and other datasets are discussed in detail. Results of demonstration process are compared and discussed with each other and previous studies.

Chapter 7 summarizes our main findings and advantages and disadvantages of described algorithms in practice. It also points out our limitations and further research issues.

CHAPTER 2

INTRUSION DETECTION

2.1. Overview

Computers and computer networks have become quite common in use for people in modern world. Many services and information sources provided online such as online encyclopedias, informational web-pages, directory listings, newsgroups, email, and online shopping sites are accessed and extensively used by millions of people everyday. However, not every user wishes to benefit from these sites and services, there are people who try to abuse people, services and enterprises.

2.2. Definition and Goals

This is the point where computer security concept is introduced. Computer security may be defined as "generic name for collection of tools designed to protect computer systems" (Stallings 2003). It consists of three characteristic properties of the system to be protected. These three goals are defined as (Pfleeger 1997):

Confidentiality: Confidentiality means the assets of a computing system are accessible only by authorized parties. This asset may contain any information not only limited to data and permissions, but also may contain existence of a fact about system or the data on the system. This term is also referred to as privacy or secrecy. In further parts of the text these terms are used interchangeably.

Integrity: Integrity means that assets can be modified by only authorized parties or only in authorized ways.

Availability: Availability means that assets are accessible to authorized parties. Availability of an object (or service) includes its presence, capacity to meet service needs, bounded waiting times, timeliness of service.

An attack or intrusion may be defined as any set of actions that attempt to compromise confidentiality, integrity and availability of a resource (Heberlein et

al.1991), which may be simply stated as violation of the three goals. Sometimes the term penetration is used instead of intrusion.
Attackers, who attack on computer systems, are also referred to as intruders. Attacker and intruders are also used interchangeably. However not all attackers are humans, viruses and worms may be accepted as attacker, since their goals and activities are similar to their human counterparts.

Intrusion Detection is, in its simplest form, a set of tools, methods and activities to detect violations of security goals. Intrusion Detection System will gather information from monitored system or network and provide information to human analyst, about suspicious activity, which also may include intrusions. Intrusion Detection is interested detecting intrusions and being aware. Intrusion Prevention extends Intrusion Detection and includes countermeasures against attackers and their activities.

2.2.1. Taxonomy of Attacks

All attacks are not the same; they may be classified into five main groups according to targeted security goals. (Lippmann et al. 2000b) describes the taxonomy of attacks:

Probes: These attacks automatically scan a network of computers or a DNS server to find valid IP addresses, active ports, host operating system types, and known vulnerabilities.

Denial of Service Attacks: Denial of Service (DoS) attacks are designed to disrupt availability of a host or network service.

Remote to Local Attacks: On a Remote to Local (R2L) attack, an attacker who does not have an account on a victim machine gains local access to the machine, exfiltrates files from the machine, or modifies data in transit to the machine.

User to Root Attacks: User to Root (U2R) attacks where a local user on a machine
is able to obtain privileges normally reserved for system administrators.

Data Attacks: goal of a Data attack is to exfiltrate special files which the security policy specifies should remain on the victim hosts.

Probe attacks are more widely applied than other methods, because of their reconnaissance-nature activities on potential victims.

2.2.2. Taxonomy of Intrusion Detection Systems

Intrusion detection systems are classified into groups according to their data sources, methods of detection and response times.

2.2.2.1. Classification of Intrusion Detection Systems by Data Source

These systems use different data sources and usually installed on different locations to operate:

2.2.2.1.1. Host Based Intrusion Detection Systems

Host based intrusion detection system (HIDS) works on a single host, using different sources of information such as security audit logs, event logs, file hashes, registry traces etc. Application based intrusion detection systems are a special subset of this group, since they use host available only to the host they are installed and an data source is an application installed on that system, other than data sources provided by operating system facilities.

2.2.2.1.2. Network Based Intrusion Detection Systems

Network based intrusion detection systems (NIDS) works by analyzing network activity on the network. In its simplest forms a network tap, which is usually another computer, is installed and all network activity passing, inbound, outbound or both directions is logged or analyzed. Most commercial systems fall into this category.

There is also a subset of network based intrusion detection, named Network Node Intrusion Detection. (Crothers 2003) describes Network Node Intrusion Detection using its working principle: “This system works by analyzing network traffic like standard network based intrusion detection does. But rather than attempting to monitor all network traffic, a network node IDS analyzes only network traffic specified for it”.

2.2.2.2. Classification of Intrusion Detection Systems by Method of Detection

Intrusion Detection Systems may be classified according to the methods used for detecting potential malicious activity. Two common and general methods exist, and have different strong and weak sides.

2.2.2.2.1. Signature Based Intrusion Detection

Signature based Intrusion detection systems rely on a set of rules (also known as signatures) for detecting intrusion activity. A signature can be described as a conditional rule, which is tested on an instance of activity, identifying a specific type (Cole et al. 2005). This instance may be an incoming network packet, streaming traffic flow, specific set of keywords or activities on a monitored system, lines of log records and lists of commands or sequence of system calls. For example, in the network packet case, an incoming packet to a network IDS may be checked against a set of rules for matching content. Matching operation usually includes comparisons of binary or text data using regular expressions. A signature can be used to detect intrusions or policy violations where applicable. Signature based detection may help previously known and modeled attacks. For undiscovered attacks, which is also known as “zero-day attacks”, they are usually useless.

2.2.2.2.2. Anomaly Detection

Anomaly detection is not opposite of signature based detection, a supportive complement instead. Anomaly detection creates a norm model characterization for monitored activities using acquired data, such as connection durations, incoming or outgoing traffic rates, frequency of commands etc. Any event which is identified as deviant or anomalous according to this model is stamped as hostile or anomalous (Cole et al. 2005). This method can be used to detect not only known attacks, but also promises to detect unknown attacks. This scheme may be useful for detecting unknown attacks and to achieve a lower false negative rate. Anomaly detection systems may not

provide much information about attacks or their nature, since they are identified by their anomalous nature, not a signature.

2.2.2.3. Classification by Processing and Response Time

A real time intrusion detection system can process data, determine existence of a suspected attack and respond or report in real time. This is most common approach and preferred, but not the only one. Another approach, off-line processing, is determining whether an attack was made in a separate time, probably using a different system. This may be preferred in cases of manual inspection, aggregation with other data, or existence of post-processing on data.

2.2.3. Background on Anomaly Detection

An anomaly is something unusual, unexpected or a form of deviation from a pre-defined standard rule, theory or measurement (WEB_3 2006). It may be an event happened or a value measured by some sort of experiment or device. Existence and definition of anomaly requires existence of some standard, which may be built on conventions or experiments. A snow storm between two sunny days in summer is possible, but if it happens, this event can be considered as anomalous.

2.2.3.1. Norm

Concept of anomaly rises from standards and expectations. Measurable quantities of events can be used to construct models and calculate statistics. Probability distribution of commands on a remote login connection of a home-office worker of some company, or distribution of some measurable information transferred on page requests from a server may be measured and a norm, a standard may be formed (Stallings 2003).

2.2.3.2. Attack

Deviation from a norm may be coincidental event; it may be tolerable up to some threshold level. Home-Office user may start using different commands with different

frequency, or page requests for server are significantly changed for some measurable time. These events may be signs of hostile activity, because of their deviations from norm, or vice versa. An anomaly may point a hostile activity. The old and the new activity profiles may overlap or may differ significantly. Deviations from the norm profile may be detected and can be signed as hostile (Stallings 2003).

2.2.4. Criteria on Evaluating Performance of Intrusion Detection Systems

Success rate of an intrusion detection system depends on a few criteria, each of which is not sufficient to be authoritative.

High detection rate is the most important goal of an intrusion detection system. A good system should have high rates of detection with a relative false alarm rate. False alarm rate becomes a limiting factor (Axelsson 1999).

Intrusion detection systems should bring minimal computational overhead to the systems which they are run. Computational overhead can affect overall performance of system. In network based systems case, high load on NIDS increases rate of dropped packets, which will result as lowered detection rates.

An intrusion detection system should be able to identify an attack with high rate of accuracy. It also should be able to inform analysts about success and possible losses, such as compromised systems or data.

An intrusion detection system should be able to correlate different events and alerts to provide a big picture of overall attack, such as existence of a distributed attacks or disguising attackers.

Attackers and attack methods evolve as intrusion detection technology evolves. Development of new attacks and evasion techniques increases false negative rate of systems. An ideal system should be able to detect unknown attacks. Signature based systems usually fails this criteria, whereas anomaly detection scheme is considered more favorable.

2.3. Related Work

Intrusion detection has become a daily reality for many enterprise network system managers. Rise of Internet's user base around the world, people joining to cyber world has brought risks for organizations. Intrusion detection aims to address these risks.

Concept of intrusion detection, terms of audit trails and user activity have emerged by (Anderson 1980). Anderson actually wrote the report for a government organization and suggested using computer audit trails to understand user behavior and detect computer misuses.

Most commonly deployed commercial systems are signature based over the world. Heuristic approaches are also used in commercial products but did not become either common or popular. There are intrusion detection algorithms and systems developed previously, some of which date before 1999. These relatively older and fundamental systems are discussed in (Axelsson 2000).

2.3.1. Signature Based Approaches

Most widely known of signature based intrusion detection system is Snort (Roesch 1999) and Bro (Paxson 1998), looking for attack signatures on monitored traffic. Snort is used to test one of the algorithms in integration with signature detection and is introduced in Section 2.6.

SHADOW (WEB_4 2006) is another intrusion detection system, provided freeware, is a set of useful scripts and programs. SHADOW has a manual and offline approach on intrusion detection, since system generates hourly reports of suspected traffic, to be inspected by analysts. It has been introduced in textbooks for educational purposes until Snort became widely available.

2.3.2. Learning Based Detection Techniques

Second part of intrusion detection realm, statistical based intrusion detection, is formalized in (Denning 1987). The model described includes signature detection, use of statistical moments etc.

(Cabrera et al. 2000) has used Kolmogorov-Smirnov test to detect DoS and probing attacks in addition to detecting attack telnet traffic with attack patterns are statistically different from ordinary connections. They used DARPA 1998 evaluation data.

Statistical Packet Anomaly Detection Engine (SPADE) (Hoagland 2000) examines network and transport layer data for rare and anomalous events. SPADE algorithm is introduced in 3.5 and evaluated in this study in integration with Snort.

(Bykova et al.2001) has tested anomalous activity for protocol specifications, such as packet headers and allowed values, address spaces and their possible reasons for intrusions. This type of anomaly detection is also called as strict anomaly detection.

(Ye and Chen 2001, Ye et al. 2001, Ye et al. 2002) have used various univariate and multivariate statistical tests to detect R2L attacks in IDEVAL data using BSM module logs of Solaris operating system.

Mahoney and Chan have developed a set of anomaly detection algorithms named Packet Header Anomaly Detection (PHAD) (Mahoney and Chan 2001), Application Layer Anomaly Detection (ALAD) (Mahoney and Chan 2002a), Learning Rules for Anomaly Detection (LERAD) (Mahoney and Chan 2002b) and Network Traffic Anomaly Detector (NETAD) (Mahoney and Chan 2003c). These algorithms used time based modeling and protocol modeling.

(Aydin and Orencik 2005) has improved PHAD's anomaly detection capability using different values for time factor of PHAD and used this modified algorithm in integration with Snort. (Yin et al. 2005) have used genetic programming based rule learning approach on LERAD to improve performance of LERAD.

(Kruegel and Vigna 2003) has used an anomaly detection algorithm on web server request query attributes, calculating anomaly scores which are derived from probability vales associated with query attributes.

(Shon et al. 2005) have used genetic algorithms search technique to select features and support vector machine (SVM) machine learning methods to detect intrusions, using different data sources.

2.4. IDEVAL Data Set

Evaluating performance of an intrusion detection system or an algorithm in terms of detecting and missing intrusions is not an easy task. Even though it is ideal to test systems in a real environment, this method has its own problems such as privacy of communication between peers, repeatability of events and uncontrollable nature of events, existence of non-typical traffic or availability of unknown attack methods.

DARPA/MIT Lincoln Labs. Intrusion Detection Systems Evaluation Data Set (IDEVAL) is one of the leading tools for evaluating measuring and comparing performance of intrusion detection systems. Offline datasets contain synthetic background traffic and labeled attacks. There are two offline evaluation datasets: 1998 and 1999.

The DARPA 1998 Intrusion Detection Evaluation was an initial attempt to perform a comprehensive evaluation of intrusion detection technology. It was designed to evaluate only DARPA funded intrusion detection technology and not complete deployable intrusion detection systems (Lippmann et al. 2000b).

1999 Evaluation and offline datasets have significant improvements on 1998 evaluation in terms of network structure, evaluation scoring criteria and attack patterns. All datasets are published and can be downloaded from (WEB_5, 2006).

2.4.1. Design of IDEVAL Simulation Environment

There are three major design principles of the evaluation. These are:

1) A Standalone network testbed was used to generate simulation environment
2) Intrusion Detection Systems' performance was measured using both attack detection rate and false alarm rate
3) An offline evaluation format allowed many systems to be evaluated and supported intrusion detection researchers with examples of background traffic (Haines et al. 2001).

1998 simulation environment aimed to simulate live traffic similar to traffic that flows between the inside and outside Internet of a United States Air Force Base. In 1998 evaluation there are 3 victim UNIX hosts running Linux, Sun OS and Solaris operating systems, providing different services to users in and out of the base. 1999 evaluation

one more server is added to testbed network, which runs Windows NT. There are two sniffers on both side of the router, recording all traffic coming to router from the side where it is located.

For evaluation, file system dumps are collected in addition to audit data collected from Solaris and Windows NT servers. Structure of the testbed network is simple enough to demonstrate capabilities of the technology. (Lippmann et al. 2000a) declares that this network architecture is not representative of an Air Force base. It is a minimal network designed to support intrusion detection systems that desired to participate in 1998 and 1999, attack types of interest and most of the network traffic types seen across many Air Force bases (Lippmann et al. 2000a).

2.4.2. Traffic Generation

There are two types of traffic simulated in the network: normal background traffic and attack patterns which are expected to be detected.

Normal background traffic was statistically compatible with traffic of an Air Force base in terms of people's web and other habits, used words in mails, traffic types and rates, frequency of words in documents. These statistics were collected from more than fifty bases were included.

There are two traffic generators in both inside and outside internets. Generator outside simulates thousands of sites and workstations accessed by clients in the base. Inside traffic generator simulates users inside the base with different activity profiles. These two generators have specific implementations in operating system kernel developed and tuned for the simulation. Figure 2.1 shows network structure diagram of network used for evaluation.

Attacks were executed by automated scripts, collected from different sources on the Internet. Some of exploits are specifically modified to evade detection (Das 2000).

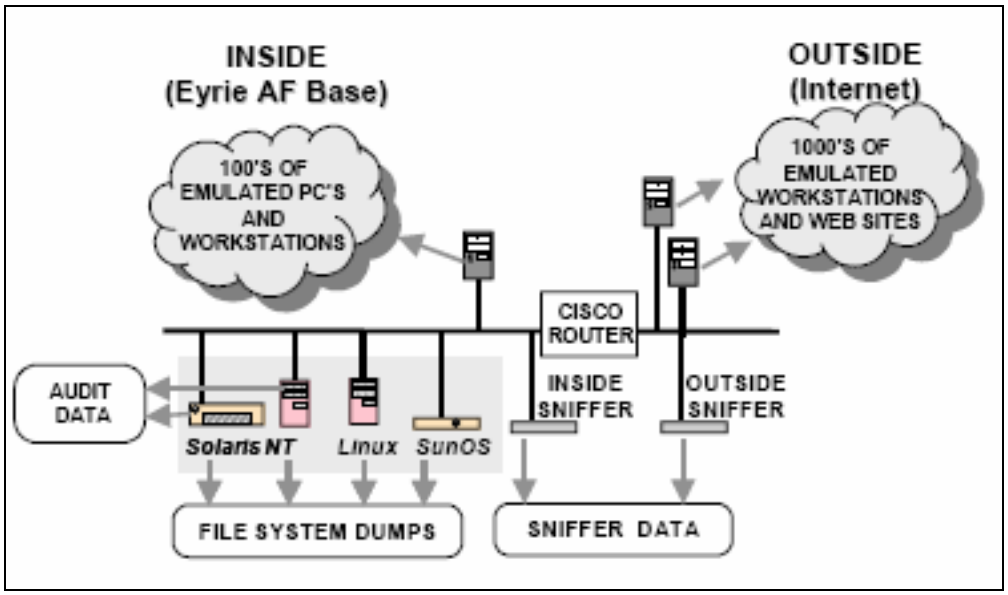

Fig. 2.1.Network structure diagram of 1999 evaluation
(Source: Haines et al. 2001)

2.4.3. Attack Scenarios

1999 Evaluation Data Set consists of five weeks of data. First three weeks are training data, where fourth and fifth are provided for testing purposes. First and third weeks' traffic is attack free, but second week has some labeled attacks. Fourth and fifth weeks contain attacks in different numbers, times and diversity. Table 2.1 shows names, types, targets and numbers of attacks. Numbers in parentheses next to attack types are number of attack instances which fall into that category. Numbers in cells are number of applicable attacks which fall into that category:

Table 2.1.Number and categorization of attacks according to attack types, victims and number of instances of each attack group (Summarized from original source: Haines et al. 2001)

	DoS(65)	**Probe(37)**	**R2L(56)**	**U2R(37)**	**Data(13)**
Solaris	8	2	6	4	1
WindowsNT	4	2	5	4	1
Sun OS	6	2	2	1	
Linux	11	4	9	3	1
Cisco	1	2	1		
All OS		3			

Details of attacks and discussion about traces, signatures and effects are explained thoroughly in (Kendall 1999) and (Marchette 2001).

In the evaluation, two attacker profiles are considered:

First profile is an amateur, relatively unsophisticated, testing his/her skills and probably has no specific goals in mind. He uses attack scripts collected from different public sources on Internet. Actually these type of attackers are known as "script-kiddie" in security community. Second profile is a more professional one, probably a "black hat", with specific goals in mind and equipped with more skills, may attempt to gain some information from the system or deny service for some time. This experienced and skilled attacker has ability to modify existing attack scripts where needed, in addition to create attacks from scratch, and capability to evade detection by means of using time and his coding ability.

2.4.4. Critics and Discussion of IDEVAL Data Sets

(McHugh 2000) has published an article criticizing 1998 evaluation and dataset. He has focused criticism on 1998 evaluation. By the time 1999 evaluation was underway and some of the ambiguous points in evaluation have not been revealed then. Some of the ambiguous points have been revealed and problems discussed are addressed in (Das 2000) and (Haines et al. 2001).

McHugh criticizes evaluation and data set under following titles:

Goals: Consistency of achieving goals in evaluation

Background data: Content and generation, similarity of simulated and real background traffic.

Attack Data: Realistic distribution of attacks distributed among background traffic in both number and taxonomic classification. For example U2R attacks are the most common in evaluation however probes and DoS attacks are more common in real world.

Testbed Network: hypothetical air force base network domain (eyrie.af.mil) has ambiguities on number and properties of client hosts.

(Mahoney and Chan 2003b) have compared attack free data of 1999 evaluation data set network traffic properties with real network traffic collected from their

departmental servers in a few months period. They have discovered some simulation artifacts useful for testing anomaly detection algorithms.

These simulation artifacts, which may result on making accurate estimations on accuracy of anomaly detection algorithms, are summarized as follows:

Regularity in Simulation: Regularity and limited diversity of TCP options in simulation traffic

Diversity in Packet Header Fields: Packet fields such as Type of Service (TOS) and Time to Live (TTL) take more diverse values in real time traffic.

Crud Packets: There are packets observed where checksums are correct but some of protocol specifications violated, such as nonzero values in reserved fields.

HTTP Requests: HTTP Requests in IDEVAL are generally in "GET url HTTP/1.0" form, followed by optional commands and "keyword:value" pairs in first 200 bytes of first data packet. Diversity is limited in these keywords and commands in simulation.

SMTP Requests: Sessions in simulation are always beginning with HELO and EHLO with 3 and 24 different arguments respectively. In real traffic, more distinct arguments have been observed.

SSH Requests: Client version is only of one type in IDEVAL, real traffic traces have shown that this is not the case.

These measurements may be counted as a single observation; hence it is not a good idea to use these results as authoritative, but informative and insightful.

Another critic on IDEVAL data sets are based on self similarity of IDEVAL traffic rates has been made by (Allen and Marin 2003). Their study show that self similarity models of network traffic fails at night. More details of these two studies are explained in 2.5.

2.5. Modeling Behavior of Network Traffic

Modeling network traffic and its properties using existing statistical models have become an interest for people. There are models and technologies developed according to these models. Network activities were assumed to happen randomly, for sake of simplicity, for example a set of computers, all of which use the same medium as

common access channel, instance of transmission for a frame by these computers was to be modeled with Poisson distribution (Tanenbaum 1996). Another similar case is valid for modeling the process of arrivals, a Poisson process, which interarrival times of events are independent (Allen 1997). However empirical evidence by (Paxson and Floyd 1995, Leland et al. 1994) has shown that this is not be the exact case. Leland shows pictorial evidence of self similarity using five different time scales of self similarity of network traffic and compares with graphical models of Poisson distribution for the same scales. Self similarity means that an object looks roughly the same on any scale (WEB_6 2006). Fractals are of this type of objects. For indicating self similarity of an object, Hurst parameter (denoted as H) is used. Hurst parameter of an object is 1 is if it is completely self similar, 0.5 when it has a Poisson distribution. H=0 means the object is not self similar. Shown that network traffic is self similar, it also implies that events in network are not independent, revealing a long range dependency. Floyd and Paxson have verified these results, by denying Poisson modeling, calling for a new method of modeling. However they also show that telnet session arrivals can be modeled using Poisson.

In another study made by (Paxson and Floyd 97), discusses issues in creating a simulation of a network and network traffic. There exist many parameters to consider for creating such type of simulation but a few of them are explained there with possible coping strategies.

2.6. Snort Intrusion Detection System

Snort is a lightweight intrusion detection system developed by (Roesch 1999, WEB_7 2006). It is an evolving system publicly available with GNU General Public License. In early versions Snort had limited functionality on limited number of platforms, however now it is supported and used on more platforms with increased functionality and effectiveness.

In addition it has been quite popular in computer security community, due to its functionality, affordability (cost effective solution), widely applicable for various environment and purposes, and publicly available supportive material and community. Many textbooks on intrusion detection provide examples using Snort.

2.6.1. Structure and Operational Properties of Snort

Snort uses libpcap in three of its four modes. Libpcap (WEB_8 2006) a software library designed and used to capture packets from specified network interfaces on the system.

Snort has four operational modes which may serve different purposes (Sourcefire 2006):

- ***Sniffer*** mode, which is simply reading the packets from network and displaying them in a continuous stream on the console
- ***Packet Logger*** mode, which logs the packet contents to disk.
- ***Network Intrusion Detection System*** (NIDS) mode is the most complex and configurable configuration, which allows Snort to analyze network traffic for matches against a user-defined rule set and performing several actions
- ***Inline*** Mode: In this mode, Snort receives packets from iptables (WEB_9 2006) instead of libpcap interface and then causes iptables to drop or pass packets based on Snort rules that use inline-specific rule types. Iptables is a mechanism and tool used to modify packet filtering mechanism of Linux kernel.

2.6.1.1. Snort Rules

Rules as detection technique, is both the strong and the weak point of Snort. A simple Snort rule consists of two logical sections : “rule header” and “rule options”:

Header section consists of four parts:

- action (alert, log, etc.),
- protocol (tcp, udp, etc.),
- source and destination (in terms of IP address and port)

Rule options section specifies descriptive features on hostile packets such as portion of application payload content or some fields in packet headers. A simple rule for a fictional attack is given in Fig. 2.2 below.

If this rule holds, alert will be triggered as ana action. Hostile packet may come from any IP address and any source port. Destination IP block is defined as 10.0.0.0/24, C class network and destination port is specified as 9999.

Rule options define that application payload content contains a specific byte string. Last part defines informational message on hostile activity. Snort can also be used to

detect weird activities in addition to detecting security and usage policy violations on networks.

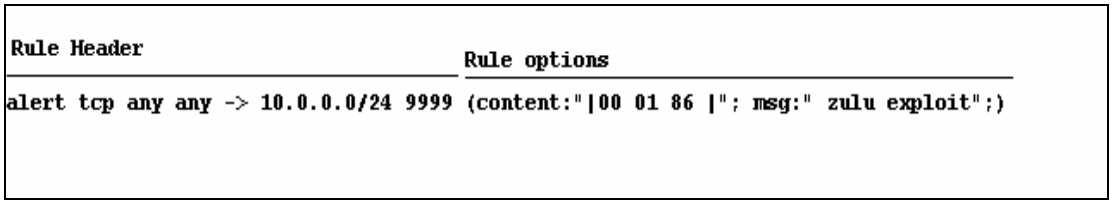

Fig. 2.2. A simple Snort rule

2.6.1.2. Snort Rule Chain

Network Intrusion Detection mode of Snort, tries to detect intrusions on a rule-matching basis. Rules are stored and used as a two dimensional linked lists, which connect chain headers and chain options. Common attributes of a set of rules are represented as chain headers, aiming to increase speed of rule-matching operations. Uncommon attributes, such as different flags on packets or payloads are linked and the list of options are connected to chain header. Snort's detection engine checks rules for matching on chain headers and options (Roesch 1999). Rule chain mechanism is shown in Fig.2.3.

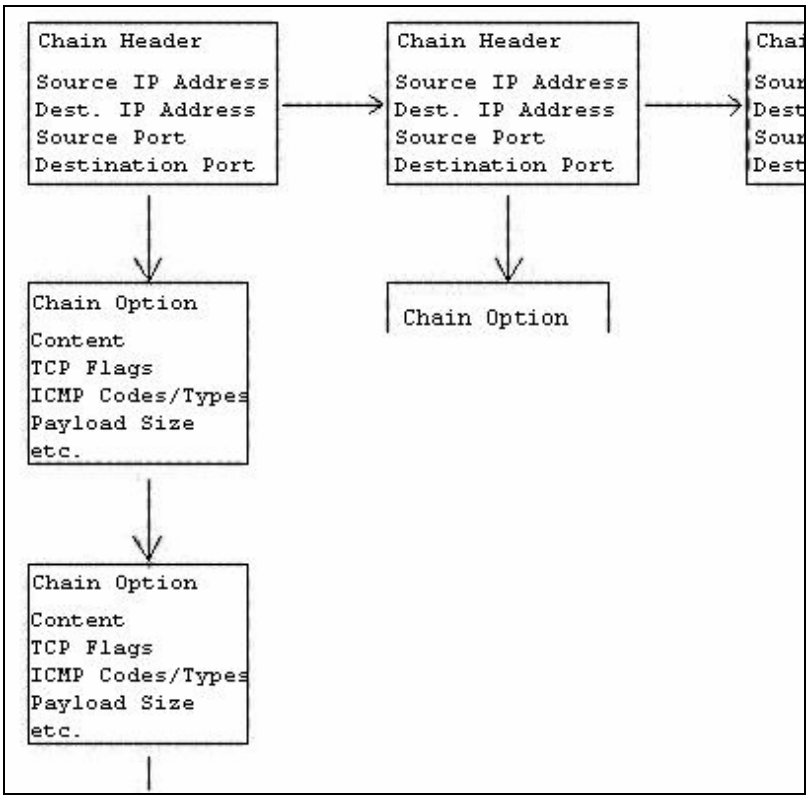

Fig.2.3.Snort's rule chain mechanism

(Redrawn, Original Source: Roesch 1999)

Snort may produce alerts, logs and warnings when a rule on a chain is matched. Rules and counter-measures on these cases may be altered by maintainers.

2.6.1.3. Preprocessors and Output Plug-ins of Snort

Snort can be extended by different plug-ins, thanks to its preprocessor and output plug-in enabling architecture. Special operations such as embedding different algorithms or additional event correlation may be added. Default snort package comes with different preprocessors such as port scan detection, http protocol inspector, fragmentation and flow control. Port scan preprocessor control incoming packets for port scanning probes and creates alerts. Another preprocessor of Snort is SPADE, which is to be evaluated in this study. Output plug-ins have different purposes such as reporting or additional post processing over data.

2.6.2. Estimates on Detection Rates of Snort

Since Snort is a rule-based system (NIDS mode is considered and compared on this context), its detection rate and success rate depends on its rules coverage. A good rule will help getting a high detection rate, with low false alarm rate. On the other hand, a badly written rule will sign a higher rate of legitimate traffic as hostile. Snort will help its users proportional to its master's ability to tune itself for more effective usage. By the way, estimates on the detection rate and effectiveness of whole system will be a rough one. Snort has different rule sources available, not limited to contributors from its user community, and Snort development team. It is also possible to subscribe for commercial rule services from different vendors.

In this study IDEVAL 99 DataSet is used for estimating this rate, in terms of successful detections and false alarm rate. Results and discussion may be found on Section 6.2

CHAPTER 3

INTRUSION DETECTION ALGORITHM IMPLEMENTATIONS

3.1. Overview

Intrusion detection aims to identify hostile activity on computer systems. Since all attacks on systems can not be modeled and identified, a range of attacks can not be detected by signature detection techniques. However it is possible to identify hostile activity by anomalies they create on systems or traces they leave. Detecting anomalies for detecting intrusion activities be a useful method for intrusion detection. Anomaly detection tries to identify events which are rare or previously unseen and unexpected events in the environment. This may be some features of packets in network traffic, clothes or people passing by some point. There is a rare or previously unseen event such as a man with a Mexican hat on an elephant passing by street, or a group of dancers dancing or a packet with some features which has not been seen before. When these events are to be modeled, need for information about the event and similar events arise. There is a need to define norm and a training sequence for defining what is rare. In addition to defining norm and training sequences, there is another question about nature of events in interest. Events, which may describe people passing, can be independent, or dependent, a group of dancers in a carnival city passing by avenue. Predicting an event's probability has been an interest for people for a long time, especially modeling purposes. There are models developed for one time predictions, such as Laplace and Good-Turing (WEB_10 2006) methods. (Mahoney 2003) has introduced a time based modeling method and used this time based method for detecting novel events and eventually for detecting anomalies in network traffic derived from Laplace's model. This model and mathematical background is summarized in next section.

3.2. Modeling Novel Events

In our everyday life we try to predict events in the future, next day's weather, color of next car which will pass along the street etc. We usually depend on our observations and ignore effects of unseen events. However unseen events also have their probabilities to happen. It is harder to accurately estimate unseen events' probabilities, or particularly novel values. It may be thought it is a good method to use experience from observations. However, observations will ignore events which may happen but have zero frequency in sampling or training period. Estimating probabilities for previously unseen events is called zero-frequency problem. Another unknown on this problem is size of alphabet or total set of possible events.

For estimating these probabilities, the method, known as Good-Turing has been developed by Jeffrey Good and Alan Turing. (WEB_11 2006) provides more information on historical perspective of the method. Good-Turing estimation has been extensively used in empirical linguistics area. On the other hand, this estimation method needs observations to be independent (Mahoney 2003). As stated before, network activity does not consist of independent events; instead the events have long range dependencies in time. The process may be described as self-similar or fractal, but not Poisson. Good-Turing estimation may not be the correct estimation method for assigning probabilities from now on, so a different method for estimating probabilities for novel events should be used. In addition, this method should not require independent events.

PPM (Prediction by Partial Match) (Cleary and Witten 1984) compression algorithm, which is an adaptive text compression algorithm, needs estimating probabilities for symbols to be predicted. In addition overall performance of compression is closely related to assigned probabilities for values not seen before. PPMC – PPM Method C (Witten and Bell 1991) has been introduced and shown to be experimentally better than other methods on estimating probabilities. For method C this probability is defined as:

$$\Pr(\text{next event will be novel}) = r / (n + r)$$

where r is the number of distinct types observed so far and n is the total number of observations. As n increases, this formula may converge to r/n.

(Mahoney and Chan 2001) provides one more coefficient for time based anomaly modeling, the effect of time and dependence of values seen before. An observation in which some values are consequently repeated, there is a probability of dependence between observations. If this is the case, calculating next event's probability will need information about previous observations. For anomaly based time modeling, let time amount of t has passed. Then average rate of seeing an anomaly becomes *t*. After amount of time *t* passes it is possible to see another novel event. So, P(novel) can be assumed as *1/t*.

(Mahoney and Chan 2001) constructs anomaly score mechanism as 1/P(novel) = *tn/r* applying previous formula since is constructed as P(novel) = (1/t)(r/n), where *n* (total number of observations) and *r* (number of types seen) are counted during the training period, and where *t* is the time in seconds since the last anomaly.

By an anomaly may occur during either training or testing, with the difference that if a novel value is observed in training it is added to the set of allowed values, but if it occurs during testing it is not. Note that in our model, P(novel) = (*r/n*)(1/*t*), which accounts for both the baseline rate of novel events, *r/n*, and a time-based model for events occurring outside the set of allowed values, 1*/t. tn/r* anomaly score is computed for each attribute on observed instance. Anomaly score of all features are added as all features are as of to be independent of each other. This anomaly scoring method will tend to produce higher anomaly scores as anomalous events are observed less frequently.

3.3. Packet Header Anomaly Detection

Packet Header Anomaly Detection (PHAD) algorithm is based on the anomaly score calculation method described above. PHAD models 33 header fields as attributes, found in Data Link (Ethernet), Network (Internet Protocol) and Transport Layer (TCP, UDP and ICMP) (Mahoney and Chan 2001). Fields for Ethernet, IP, TCP, UDP and ICMP Headers are shown in A.1, A.2, A.3, A.4 and A.5 in Appendix A.

All fields on the TCP/IP stack headers are not equal in size, longer fields such as MAC Address field in Ethernet frame has been split into two three byte long fields. One

byte fields such as SYN and ACK are grouped into one byte field. The list below is the list of fields modeled on PHAD grouped by layer and protocol (Mahoney 2003):

- **Ethernet header** (found in all packets): packet size, source address (high and low 3 bytes), destination address (high and low 3 bytes), and protocol (usually IPv4).
- **IP header:** header length, TOS, packet size, IP fragment ID, IP flags and pointer (as a 2 byte attribute), TTL, protocol, checksum (computed), and source and destination addresses.
- **TCP header:** source and destination ports, sequence and acknowledgment numbers, header length, flags, window size, checksum (computed), urgent pointer, and options (4 bytes if present).
- **UDP header:** source and destination ports, checksum (computed), and length.
- **ICMP header:** type, code, and checksum (computed).

PHAD stores values in a clustered approach. If a novel value for a field arrives during training phase, this value is merged into cluster it. If no group contains the novel value, it is merged into the cluster which the novel value is closer to.

3.4. Application Layer Anomaly Detector

Application Layer Anomaly Detector is an anomaly detection algorithm developed by (Mahoney and Chan 2002a). This algorithm differs from PHAD in three points. First difference from PHAD is that ALAD algorithm uses application payload for detecting anomalies, where PHAD observed fields of lower layer protocols. ALAD algorithm calculates anomalies using features found in incoming TCP server connections. Instead of using all incoming packets coming to monitored server port, features only found in three packets are used. These are first, next to last and last packet in communication. Their difference is that only text based application protocols are included (e.g. HTTP, SMTP, etc.), binary protocols such as (e.g. DNS, RPC, etc.) are excluded. Model covers at most 1000 bytes of application payload, arguments and keywords after 1000^{th} byte will not be included. ALAD can use conditional probability models applied on the features, such as P (source IP | destination IP) instead of using

only single probabilities, such as P (source IP). Features used to calculate conditional probabilities may be arbitrarily selected.

ALAD uses six features on reassembled TCP streams. These are:

- Source IP address, which is also the client
- Destination IP address, which is also the server
- Destination port number of the server application, which is related to service protocol.
- TCP Flags of first and last two packets, e.g. SYN, SYN/ACK or FIN/ACK. Flags are used for modeling state of TCP connection.
- Application keywords, the first word on a command line, such as a GET, POST or

HEAD in a HTTP request

- Application arguments, the rest of request command delimited by a line feed, such as

"/somefile.html HTTP/1.1", "Host: www.iyte.edu.tr" or "User-Agent: Mozilla/5.0" in a HTTP request.

Selecting features to be used in calculating conditional probability calculations has a large space of probable combinations of features. For example, in presence of 5 features, there exist 32 combinations ($2^5 = 32$) of features for antecedent event which contains no feature choice to all features choice. For consequent event, almost the same combination space applies, but alternatives which include chosen features for antecedent event are eliminated. Combination space grows exponentially as more usable features are considered for detection. It is obvious that all combinations of selected attributes and probabilities will show different detection rates. Selecting right combinations of probabilities will certainly help getting good detection rates and lower false alarm rates.

ALAD uses the same time based modeling model with PHAD. The same anomaly score detection formula, *anomaly score = tn/r*, applies for ALAD. Scoring is made on observations in testing phase, using information obtained in training phase. In conditional model, separate t, n and r values are maintained for each distinct value of antecedent event, where n holds total number of observations that consequent event also

happened, t is the time past since last anomaly, r is number of distinct values for the consequent event.

3.5. Learning Rules for Anomaly Detection

Learning Rules for Anomaly Detection (LERAD) (Mahoney and Chan 2003a) is an anomaly detection algorithm which generates rules from arbitrary combinations of nominal attributes (Attributes are denoted with A_i and values for the attributes are denoted with v_i). This algorithm does not need selecting rules; instead it generates and selects itself. It uses a rule mechanism composed of a set of constraints on attributes and defines set of allowed values for another attribute under these constraints (Mahoney 2003). Rules have the form:

$$A_1 = v_1 \text{ and } A_2 = v_2 \text{ and } \ldots A_k = v_k \Rightarrow A_{k+1} \in V_k = \{\text{set of allowed values}\}$$

Chosen rules are the ones which produce high values of n and low values of r, where n stands for number of constraints in antecedent events and r stands for number of elements for the set of allowed values for A_{k+1}. Rules are generated by observing instances in training session. Choosing the rule step has generation and elimination steps. This step defines normal behavior and rule set for normal. In testing phase anomalies for deviations from normal are to be found by checking instances for fitting these rules. If a rule is violated, then an anomaly score is calculated using the same formula as of PHAD and ALAD; anomaly score = tn/r. Total anomaly score is summation of anomaly scores for each violated rule. However there are separate n, r and t values for each rule. Time variable t is calculated from the last anomaly either in training or testing. Instances which do not violate a rule take 0 as anomaly score.

LERAD uses attributes similar to ALAD, but excludes binary protocols and makes use of keywords and certain information found in lower layer protocol headers.

3.5.1. Rule Set Generation

LERAD reviews training data and creates candidate rules using a randomized algorithm and tests them with more training data. It selects random pairs of instances from set of training instances. Using the values for attributes in chosen instances it tries to generate rules which produce n/r rate with 2/1. An attribute common in the training

pair becomes consequent event. Consequent attribute can take all or some of other attributes as antecedent. If more than one attribute is common, more rules can be generated. The more attributes exist; the more rules can be generated using same pair of data. This leads to a large set of rules which includes redundant and weaker rules.

3.5.2. Rule Elimination

Generated rule set may contain rules which may be superseded by other rules, in terms of their capability to cover training instances. These rules actually don't add extra capability for prediction to current rule set, thus they may be considered to be redundant. This leads to general rules with fewer numbers, instead of specific rules in vast numbers. There is another set of rules which are not redundant but perform poorly on detections. The rules which generate false alarms towards to the end of full training set are considered to perform poorly. This is due to nature of the attributes. Some attributes will have vast amounts of values increasing over time with a significant rate (e.g. names of ships visiting an international port), whereas some attributes will converge to a value set and will almost the same values over some time (e.g. names of ships coming to a port near a lake). Performance of rules is benchmarked by checking alarms they create at some limited time at the end of training phase. Since training phase doesn't contain traffic with attacks, bad rules will tend to produce more anomalies (i.e. false alarms in this context). Poor rules are removed before testing phase. Fig. 3.6 shows a graphical demonstration of good and poor rules growth rates and quality.

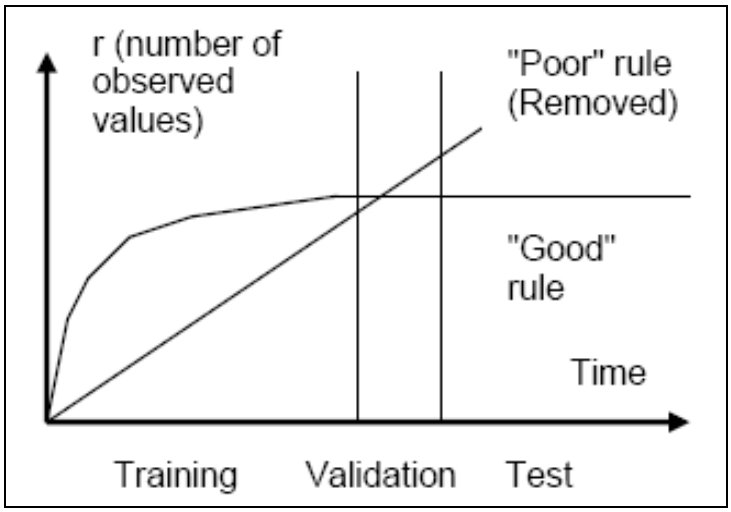

Fig. 3.1.Growth rate for rules in terms of performance
(Source: Mahoney and Chan 2003a)

3.6. Network Traffic Anomaly Detector

Network Traffic Anomaly Detector (NETAD) (Mahoney and Chan 2003b) is an anomaly detection algorithm, which detects anomalies in non-novel events in addition to detection of novel events. Some of non-novel events may also be considered anomalous if they occur sufficiently rare and not recently in this continuous modeling algorithm. This algorithm is more suitable for real world online systems because of its ability to adapt into changing nature of traffic in addition to its capability to accept mixed traffic (i.e. traffic which contains both attacks and attack-free traffic) in learning phase.

NETAD, similar to PHAD, works packet base, but uses only start of inbound client session traffic, unlike PHAD. This makes a small percentage of overall network traffic; ignoring rest of the traffic is some sort of trade-off for detecting anomalies in outbound responses. NETAD filters certain types of traffic since it is not used. These types of traffic are (Mahoney 2003):

- All non-IP packets (e.g. ARP, IPX etc.), since alarms are identified with their IP addresses
- All outgoing packet traffic.
- All TCP streams that begins with SYN-ACK.

- UDP packets to port number higher than 1023 (response to a local client)
- TCP packets with sequence numbers more than 100 past the initial sequence number (i.e. after the first 100 bytes of incoming client data).
- Packets addressed to any address/port/protocol combination (TCP, UDP, or ICMP) after the first 16 packets in 60 seconds (to limit bursts of UDP or ICMP traffic).

NETAD only needs first 100 bytes of the inbound IP traffic and uses first 48 byte out of 100. First 48 bytes contain IP and TCP Headers in addition to first 8 bytes of application payload. In presence of non-empty IP and TCP option fields, payload contents are not covered either. However NETAD considers each byte is an attribute, any shift in fields or values are simply out of interest.

NETAD model covers 9 types of packets. These nine types of traffic lead to 432 rules (Mahoney and Chan 2003c) in the same form of LERAD's rules. Nine models represent commonly exploited protocols in IDEVAL data set. The following rules are selected for NETAD, according to experimental results:

1. All IP packets (no antecedent).
2. All TCP packets (if protocol = TCP (6))
3. TCP SYN (if TCP and flags = SYN (2))
4. TCP data (if TCP and flags = ACK (16))
5. TCP data for ports 0-255 (if TCP and ACK and DP1 (dest. port high byte) = 0)
6. telnet (if TCP and ACK and DP1 = 0 and DP0 = 21)
7. FTP (if TCP and ACK and DP1 = 0 and DP0 = 23)
8. SMTP (if TCP and ACK and DP1 = 0 and DP0 = 25)
9. HTTP (if TCP and ACK and DP1 = 0 and DP0 = 80)

Anomaly score is calculated by summing anomaly scores of a packet for each rule. Anomaly score of a packet for a rule may be calculated one of the methods explained below:

Novel values only: anomaly score = tn/r. n is number of training packets satisfying the prior event, r is number of values seen in training for that field. t is the time passed since last anomaly either in training or in testing.

Validation weighed novel values: anomaly score = tn_a/r where n_a denotes number of packets satisfying the antecedent from the last training anomaly to end. This gives more weight for "better" rules than "good" rules.

Fast Uniformity Detection: Score = $tn_a(1 - r/256)/r$. This method helps degrade effect of rules for fields, in which most of possible values (out of [0,255] period) are already observed during training stage.

Non-novel values: Score = $t_i n/(n_i + 1)$, where t_i is the time (packet count, training or test)
since the value i was last seen, and n_i is the number of times i was seen in training. It reduces to $t_i n$ for novel events and t_i / f_i (with a Laplace approximation of $fi = n_i/n$ where n stands for overall packet count) for non-novel events.

Weighed model: Score = $t_i n/(n_1 + r/W)$, where $W = 256$ is an experimentally determined
weight emphasizing novel events. It reduces to $Wt_i n/r$ for novel events and approximately
t_i/f_i for non-novel events.

NETAD combined model: Score = $tn_a(1 - r/256)/r + t_i n/(n_1 + r/W)$. Combined model is sum of weight model and fast uniformity detection models (Mahoney 2003).

3.7. Statistical Packet Anomaly Detection Engine

Statistical Packet Anomaly Detection Engine (SPADE) (Hoagland 2000) is pre-processor plug-in developed for Snort. It had been actively developed by Silicon Defense, however due to financial problems, development ended in 2003. Since its source code was copyrighted with a free software license, source code has been open. Source code maintenance and distribution is now performed under Bleeding Edge Snort community (WEB_12 2006) sponsorship.

SPADE looks for anomalies in traffic by its task-specific detectors. SPADE can not actually determine a packet is hostile or friendly, but can say how unusual or how anomalous a packet is. SPADE, similar to PHAD, only monitors network and transport layer fields. SPADE has 5 different detector modes:

- **Closed Destination Port (closed-dport):** closed-dport detector watches TCP and UDP traffic for use of closed or rarely used destination ports on home (local) network. This detector may be useful for detecting probing attempts. Closed-dport detector is the oldest detector of SPADE. Waiting period can be defined for this detector type, which helps removing passive FTP issues. Alerts are fired if a closed destination port replies as RST packet or ICMP unreachable response message. Third case covers anomalous but open destination ports which are accessed rarely. SPADE has four different probability modes (*probmode* option) for this detector type:
  1. **Mode 0:** a Bayesian network approximation of P (sip, sport, dip, dport)
  2. **Mode 1:** P (source IP, source port, destination IP, destination port)
  3. **Mode 2:** P (source IP, destination IP, destination port)
  4. **Mode 3:** P (destination IP, destination port) (this is the default choice)

- **Dead Destination (dead-dest):** dead-dest detector watches traffic for use of IP addresses that are not used actually. Some worms use an exhaustive try-fail technique to spread in local and remote networks. This detector may help detecting these activities.

- **Odd Destination Port (odd-dport):** This detector watches traffic for port usage which differs from existing normal usage patterns. This detector may help discovering recently installed covert channels, special backdoors on possibly compromised machines.

- **Odd Port Destination (odd-port-dest):** This detector reports connections performed by clients on servers using unusual port numbers. This may help discovering compromised hosts on home network.

- **Odd Type Code (odd-typecode):** odd-typecode detector reports unusual ICMP traffic (in terms of ICMP type and code values) observed on network.

SPADE defines its normal usage by maintaining probability tables for monitored events. Records are weighted according to their occurrence time; newer events have more weight and older events have less weight on calculations. According to its probability tables all packets received by SPADE get an anomaly score.

There are two anomaly score calculation methods used in SPADE (Biles 2006). These are:

- **Raw Anomaly Score:** This score is calculated quite straightforward using formula

$$A(X) = -\log_2 (P(X))$$

Raw anomaly score can be confusing to remember and benchmark with similar scores. To overcome this confusion, relative anomaly scoring method is introduced.

- **Relative Anomaly Score:** Relative Anomaly Score tries to remove confusion caused by raw anomaly score. Relative AS is calculated simply by dividing raw anomaly score by highest possible raw anomaly score. This method will always produce anomaly values between 0 and 1, so it is easier to comment on rarity of the event.

3.8. Results of Previous Studies

Five anomaly detection algorithms, namely PHAD, ALAD, LERAD, NETAD and SPADE, are introduced in this chapter. These five algorithms have been trained and tested using IDEVAL data set and produced different results. Table 3.1 shows comparative results of detection performance of each algorithm tested on IDEVAL data set. Data is based on (Mahoney and Chan 2002b) and (Mahoney and Chan 2003c).

Table 3.1.Comparison of Detections of Selected Algorithms on IDEVAL Data

Algorithm/System	**Number of Detections (at 100 FA)**
PHAD	54
PHAD + ALAD	60
LERAD (avg.)	114
PHAD + ALAD + LERAD	85
NETAD	132

During development stage of this study, it has become obvious that there exist simulation artifacts in IDEVAL data, which could result misleading results. For example TTL field is considered an artifact since it doesn't change over time and has almost the same values in all simulation. It is far from reflecting real-world case. Even though background of IDEVAL data seems realistic generally, there are problems to be uncovered and probably there is more to uncover. Analysis of 1999 Evaluation data set has been made and revealed other artifacts found in simulation (Mahoney and Chan 2003b).

CHAPTER 4

DATA COLLECTION OPERATION

In Chapter 3, five algorithm implementations and their working principles are introduced. These implementations were tested on IDEVAL 99 Data Set and local data collected from a server from researchers' university. To test these algorithms with a different data, another data set is created with efforts of Izmir Institute of Technology (IZTECH) Computer Application and Research Center (CARC). This chapter introduces and describes some fundamental properties of local data set, provides comparison with other datasets.

4.1. Overview

There are about 2000 computers working in a weekday in IZTECH campus, serving as clients and servers. Client computers are used for accessing the Internet and different servers in campus network. Users access Internet via a router which connects whole campus network to Internet Service Provider (ISP). There are two IP address blocks used in campus network, one real, one virtual. Real IP address block is generally used for servers on campus allowing to be accessed from the Internet. Virtual IP address block, which contains addresses available for private uses of enterprises, are used to connect clients to the Internet. These computers are not accessible from outside of the campus. These clients connect to the Internet via a gateway. Table 4.1 shows these address blocks and their properties

Table 4.1.Available IP address blocks in IZTECH campus

Network	Real/Virtual	Assignment	Usage
193.140.248.0/22	Real	Assigned by ISP	Servers and Clients
10.10.0.0/16	Virtual	Free for private uses	Clients

4.2. Data Collection Environment and Properties

Data collection operation is performed by a dedicated computer installed by IZTECH-CARC personnel, between 04.06.2006 and 04.17.2006 for sampling from servers for data inspection and intrusion detection research purposes. Data consists of sniffed network traffic of four servers located in campus, each of which has different purposes and operating systems. List of servers and their operating systems are shown in Table 4.2.

Table 4.2.List of servers, whose traffic has been collected.

Server Name	**Goal**	**Operating System**
bbsserver	-	Solaris
likya	-	Linux
gulbahce	-	Linux
ftp	file transfer	Windows NT Server

The servers and the sniffer computer are connected to the same network switch. All servers are connected with a 10/100 Mbit Ethernet card. All traffic of the servers was cloned into sniffer's port using "port mirroring" feature of network switch. Simplified architecture of data collection operation and campus network is shown in Fig. 4.1.

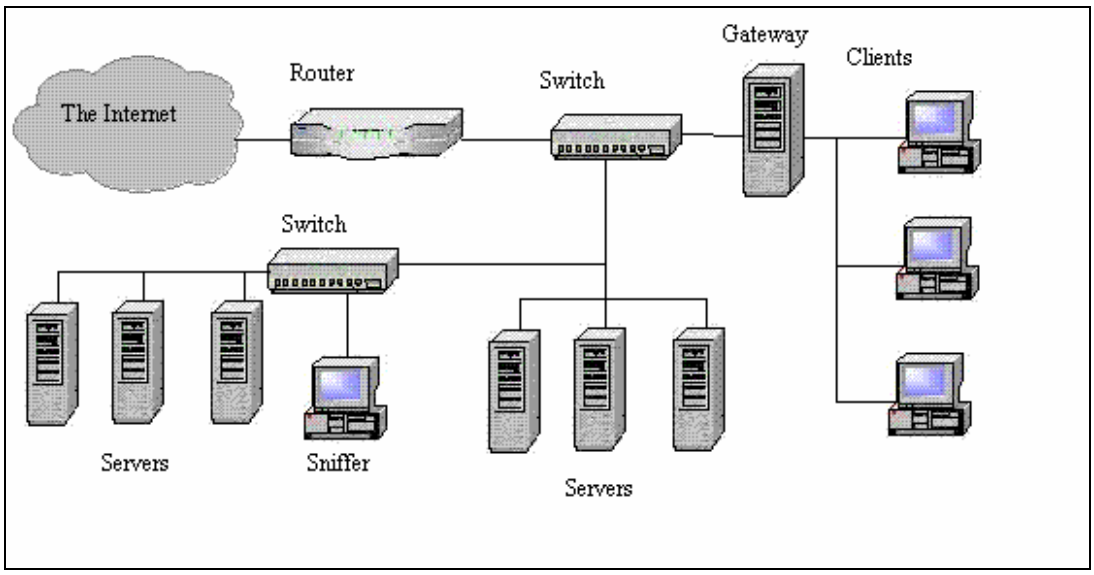

Fig. 4.1.Simplified architecture of campus network and data collection operation

Traffic collection starts in 04.06. 2006 at 14:32:14 and ends in 04.17.2006 at 17:14:16. It spanned in 12 days, but collected traffic data is almost 11 days long, takes space about 39 GB. Whole data is not is saved as a single file, instead divided into numerous parts with different sizes. These files were reorganized to reflect daily traffic, using a patched version of tcpslice program. Tcpslice is a program is used to perform cut and paste operations on fragments of network traffic data files. Detailed information about these such as size and numbers is given in Appendix B. Main findings and certain properties of dataset are described in Chapter 6.

CHAPTER 5

DEMONSTRATION OF ALGORITHM IMPLEMENTATIONS

PHAD, ALAD, LERAD (both variants), NETAD and SPADE algorithms are to be tested using IDEVAL and data collected from IZTECH campus. PHAD, ALAD, LERAD and NETAD are standalone applications, SPADE is a preprocessor plug-in developed for integration with Snort. This chapter describes demonstration process of these implementations.

5.1. Testing Environment and Tools

5.1.1. Testing Environment

Hardware: Tests were made on a PC with Pentium IV 2.9 GHz (with Hyper Threading support) processor and 1GB of RAM.
Operating System: Fedora Core 4 with kernel version 2.6.11-1.1369_FC4.

5.1.2. Testing Tools

Following software packages were used for evaluation:

- tcpdump tool version 3.9.4
- Libpcap packet capture library with version 0.9.4 for individual operations
- Snort lightweight IDS version 2.4.2 with SPADE - Integrated tarball version
- Ethereal version 0.99.0 with libpcap version 0.8.3. This older version of libpcap did not have any version conflicts with latest version
- Source codes for PHAD, ALAD, LERAD, NETAD, EVAL and other supplemental material provided by (WEB_14 2006)
- gcc compiler v.4.0.0 20050519 (Red Hat 4.0.0-8)

5.1.3. Modifications on Configurations and Programs

5.1.3.1. Configuration of Snort and SPADE

Registered User Release of Snort rules available as of May 25 2006 are used. Default configuration is modified to include more rules in numbers, with allowing minimum rules about non-hostile activities, such as online gaming or IRC chat. Default running configurations of preprocessors are not changed except addition of SPADE configuration lines. 030125.1 version of SPADE is used.

Snort's and SPADE's "home network" settings were modified to reflect IP address blocks of evaluation networks. No extra rules have been written to adapt specific properties of networks. No special tuning operation was performed on Snort except changes explained below. Following rule sets were excluded on both operations:

- **chat:** rules for detecting IRC and instant messaging (IM) activity such as MSN
- **multimedia:** rules for detecting various multimedia material transfer
- **p2p:** rules for detecting peer to peer networking activities such as file sharing
- **experimental:** experimental rules, which was already empty in evaluation
- **porn:** rules for checking sexually explicit material on content

In IDEVAL scenario, servers are remotely monitored by remote Air Force computers. Two SNMP rules and one other rule caused more than 40000 alerts in a single attack-free day and hence they were disabled. The disabled rules are as follows:

- "Web bug 1x1 gif attempt" alert (web bug is a 1x1 gif file used for tracking page visitors' trends (WEB_15 2006))
- "SNMP public access udp" alert (Simple Network Management protocol is designed for remote monitoring and management, which generally uses UDP as transport protocol).
- "SNMP request udp" alert (a SNMP request is made)

SPADE had one detector open with following configuration:

- Detector type: closed-dport
- TCP Flags: synonly (only SYN flag is set)

- To: home (packets are destined to home network clients)
- Wait: 3 (Determines time out period for a host to reply an incoming SYN packet)
- Protocol: TCP
- Probability Mode: 3 (the probability model: P (dest. IP, dest. port), explained in section 3.6)

All alerts (saved into a text file named “alert”), logs (saved into a binary file named Snort.log.x where is a positive integer, indicating startup time in Unix time format) and SPADE state files (spade.rcv) are logged into /var/log/snort directory.

5.1.3.2. Modifications on Algorithm Implementations

Original source code of PHAD failed to process IZTECH dataset, because of byte order difference problem (WEB_16 2006). IDEVAL data was collected and saved on big-endian machines (Haines et al. 2001). Original source code could only process big-endian data. Since IZTECH dataset was collected on a PC (with 80x86 architecture), it is saved as little-endian. Modifications were made to original source code and tests were performed to check both accuracy of modified version and consistency with original program. Modified versions of the source code are presented in a CD provided with this study. Contents of the CD are in Appendix C.

5.2. Training Systems

Snort did not need any training, because of being a signature based system. SPADE has no prior training period, for first run. In every run SPADE records observations for monitored activities, after 50000 updates. At the end of each running session, SPADE records its current state, which holds statistical information. SPADE uses incoming packets provided by Snort and needs no extra preprocessing for both training and test periods. At each start of Snort, it searches for state file. Snort logs and alert files were relocated for further processing in order to store separate alert files for different days. SPADE state files were not relocated for consequent evaluations for members of the same dataset.

PHAD and other algorithms were trained using IDEVAL data set's attack free traffic provided for systems training. In fact, real time traffic can contain novel attacks. For algorithms which can accept mixed traffic, all IDEVAL traffic is used.

PHAD, ALAD, LERAD and NETAD are standalone applications which run using given parameters from command line. PHAD takes two primary inputs; training time in seconds, list of data files. Input files are ordered chronologically. Training phase starts from the instance of earlist packet in first data file and spans as long as given training time parameter. Each packet after training period ends is used as test data.

ALAD, LERAD and NETAD have a two pass approach. First pass covers preparation of data for processing, such traffic filtering, keyword or feature extraction from data files, etc. There are three supplemental programs used for data preparation. These are

- te: traffic extraction utility, extracts TCP streams from tcpdump files.
- a2l.pl: a perl script which converts output of te to LERAD compatible format
- tf: traffic filtering program

ALAD uses interim files provided by te as data. It takes two parameters, one for training data file and other for test data file, both of which are interim files. Output is in sim format.

LERAD uses data files similar to ALAD. ALAD compatible files are converted for use of LERAD using a2l.pl. Outputs are text files. LERAD uses three parameters; first two is similar to ALAD, these are training and testing files. Third parameter is random number seed - required for LERAD's randomized algorithm. Another variant of LERAD uses interim data files generated using tf.

NETAD also uses data files generated by tf. Since training period is hardcoded in original version, it does not use external parameters other than training and test file names.

5.3. Testing Systems

Detection performance operations on IDEVAL were performed by EVAL program, which is compatible with original evaluation detection criteria. For demonstration on real data, Snort is used for benchmarking since there is not an available evaluation program for analyzing live data similar to EVAL. A portion of

IDEVAL training data is used for training systems for algorithms which require attack-free data. In addition, a portion of real data is used for testing after training.

CHAPTER 6

RESULTS AND DISCUSSION

6.1. Findings on IZTECH Dataset

There are significant differences in simulated traffic of IDEVAL data set and a real traffic data. Some of simple and significant differences between IDEVAL, IZTECH and FIT data sets are described in the following sections. Since FIT data is not publicly available, an available source of statistical information for this dataset is (Mahoney and Chan 2003c). This data source is served as main tool for making comparisons. Ethereal software package tools (tethereal and capinfos) and Snort was used to extract statistical information from datasets. Graphics are generated using Microsoft Excel.

6.1.1. Description of Collected Data

6.1.1.1. General Properties

Data set contains about 74 million packets and takes 39 GB of disk space. Daily distribution of collected packets and size of traffic data is shown on Fig. 6.1.
Collected data is almost continuous, except a 50 min. gap in Apr 7 2006. Average bandwidth usage has been 339 Kbits/sec for overall traffic. Data rates of individual samples vary between 40 KBits/sec and 20 Mbits/sec. Average bandwidth usage for days are shown in Fig. 6.2. Average packet size of samples varies between 324 and 691. Daily averages for packet size is shown in Fig. 6.3. Most common protocols on daily traffic are shown in Fig 6.4 and 6.5.

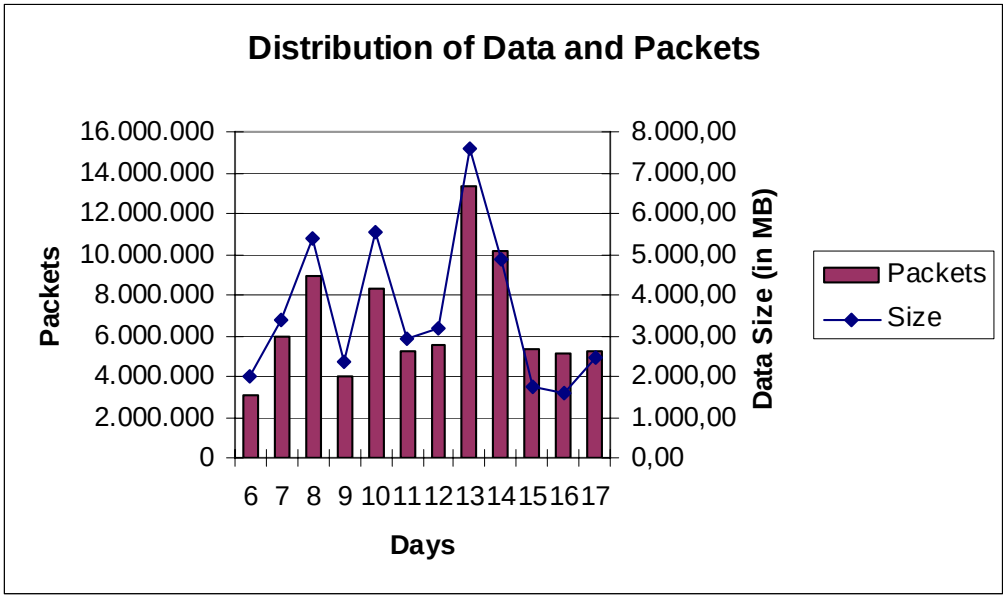

Fig. 6.1. Daily distribution of data and packets

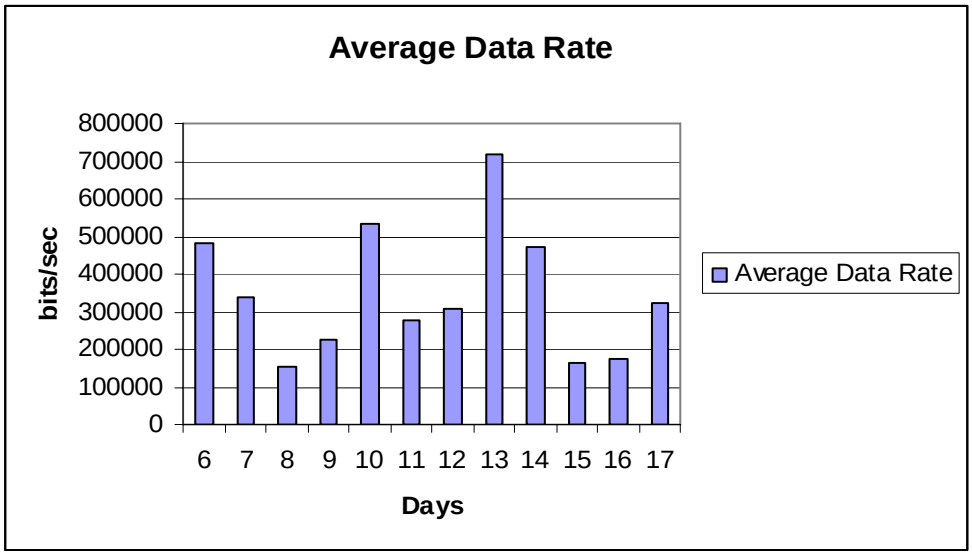

Fig.6.2 Daily data rates in data set

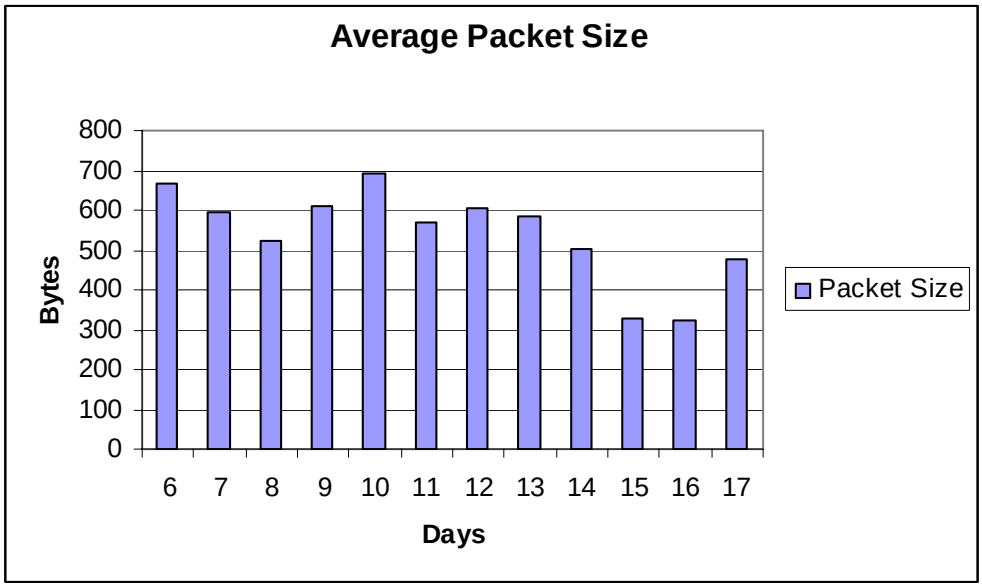

Fig.6.3.Average packet size graph for data set

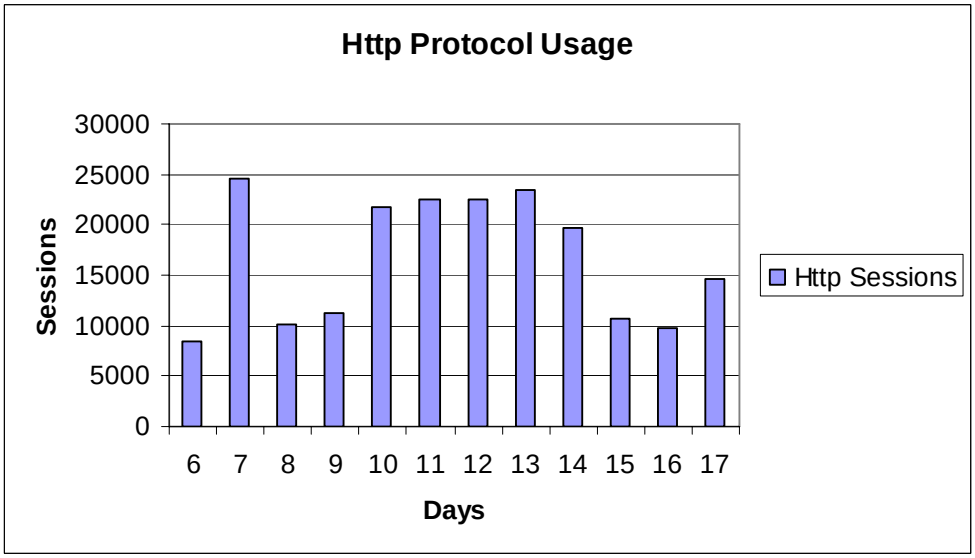

Fig.6.4.Daily distribution of HTTP traffic

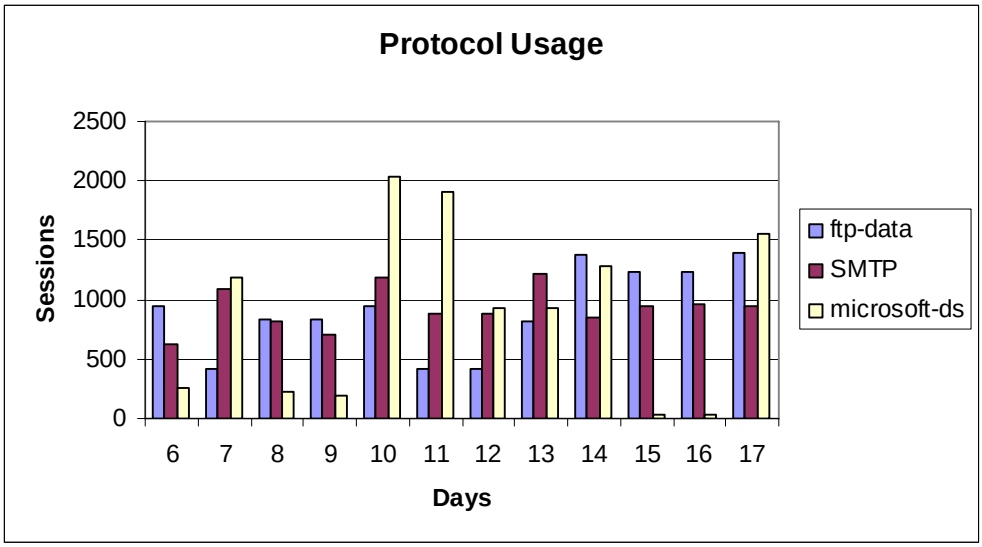

Fig.6.5.Daily distribution of other three frequently used protocols

6.1.1.2. Intrusion Activity

At first glance, collected traffic may seem to be relatively clean. However after careful examination, it has become clear that attack patterns exist and distributed among all data. Only one of 43 data file was found to be clean, however this may be considered trivial, since file contains about 497 packets recorded in 10 seconds and takes only 90 KB in disc.

Evidences of probing attacks can be found in almost every data sample. Most probes were checking for open SQL Server ports (port no: 1433). Most frequently scanned ports are given in Table 4.3 with their possible attacker and application.

Table 6.1. Most frequently scanned ports and count of instances

Port	Application (Legitimate or Malicious) and Possible Reason	Probe Count
1433	Microsoft SQL Server	2230
445	Microsoft-ds - Server Message Block (SMB) and worms	2181
139	Netbios Session Service and Trojans	2132
80	WWW and various Trojans	2069
1080	SOCKS Proxy Server, trojans, worms or spammers	1133
15118	dipnet trojan backdoor, worms	746
8080	www alternative port, trojans and backdoors	663
5900	Real Virtual Network Computing (VNC)	659
3372	Microsoft Distributed Transaction Coordinator (DTC)	455

22	Secure Shell (ssh) and Trojans	406
1026	Windows Remote Procedure Call (RPC), worms	Unknown
1027	ICQ Instant messenger	Unknown

Two ports have experienced unknown number of probes. This is due to uncertainties in probing activities and imprecise nature of port-scan preprocessor of Snort. Probe counts are obtained after processing of Snort alert files. The port-scan and Spade preprocessor alerts have alerted for different activities, sometimes overlapping.

There are traces of spammer activity in traffic, looking for badly configured mail servers for relaying mail messages or directly sending messages to server. Probing for sending spam may be considered as commercial activity employing illegal methods.

There are two types of probes: probes which focus on certain port on a range of computers and probes focus on open ports of a specific computer. In addition there are other probes, such as ping probes for various IP address ranges, usually used for checking whether a specific computer is connected. These ping and port probes may be used together to cooperate by attackers.

There is an attack wave between 04.13.2006 to end of data collection directed to open port 3306 on one of servers. This port is used by MySQL Database Management System. Remote attacker has tried to guess root password of the server. Total number of detected trials in that period is 663451. There is not an evidence of success of attacker, since no data exists about successful login and probing operation was underway at the time when data collection ended. It should have ended before end of collected data. Port 3306 has also been probed in port scans, but not as much as other ports. In addition there are other password trials in other days, but numbers of consecutive trials are much lower.

6.1.2. Comparison with IDEVAL Dataset

IDEVAL dataset is produced after a simulation effort, made in 1999, according to statistical information collected in 1998 from various US Air Force bases. Content of simulated traffic differed from our traffic in many ways, such as data and packet rates, protocols etc.

IDEVAL traffic data has a smaller data rate and packet size, producing a smaller data set. Generated traffic, which covers only weekdays of 5 weeks, contains about 52M

packets and takes about 10.8 GB of space. Daily distribution of traffic in IDEVAL is shown in Fig. 6.6.

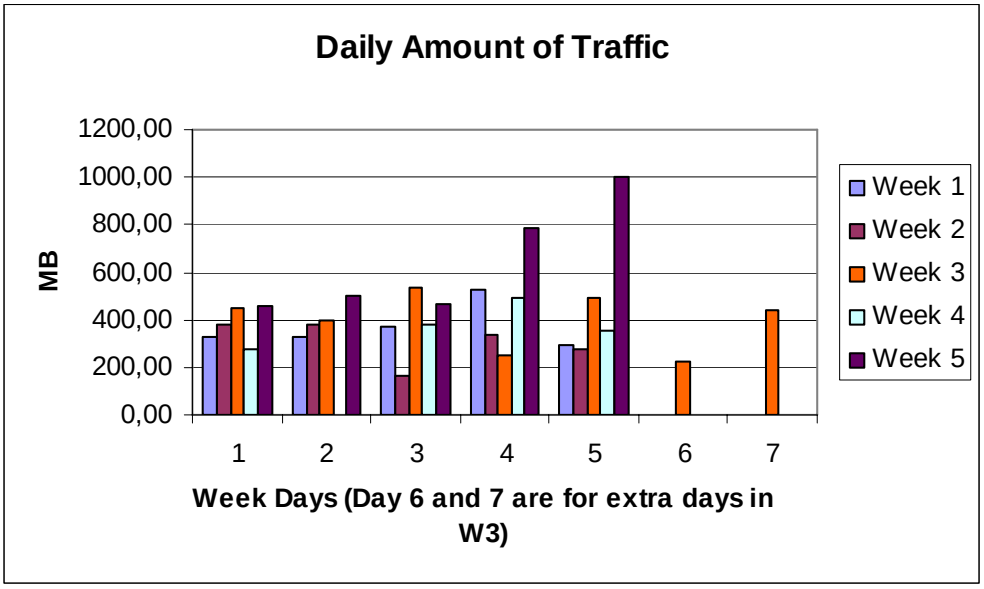

Fig.6.6.Daily traffic distribution in IDEVAL

IDEVAL traffic is different from our dataset in many ways:

Synthetic Traffic: IDEVAL traffic is created in laboratory environment using scripts and other sorts of programs. Properties of the traffic which is to be synthesized is based on statistical information that came from military based in different locations. Our traffic is collected from a campus network and is generated by users in campus, people and software accessing the servers. These large communities with different things in mind have helped to create a traffic record with an unrepeatable and surprising nature.

Age and Span: IDEVAL data sets have been built according to statistical data which belongs to 1997 (Haines et al. 2001), reflecting trends in that time. It spans about 6 weeks (Extra days of Week 3 is actually fourth week of March) and is recorded into daily partitions. Weekend days are excluded in simulation. However our recorded traffic spans 12 days, including weekends and almost continuous.

Different trends: IDEVAL traffic is created based on different trends and usage policies. By time some of these policies or trends change. For example using finger to check whether a user exists in a mail server before sending a mail is a trend in IDEVAL traffic. However finger action has long been discouraged from practical usage (WEB_13 2006). Visited sites and tools may change place to place and people to

people. This is another difference. In IDEVAL traffic there is no crawler activity, however these agents have become most common visitors of websites.

Daily Data Size and Packet Counts: Daily data size of IDEVAL is very different from our traffic. It has smaller data size, lower number of packets and thus a lower bandwidth usage. IDEVAL case may be considered as unrealistic for today's Internet. As it is obvious in Fig. 4.3 and 4.7, highest traffic load in IDEVAL is about 25% of lowest traffic load in our dataset. However this information should not be considered decisive; since more real time data may help denying this proposition.

Protocol Distribution and Variety: There are similar protocols used in simulation and in real world, these are ARP, IP, TCP, UDP, ICMP, SMTP, POP3, FTP,FTP-DATA, IMAP, nbname, nbdgram protocols. No IMAP traffic was captured in campus but had different other protocols running such as Spanning Tree Protocol (STP), nbss, Microsoft-ds and non-IP based protocols. There is limited use of telnet and ssh in traffic.

Traffic Scope: IDEVAL network traffic has been collected by two sniffers, labeled as inside and outside, recording all traffic passing by. For inside traffic, it covers communication between hosts, too. Our collection has a limited scope of traffic that is about a group of servers, connected to same switch. Our approach excludes clients' activities in general except their communication with sniffed servers.

Traffic Regularity: 5 weeks of synthetic data has about 40 million packets but none of these contain bad checksums, either IP or TCP. Our samples, which covered more than half of overall traffic, has bad IP and TCP checksums in different numbers, but with quite low percentage in overall data. Bad checksum probability is about 10^{-5}.

Discarded Packets: On evaluation phase, Snort and SPADE has been used to search intrusions in IDEVAL data for evaluating detection rates. The same procedure was also applied to IDEVAL data. During inspection Snort discarded 18 packets in our dataset. However Snort did not discard any packets when inspecting IDEVAL data. Snort discards only packets which could not be parsed. Reasons for this behavior are not clear, malformed packets exist in our dataset. It is also unknown that whether this is a feature or possible bug of the dataset.

Variety in Protocol Usage: IDEVAL dataset has only GET commands in HTTP conversations. However this is unrealistic when compared to our results. There are other commands used, even much less frequent, such as POST, HEAD, OPTIONS,

PROPLINK, PUT and CONNECT. In addition, available protocol versions are 1.0 and 1.1, this due to the fact that our data is recorded in 2006.

6.1.3. Comparison with FIT Dataset

Dataset used in (Mahoney, 2003b) has been collected from a departmental server, which served web pages and several accounts. Dataset was collected in weekdays over 10 week period.

Our dataset was collected using feature of the network switch; this helped monitoring more than one server in dataset. Department server is said to be behind a firewall, providing additional protection for probes coming from external probes and possible exploit attempts. Our dataset has more probes and brute force attacks included.

For HTTP traffic our observations are similar, more keywords for HTTP exist:

FIT group has experienced more values for http commands in their dataset, found 9 commands in data (GET, HEAD, POST, OPTIONS, PROPFIND, LINK and two malformed) where GET is dominant over others by 99%. We have observed PUT and CONNECT commands as different commands but not observed any LINK in 199302 requests. 99% percent of commands were GET.

Similar points and significant differences between three datasets are discussed. Two of these datasets were collected from real networks. Since second real traffic dataset was not publicly available, only source of information has been used for comparison. Third dataset, IDEVAL, was other available dataset but differed from other two, because of its synthetic nature.

6.2. Results of Demonstration of Algorithms

In this study, performance of two anomaly detection algorithms, one as a standalone implementation, one as a preprocessor plug-in integrated with Snort was demonstrated. In original study of (Mahoney and Chan 2001), results were promising but on further studies it was revealed that results could be misleading because of existing artifacts in simulation (Mahoney and Chan 2003b) – which could hide real performance of algorithms and systems.

IDEVAL 99 data set has been the most comprehensive and publicly available work in evaluation of intrusion detection systems. Dataset included real attacks on hosts, with simulated background traffic, created with statistical information from real production environments. Further studies showed that there are issues with possible bugs, probably caused by idiosyncrasies of simulation. There are problems with its nature and structure of its compatibility of modern internets. Statistical data used for IDEVAL sets belonged to 1997, compatible with its time, but not with today's network trends and traffic. New trends, technologies and threats have emerged by time after 1997, but nature of this dataset prevents development of anomaly detection algorithms which will fit with today's world.

For evaluation of algorithms in a real network environment, network traffic data was collected from servers of our university. Collected data has shown significant differences with simulated data. Real dataset contained real attacks distributed into data, making it harder to use with clean-data sensitive algorithms. This is a certain disadvantage for systems with learning based approaches. Reducing number of these attacks may be succeeded with more restrictive firewall and network usage policies.

SPADE algorithm, integrated and evaluated with Snort, had difficulties in detecting intrusions, producing so many alerts, some of which were false alarms. False alarms of Snort came from http-inspect preprocessor, which comes with standard package and is started with Snort by default. It produced false alerts on requests of files which included letters encoded in URL format (e.g. using "%20" instead of space character), which contains characters found in Turkish alphabet and not in English alphabet, such as ğ, ş and İ. Another preprocessor named portscan also produced relatively higher amount of false alerts, which is thought to be related for being untuned. Snort rules were configured to run with minimum changes, closer to default configuration. Snort and other intrusion detection systems need monitoring and modifications from default configuration when deployed in real environment. This may be thought as a form of training to reduce number of false alerts.

SPADE is a useful tool, especially for detecting probes to unusual ports. However it has a serious flaw: it has no correlation mechanism between events in a time window, but portscan has. For example, when a probing event occurs for x ports on y machines, SPADE will produce (x*y) alerts if all packets are over defined threshold level. A similar tool, port-scan preprocessor has correlation capability and will produce only one alert for this probing activity. Approach of port-scan detector is better than

SPADE in detection of closed ports, even though alerts were overlapped in evaluation. Spade produced many alerts in evaluation with both datasets. In availability of port-scan preprocessor, closed-dport detector is not necessary.

PHAD algorithm was evaluated after Snort and SPADE. Special modifications were made on original code. PHAD had problems in evaluation with real time data. PHAD algorithm failed to fit test environment due to its rigidity and tight bounds to underlying structure. Another reason is duration and difference of training data.

6.2.1. Snort and SPADE on IDEVAL

Snort and SPADE data has been used to test IDEVAL data. It is aimed to estimate detection and false alarm rates on following configuration. Snort and SPADE settings of this test have been explained in 5.1.3.1. Snort and SPADE records all alerts into one single text file. Evaluation of detection rate is made using EVAL where all alert data must be entered in form:

ID	Date	Time	Victim IP	Alarm Score	Comment
0	03/29/1999	14:33:12	170.70.71.73	0.854322	# notes

ID value has no specific meaning and is ignored in evaluation; date is given in MM/DD/YYYY format; victim's IP address in dotted decimal notation. Alarm score determines certainty of attack. Higher alarm score means, this activity has higher probability of being hostile. Alarm score starts from 0. Attacks are reported in Eastern Standard Time (EST) for first four weeks. Valid time zone for fifth week is Eastern Daylight Time. For converting snort alert files into sim format, all alarms were given alarm score 1.000000.

Snort alert files were processed using a conversion program to be compatible with input format of described above and daily alert outputs were converted to sim files. Each alert were given score 1.000000. On first evaluation, each sim file was evaluated separately using EVAL. In second evaluation, all created sim files were merged into a larger file. The merged sim file was used for evaluation with EVAL, in second evaluation step. Tables 6.2 and 6.3 shows number of true detections and alerts for training and test weeks. Figure 6.7 shows Detections/False Alarms Threshold Level

curve created for detection rates of Snort and SPADE with different tolerance of false alarms.

Table 6.2. Number of alerts, true detections and packets in training data

Week/Day	Alerts	Attacks	Detections	Packets
W 1 / D1	220	0	0	1.495.808
W 1 / D2	274	0	0	1.240.260
W 1 / D3	563	0	0	1.730.292
W 1 / D4	476	0	0	1.951.904
W 1 / D5	605	0	0	1.487.186
W 2 / D1	915	7	0	1.755.742
W 2 / D2	3965	9	2	1.588.037
W 2 / D3	11194	6	0	1.014.119
W 2 / D4	18760	9	1	1.566.930
W 2 / D5	17487	12	0	1.365.264
W3 / D1	590	0	0	2.110.223
W3 / D2	762	0	0	1.834.417
W3 / D3	949	0	0	1.853.383
W3 / D4	872	0	0	1.562.141
W3 / D5	858	0	0	1.638.336
W3 / D6	981	0	0	1.683.381
W3 / D7	332	0	0	2.157.318
Total		43	3	

Table 6.3.Number of alerts, true detections and packets in test data

Week/Day	Alerts	Attacks	Detections	Packets
W4 / D1	297	17	3	1.651.481
W4 / D2	0	12	0	0
W4 / D3	1134	19	6	1.768.940
W4 / D4	1220	10	3	2.359.214
W4 / D5	1119	17	3	1.949.641
W5 / D1	5776	27	12	2.294.746
W5 / D2	7456	25	10	3.407.858
W5 / D3	1656	17	10	2.091.431
W5 / D4	13567	21	15	3.205.259
W5 / D5	1246	32	16	3.397.462
		197	78	

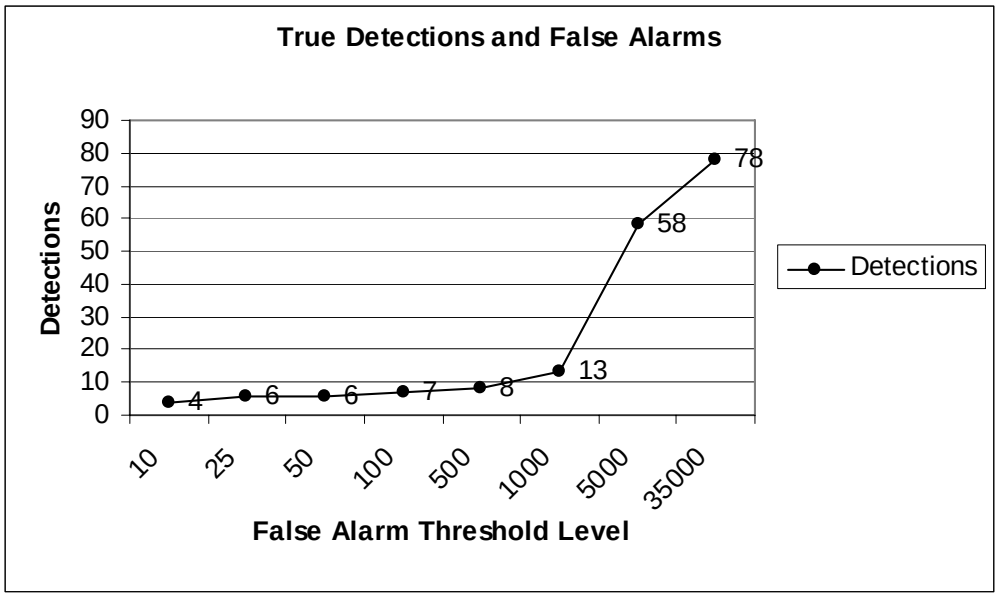

Figure 6.7.Detection/False Alarm Threshold Level Curve for test weeks of Snort and SPADE

6.2.2. Snort and SPADE on IZTECH Data

Snort and SPADE has been used to detect intrusions on real time dataset. Snort and SPADE's alerts are shown in Table 6.4. Alerts are grouped to show SPADE's effect. Ratio of SPADE column shows alert groups created by SPADE to all alert groups ratio.

Table 6.4.Top alerts and ratio of SPADE related alerts in total groups of alerts

Day	Part	Packets	Alerts	Top Alert	Top Alert Definition	Ratio of SPADE
6	1	1.582.906	391	29	robots.txt	235/256
6	2	502	0	0		
6	3	1.482.899	1079	440	ICMP Ping	224/252
		3.066.307				
7	1	2.779.952	1961	144	robots.txt	650/678
7	2	1.354.134	526	111	SPADE	207/226
7	3	1.379.683	373	26	robots.txt	177/200
7	4	388.390	155	45	robots.txt	36/55
		5.902.159				
8	1	2.411.406	726	190	robots.txt	175/208
8	2	775.667	292	73	robots.txt	90/116
		3.187.073				
9	1	2.527.108	74836	4657	SPADE	281/319
9	2	1.343.585	184	61	robots.txt	41/63

9	3	151.694	48	21	robots.txt	3 / 14
		4.022.387				
10	1	1.995.399	2059	757	SPADE	324/352
10	2	2.523.863	2774	761	SPADE	244/273
10	3	1.377.922	217	49	SPADE	44/59
10	4	1.537.600	53	6	SPADE	20/28
10	5	881.384	682	86	robots.txt	79/111
		8.316.168				
11	1	1.478.832	591	112	robots.txt	153/183
11	2	2.878.084	3230	814	SPADE	476/506
11	3	924.230	822	218	SPADE	149/180
		5.281.146				
12	1	1.424.997	780	165	SPADE	137/162
12	2	2.733.034	1035	115	SPADE	349/375
12	3	1.362.016	805	164	SPADE	126/150
		5.520.047				
13	1	690.701	319	97	robots.txt	34/54
13	2	1.864.309	399	59	SPADE	148/168
13	3	2.589.204	26	6	robots.txt	5 /17
13	4	2.342.495	174	25	ping	48/72
13	5	2.002.180	429	128	SPADE	111/134
13	6	2.202.247	333	113	SPADE	64/76
13	7	1.654.141	6515	5854	MYSQL 4.0 root login attempt	80/106
		13.345.277				
14	1	664.181	32409	32227	MYSQL 4.0 root login attempt	80/108
14	2	2.161.180	44526	43457	MYSQL 4.0 root login attempt	17/37
14	3	3.267.559	18025	16359	MYSQL 4.0 root login attempt	235/260
14	4	2.127.386	1081	1045	MYSQL 4.0 root login attempt	486/515
14	5	1.902.083	52803	51927	MYSQL 4.0 root login attempt	15/28
		10.122.389		145015		
15	1	1.257.635	66032	64564	MYSQL 4.0 root login attempt	39/63
15	2	2.507.106	71781	71190	MYSQL 4.0 root login attempt	40/69
15	3	1.566.383	71286	70187	MYSQL 4.0 root login attempt	105/134
		5.331.124		205941		
16	1	3.828.679	145537	144418	MYSQL 4.0 root login attempt	104/137
16	2	1.313.619	59731	58400	MYSQL 4.0 root login attempt	91/115
		5.142.298		202818		
17	1	2.332.617	89260	87729	MYSQL 4.0 root login attempt	351/372
17	2	2.715.846	14265	13050	MYSQL 4.0 root login attempt	511/536
17	3	183.172	3080	3044	MYSQL 4.0 root login attempt	15/28
		5.231.635		103823		
		74.468.010				

Alerts were grouped according to their names and SPADE groups were also grouped according to created anomaly scores. This is the main reason of SPADE alert groups. These alerts are created when a probing event occurs either by a human or by a worm. Combinations of IP addresses and port numbers result with many alert groups with relatively low population.

6.2.3. Anomaly Detection Algorithms on Iztech Data

6.2.3.1. PHAD

Original PHAD source code could not handle Tcpdump files stored on a PC, modifications were made to solve this problem (modified version will be referred as PHADm from now on). First PHADm run used Week 3 of IDEVAL data for training and a sample from data set for test. It resulted with 99996 anomalies. 4 of top 5 fields for anomalies belonged to Ethernet protocol header. In second run, in order to include into traffic from our institute, clean data from our dataset has been added to evaluation. PHADm also included part of real network data for evaluation. On third trial, real and synthetic data have been used to train system and real time data to test. Both evaluations resulted with fewer number of alerts but still very high. Total number of reported anomalies was more than 20000. Alerts in high numbers have shown that this algorithm is sensitive to amount of training and significant changes in underlying network structure.

6.2.3.2. ALAD

ALAD, even though used a different method of anomaly calculation, it is not evaluated due to problems occurred in evaluation of PHAD.

6.2.3.3. LERAD

LERAD algorithm relies on data provided by interim-data created for evaluation of ALAD. Cancellation of ALAD evaluation also cancels LERAD evaluation because of similar algorithmic handicap and lack of usable data.

6.2.3.4. NETAD

NETAD algorithm evaluation did not happen due to limited time and similar algorithmic handicaps for successful evaluation.

CHAPTER 7

CONCLUSION

In this study five anomaly detection algorithms (PHAD, ALAD, LERAD, NETAD and SPADE) and Snort, a commercial signature based intrusion detection system, is introduced.

In first step of demonstration, Snort and SPADE were tested on synthetic IDEVAL dataset. After that they were used to test real network traffic data collected from servers of our university. PHAD was also used to test these data for intrusions.

Collected network traffic data contained various attacks distributed into samples and a few attack waves which last longer than other attacks. Since Snort and Spade were not affected underlying structure or content of the network, demonstration has been completed. However PHAD algorithm required clean training data obtained from the network which it is deployed. This type of data was not available in large amounts, so the process ended with many false alerts, a sign of tight bounds between algorithm and structure and content of the network. Demonstration of other algorithms in real traffic was cancelled because of similar algorithmic background.

SPADE has proven to be useful for detecting port scans but has a serious lack of event correlation ability. Performance of the selected detector of SPADE has been superseded by default port-scan preprocessor plug-in of Snort package. Not all detected probes of SPADE and port-scan overlap each other, thus providing more information on activities missed by other detector. SPADE has been more informative and precise on scanned ports than port-scan.

As future work, evaluation of these algorithms may be performed with more real attack-free training data. Demonstrating performance of other detectors of SPADE may be added to the overall process.

REFERENCES:

Please note that last access dates are in MM.DD.YYYY format for web sources.

Allen, A.O., 1997. *Probability, Statistics and Queuing Theory with Computer Science Applications, 2nd Edition*, Academic Press Inc.

Allen, W.H. and Marin, G.A., 2003. "On the Self-similarity of Synthetic Traffic for the Evaluation of Intrusion Detection Systems", Proceedings of the 2003 Symposium on Applications and the Internet (SAINT'03)

Anderson, J., 1980. "Computer Security Threat Monitoring and Surveillance", James P. Anderson Co., Fort Washington, PA.

Axelsson S., 1999, "On a difficulty on Intrusion Detection", 1999, Proceedings of the Second International Workshop on Recent Advances in Intrusion Detection, W. Lafayette, Indiana, 1999.

Axelsson S., 2000. "Intrusion Detection Systems: A Survey and Taxonomy". Technical Report 99-15, Depart. of Computer Engineering, Chalmers University, march 2000

Aydin, M.A. and Orencik B. 2005. "Bilgisayar Ağlarında Saldırı Tespiti için İstatistiksel Yöntem Kullanımı ve Bir Karma Saldırı Tespit Sistemi Tasarımı", Proceedings of First Symposium on Network and Information Security, Istanbul

Biles S., "Detecting the Unknown with Snort and the Statistical Packet Anomaly Detection Engine (SPADE)", Technical Report, 01.01.2006, available via web address http://www.computersecurityonline.com/spade/SPADE.pdf

Bykova, M., Ostermann, S. and Tjaden, B., 2001. "Detecting Network Intrusions via a Statistical Analysis of Network Packet Characteristics", Proceedings of the 33rd Southeastern Symposium on System Theory.

Cabrera J.B.D, Ravichandran B. and Mehra, R.K. 2000, “Statistical Traffic Modeling for Network Intrusion Detection”,.in Proceedings of the Eighth International 13th Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, pages 466--473, San Francisco, CA, August 2000. IEEE Computer Society

Cleary J.G. and Witten I.H., 1984, “Data Compression using adaptive coding and partial string matching”, *IEEE Transactions on Communication*, vol.COM-32, no.4, pp. 396-402

Cole, E., Krutz, R. and Conley, J.W, 2005. *Network Security Bible*, Wiley Publishing Inc.

Crothers T., 2003. *Implementing Intrusion Detection Systems A Hands on Guide for Securing the Network*, Wiley Publishing

CSI/FBI, 2005. “Computer Crime and Security Survey”, Computer Security Institute

Das K., 2000. “Attack Development for Intrusion Detection Evaluation”, Master Thesis, Massachusetts Institute of Technology

Denning, D. 1987. An Intrusion Detection Model. *IEEE Transactions on Software Engineering* 13, 2 (Feb.), 222--232

Haines, J.W., Lippmann, R.P., Fried, D.J., Zissman M.A., Tran, E. and Boswell S.B., 2001, “1999 DARPA Intrusion Detection Evaluation: Design and Prıcedures”, MIT Lincoln Laboratory Technical Report, TR-1062, Massachusets, USA

Heberlein L. T., Levitt K. N. and Mukherjee B.. “A Method To Detect Intrusive Activity in a Networked Environment” Proceedings of the 14th National Computer Security Conference, pages 362 371, October 1991.

Hoagland J., Staniford S., 2000. “Statistical Packet Anomaly Detection Engine”

Kendall K., 1999 “A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems”, Master Thesis, Lexington, Massachusetts

Kruegel, C. and Vigna, G., 2003. “Anomaly Detection of Web based Attacks”, Proceedings of the 10th ACM conference on Computer and communications security p.251-261.

Leland, W.E., Taqqu, M.S., Willinger W. and Wilson, D.V., 1994. “On the Self-Similar Nature of Ethernet Traffic (Extended Version)”, *IEEE/ACM Transactions on Networking* Vol. 2, No 1, Feb 1994

Lippmann R., Haines, J.W., Fried, D.J., Korba, J. and Das K., 2000a. “Analysis and Results of the 1999 DARPA Off-Line Intrusion Detection Evaluation”, Proceedings od Recent Advances in Intrusion Detection: Third International Workshop, RAID 2000, Toulouse, France, October 2000

Lippmann R.,Haines J.W., Fried, D.J., Korba J. and Das K., 2000b. “The 1999 DARPA Off-Line Intrusion Detection Evaluation”, *Computer Networks*, 34(4), p579-595

Mahoney M.V., Chan P.K., 2001. “PHAD: Packet Header Anomaly Detection for Identifying Hostile Network Traffic”, Florida Inst. of Tech. Technical Report, CS-2001-04

Mahoney, M.V. and Chan P.K, 2002a, “Learning Nonstationary Models of Normal Network Traffic for Detecting Novel Attacks”, Proceedings of ACM Eighth International Conference on Knowledge Discovery and Data Mining (SIGKDD), p376-385.

Mahoney M.V., Chan P.K., 2002b. “Learning Models of Network Traffic for Detecting Novel Attacks”, Florida Institute of Technology Technical Report CS-2002-08

Mahoney M.V., Chan P.K., 2003a. “Learning Rules for Anomaly Detection of Hostile Network Traffic”, Proc. Third International Conference on Data Mining (ICDM 2003), Melbourne FL

Mahoney M.V., Chan P.K., 2003b. “An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection”, Florida Institute of Technology Technical Report CS-2003-02.

Mahoney M.V., Chan P.K., 2003c. “Network Packet Anomaly Detection Based on Packet Bytes”, Proceedings of ACM-SAC, Melbourne FL, p. 346-350

Mahoney M.V., 2003. “A Machine Learning Approach to Detecting Attacks by Identifying Anomalies in Network Traffic”, Ph.D. Thesis dissertation, TR-CS-2003-13

Marchette, D. J., 2001. *Computer Intrusion Detection and Network Monitoring : A Statistical Viewpoint*, Springer-Verlag

McHugh J., 2000, “Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 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.

Paxson, V., Floyd, S., 1995. “The Failure of Poisson Modeling”, *IEEE/ACM Transactions on Networking* Vol.3 No:3 p.226-244.

Paxson, V. and Floyd, S., 1997. “Why we don't know how to simulate the internet”. In Proceedings of the 1997 Winter Simulation Conference

Paxson V., 1998. “Bro: A System for Detecting Network Intruders in Real-Time”, Proceedings of 7th USENIX Security Symposium.

Pfleeger, C.P., 1997. *Security in Computing 2nd Edition*, Prentice Hall PTR,

Roesch M., 1999. “Snort – lightweight intrusion detection for networks”, in Proceedings 13th USENIX Systems Administration Conference (LISA ’99), Seattle, WA, Nov. 1999

Shon, T., Kim., Y., Lee, C. and Moon, J., 2005. “A Machine Learning Framework for Network Anomaly Detection using SVM and GA”, Proceedings of the 2005 IEEE Workshop on Information Assurance and Security, United States Military Academy, West Point, NY.

Sourcefire Inc., 2006. “Snort Users Manual 2.4.0 RC1”

Stallings, W., 2003. *Network Security Essentials Applications and Standards 2nd Edition*, Pearson Education Inc.

Stevens W.R., 1994. *TCP/IP Illustrated Vol.1: Protocols*, Addison Wesley Publishing

Tanenbaum, A.S., 1996, *Computer Networks 3rd Edition*, Prentice Hall

Witten, I.H., Bell, T., 1991. “The Zero-Frequency Problem: Estimating the Probabilities of Novel Events in Adaptive Text Compression”, *IEEE Transactions on Information Theory*, vol.37, No:4

Ye, N., Li, X., Chen, Q., Emran, S.M., Xu, M., 2001. “Probabilistic techniques for intrusion detection based on computer audit data”, *IEEE Transactions on Man and Cybernetics, Part A: Systems and Humans*, Vol.31, No:4

Ye N. and Chen, Q., 2001. “An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems”, *Quality and Reliability Engineering International* Vol. 17, No:2 **,** Pages 105 – 112

Ye, N., Emran, S.M., Chen, Q. and Vilbert, S., 2002., “Multivariate statistical analysis of audit trails for host-basedintrusion detection” *IEEE Transactions on Computers*, Vol.51., No:7, July 2002

Yin C., Tian S., Huang H., He J., 2005. “Applying Genetic Programming to Evolve Learned Rules for Network Anomaly Detection”, Proceedings of First International Conference on Advances in Natural Computation, ICNC 2005, Changsha, China, August 27-29, Part III , p.323

(WEB_1 2006) Morris Worm article in Wikipedia, 07.01.2006, http://en.wikipedia.org/wiki/Morris_worm

(WEB_2 2006) CERT/CC Statistics 1998-2006, 07.01.2006, http://www.cert.org/stats/

(WEB_3 2006) Anomaly Web Article in Wikipedia, 07.01.2006, http://en.wikipedia.org/wiki/Anomaly

(WEB_4 2006) SHADOW Intrusion Detection System, 07.01.2006, http://www.nswc.navy.mil/ISSEC/CID/index.html

(WEB_5 2006) MIT Lincoln Laboratory IDEVAL Home Site, 07.01.2006, http://www.ll.mit.edu/IST/ideval/

(WEB_6 2006) Self-Similarity article From MathWorld by Eric Weisstein, 07.01.2006, http://mathworld.wolfram.com/Self-Similarity.html

(WEB_7) Snort Home Site, 07.01.2006, http://www.snort.org

(WEB_8 2006): Tcpdump.org web site (also libpcap’s web site), 07.01.2006, http://www.tcpdump.org

(WEB_9) Netfilter Web Site (also iptables’ web site), 07.02.2006, http://netfilter.org

(WEB_10) Good Turing Estimator Web Article in Wikipedia, 07.01.2006, http://en.wikipedia.org/wiki/Good-Turing_estimator

(WEB_11 2006) The Good-Turing Estimator, Geomblog Web Resource, 07.01.2006, http://geomblog.blogspot.com/2005_07_01_geomblog_archive.html

(WEB_12) Bleeding Edge Snort Community Home Site, 07.01.2006, http://www.bleedingsnort.com

(WEB_13 2006) History of the Finger Protocol, 07.02.2006, http://www.rajivshah.com/Case_Studies/Finger/Finger.htm

(WEB_14 2006) Source Code for PHAD, ALAD, LERAD, NETAD and EVAL. 07.01.2006, http://www.cs.fit.edu/~mmahoney/dist/

(WEB_15 2006) A Frequently Asked Questions (FAQ) on Web Bugs, 06.01.2006, http://www.eff.org/Privacy/Marketing/web_bug.html

(WEB_16 2006) A Tutorial on Byte Orders, 07.01.2006, http://www.netrino.com/Publications/Glossary/Endianness.html

APPENDIX A

FIELDS OF LOWER LAYER PROTOCOL HEADERS

Five anomaly detection algorithms covered in Chapter 4 use different fields on lower layer protocols. These fields in lower layer protocols and their order in packet payloads are provided in the following figures

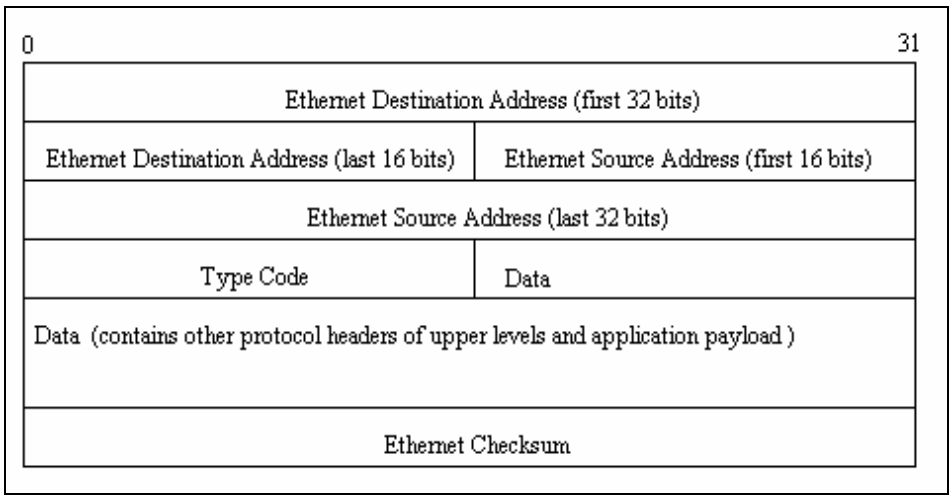

Fig. A.1.Fields of Ethernet Packet

(Redrawn, Original source: Stevens 1994)

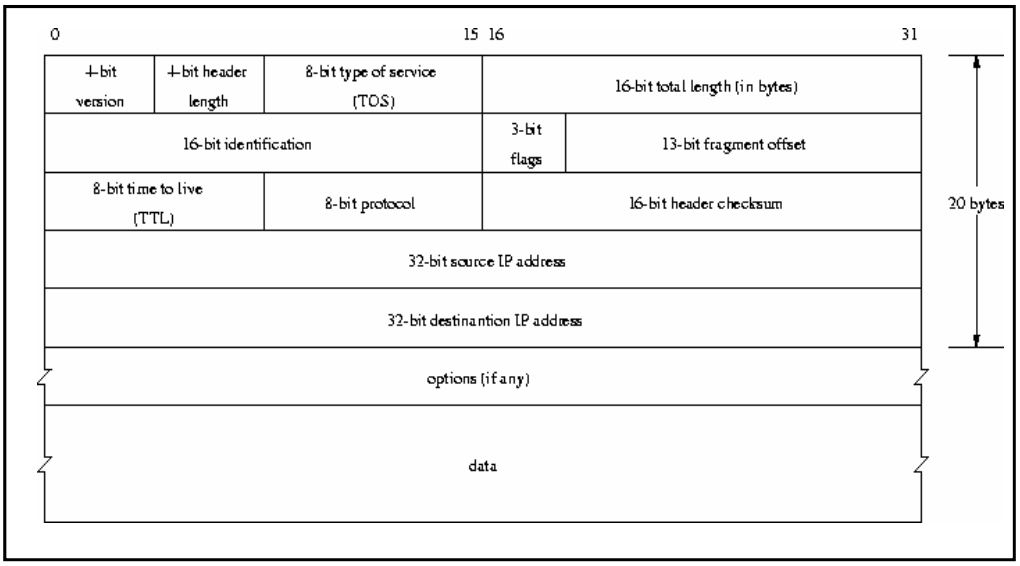

Fig. A.2.Fields of IP Header

(Source: Stevens 1994)

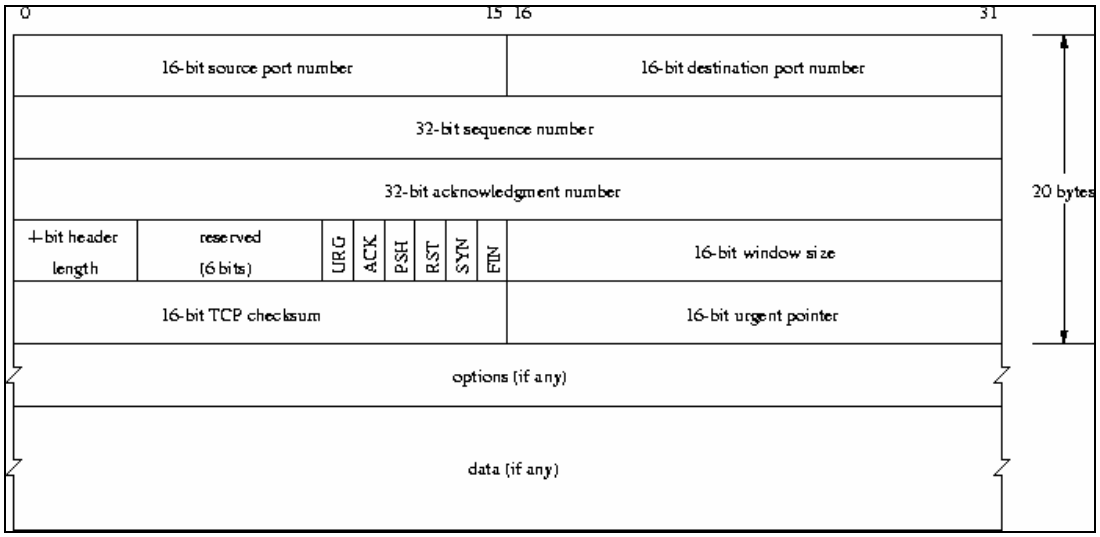

Fig. A.3.Fields of TCP Header

(Source: Stevens 1994)

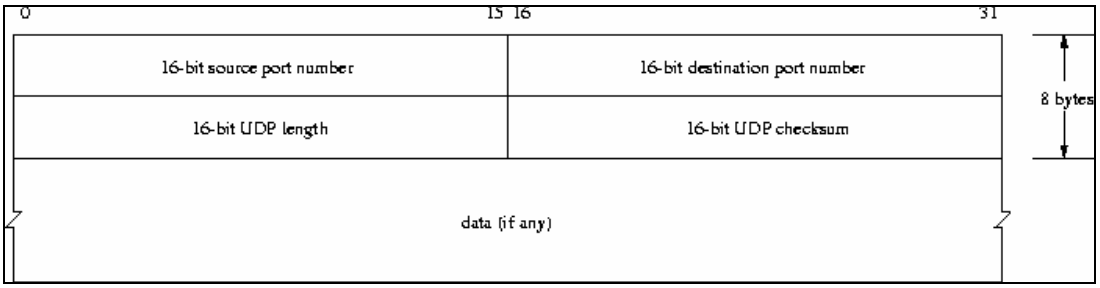

Fig. A.4.Fields of UDP Header

(Source: Stevens 1994)

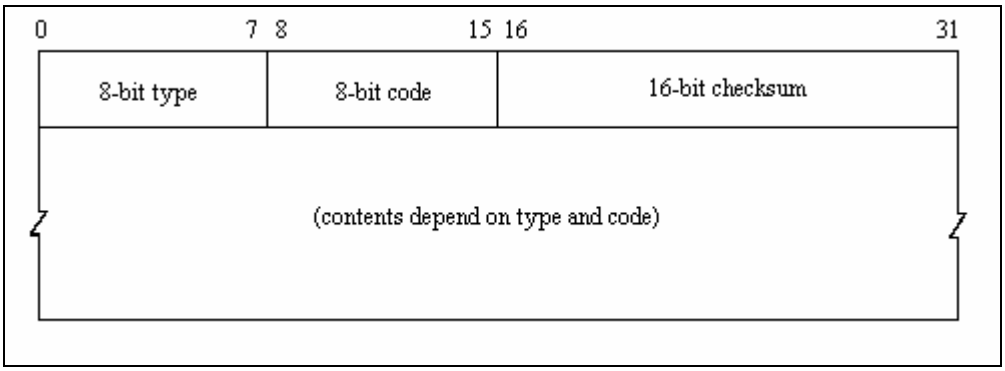

Fig.A.5.ICMP Header

(Redrawn, Original source: Stevens 1994)

APPENDIX B

LIST OF DATA FILES

Table B.1 shows list of available data files. File names have no extension. Data files of the same day are grouped and summed in empty line after each group.

Table B.1. List of tcpdump data files collected by Snort

	File Name	Starts at (* Real time)	Ends at (* Real time)	Length(MB)
1	snort-tcpdump-6-1	Thu Apr 6 14:32:14	Thu Apr 6 16:38:18	1.200,00
2	snort-tcpdump-6-2	Thu Apr 6 16:38:36	Thu Apr 6 16:38:46	0,09
3	snort-tcpdump-6-3	Thu Apr 6 16:38:58	Thu Apr 6 23:59:59	783,50
				1.983,59
4	snort-tcpdump-7-1	Fri Apr 7 00:00:00	Fri Apr 7 14:26:42	1.200,00
5	snort-tcpdump-7-2	Fri Apr 7 15:16:41	Fri Apr 7 16:25:01	1.100,00
6	snort-tcpdump-7-3	Fri Apr 7 16:25:01	Fri Apr 7 20:00:01	954,80
7	snort-tcpdump-7-4	Fri Apr 7 20:00:01	Fri Apr 7 23:59:59	128,30
				3.383,10
8	snort-tcpdump-8-1	Sat Apr 8 00:00:00	Sat Apr 8 16:40:01	1.300,00
9	snort-tcpdump-8-2	Sat Apr 8 16:40:01	Sat Apr 8 23:59:59	305,70
				1.605,70
10	snort-tcpdump-9-1	Sun Apr 9 00:00:00	Sun Apr 9 16:30:01	1.300,00
11	snort-tcpdump-9-2	Sun Apr 9 16:30:02	Sun Apr 9 22:20:01	979,50
12	snort-tcpdump-9-3	Sun Apr 9 22:20:02	Sun Apr 9 23:59:59	59,20
				2.338,70
13	snort-tcpdump-10-1	Mon Apr 10 00:00:00	Mon Apr 10 11:20:01	947,10
14	snort-tcpdump-10-2	Mon Apr 10 11:20:01	Mon Apr 10 16:30:01	1.600,00
15	snort-tcpdump-10-3	Mon Apr 10 16:30:01	Mon Apr 10 17:10:01	1.200,00
16	snort-tcpdump-10-4	Mon Apr 10 17:10:01	Mon Apr 10 17:20:01	1.400,00
17	snort-tcpdump-10-5	Mon Apr 10 17:20:01	Mon Apr 10 23:59:59	371,10
				5.518,20
18	snort-tcpdump-11-1	Tue Apr 11 00:00:00	Tue Apr 11 10:10:01	784,20
19	snort-tcpdump-11-2	Tue Apr 11 10:10:01	Tue Apr 11 16:40:01	1.800,00
20	snort-tcpdump-11-3	Tue Apr 11 16:40:02	Tue Apr 11 23:59:59	331,00
				2.915,20
21	snort-tcpdump-12-1	Wed Apr 12 00:00:00	Wed Apr 12 10:40:01	637,00
22	snort-tcpdump-12-2	Wed Apr 12 10:40:01	Wed Apr 12 16:30:01	1.800,00
23	snort-tcpdump-12-3	Wed Apr 12 16:30:01	Wed Apr 12 23:59:59	741,90
				3.178,90

	File Name	Starts at (* Real time)	Ends at (* Real time)	Length(MB)
24	snort-tcpdump-13-1	Thu Apr 13 00:00:00	Thu Apr 13 08:50:01	214,40
25	snort-tcpdump-13-2	Thu Apr 13 08:50:01	Thu Apr 13 12:20:01	1.016,80
26	snort-tcpdump-13-3	Thu Apr 13 12:20:01	Thu Apr 13 12:40:01	1.300,00
27	snort-tcpdump-13-4	Thu Apr 13 12:40:01	Thu Apr 13 14:10:01	1.500,00
28	snort-tcpdump-13-5	Thu Apr 13 14:10:01	Thu Apr 13 16:00:01	1.200,00
29	snort-tcpdump-13-6	Thu Apr 13 16:00:01	Thu Apr 13 16:30:01	1.500,00
30	snort-tcpdump-13-7	Thu Apr 13 16:30:01	Thu Apr 13 23:59:59	849,30
				7.580,50
31	snort-tcpdump-14-1	Fri Apr 14 00:00:00	Fri Apr 14 04:00:01	105,20
32	snort-tcpdump-14-2	Fri Apr 14 04:00:01	Fri Apr 14 10:50:01	1.000,00
33	snort-tcpdump-14-3	Fri Apr 14 10:50:01	Fri Apr 14 16:10:01	1.500,00
34	snort-tcpdump-14-4	Fri Apr 14 16:10:02	Fri Apr 14 16:40:01	1.500,00
35	snort-tcpdump-14-5	Fri Apr 14 16:40:01	Fri Apr 14 23:59:59	787,70
				4.892,90
36	snort-tcpdump-15-1	Sat Apr 15 00:00:00	Sat Apr 15 08:00:01	170,60
37	snort-tcpdump-15-2	Sat Apr 15 08:00:01	Sat Apr 15 16:30:01	1.200,00
38	snort-tcpdump-15-3	Sat Apr 15 16:30:01	Sat Apr 15 23:59:59	355,40
				1.726,00
39	snort-tcpdump-16-1	Sun Apr 16 00:00:00	Sun Apr 16 16:30:01	1.300,00
40	snort-tcpdump-16-2	Sun Apr 16 16:30:01	Sun Apr 16 23:59:59	287,70
				1.587,70
41	snort-tcpdump-17-1	Mon Apr 17 00:00:00	Mon Apr 17 11:00:01	668,30
42	snort-tcpdump-17-2	Mon Apr 17 11:00:01	Mon Apr 17 16:40:01	1.700,00
43	snort-tcpdump-17-3	Mon Apr 17 16:40:01	Mon Apr 17 17:14:16	77,30
				2.445,60

APPENDIX C

CONTENTS OF CD

root directory:

- binaries (dir)
- extras (dir)
- TITLE.doc
- original (dir)
- programs (dir)
- README.txt
- sources (dir)
- TABLE OF CONTENTS.doc
- THESIS.doc

binaries directory:

- ethereal-0.99.0-fc4.1.i386.rpm
- ethereal-gnome-0.99.0-fc4.1.i386.rpm
- Files.txt
- libnet10-1.0.2a-8.fc4.i386.rpm
- libpcap-0.8.3-14.FC4.i386.rpm
- pcre-5.0-4.1.fc4.i386.rpm
- pcre-devel-5.0-4.1.fc4.i386.rpm
- snort-2.4.4-3.fc4.i386.rpm
- tcpdump-3.8.2-14.FC4.i386.rpm
- unrar-3.5.4-0.lvn.1.4.i386.rpm

extras directory:

- snortrules-snapshot-CURRENT.tar.gz
- tarihler.iztech.txt
- tethereal.manual.txt

original directory:

a2l.txt
alad.txt
eval.cpp
IDS Distribution.htm
lerad.cpp
leradp.cpp
netad.cpp
phad.cpp
sad.cpp
te.cpp
tf.cpp

programs directory:

Files.txt
libpcap-0.9.4.tar.gz
snort-2.4.4.tar.gz
spade.tar.gz
tcpdump-3.9.4.tar.gz
tcpslice-1.1a3.tar.gz
tcpslice_mod.tar.gz

sources directory:

doopen-final.sh
Files.txt
generate-final.sh
Makefile
myenhtest-final.pl
mytest2-final.pl
phadm.cpp
pssum-final.c
saviour-final.sh
saviour sliced-final.sh