An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection

TR CS-2003-02

Matthew V. Mahoney and Philip K. Chan

Computer Science Department, Florida Institute of Technology
150 W. University Dr., Melbourne, Florida 32901
{mmahoney, pkc}@cs.fit.edu

Abstract. We investigate potential simulation artifacts and their effects on the evaluation of network anomaly detection systems in the 1999 DARPA/MIT Lincoln Laboratory off-line intrusion detection evaluation data set. A statistical comparison of the simulated background and training traffic with real traffic collected from a university departmental server suggests the presence of artifacts that could allow a network anomaly detection system to detect some novel intrusions based on idiosyncrasies of the underlying implementation of the simulation, with an artificially low false alarm rate. The evaluation problem can be mitigated by mixing real traffic into the simulation. We compare five anomaly detection algorithms on simulated and mixed traffic. On mixed traffic they detect fewer attacks, but the explanations for these detections are more plausible.

1. Introduction

Signature detection seeks to identify known attacks and, by definition, it is always one step behind the attackers. Anomaly detection can potentially detect novel attacks. Evaluating signature detectors is easier than evaluating anomaly detectors. Signature detectors can be tested against known attacks, probably with more emphasis on more recent attacks since older attacks become relatively harmless after vulnerabilities have been fixed. However, it is implausible to predict future attacks (if we can, we do not need intrusion detection). Hence, anomaly detection is usually tested against known attacks, of which the algorithms have no apriori knowledge. Since anomaly detectors model normal behavior and has no knowledge of the attacks, the recency of known attacks is much less, if at all, significant than in the evaluation of signature detectors.

Scientific advances rely on reproducibility of results so that they can be independently validated and compared. Much of the evaluation in intrusion detection has been based on proprietary data and results are generally not reproducible. One of the main problems of releasing data stems from privacy concerns. To reduce this problem, Lincoln Laboratory (LL), under sponsorship of DARPA, created the IDEVAL datasets that serves as an evaluation benchmark [1-3]. With large amounts of careful considerations, the LL team designed a behavior model for users, synthesized the behavior, and recorded the real network traffic (and other events) caused by the synthetic behavior. To our knowledge, these datasets also constitute the largest publicly available benchmark. McHugh [4] has

reported some of the problems with the datasets, but they continue to be the largest publicly available and most sophisticated benchmark for researchers. In this study, we try to understand in a more detailed manner some of the shortcomings and provide a procedure that allows other researchers to evaluate their techniques using the benchmark without the weaknesses identified by us. As explained above, the recency of attacks is much less significant in evaluating anomaly detectors than signature detectors, hence the IDEVAL dataset in this study, which was collected in 1999, is still viable for evaluating anomaly detectors.

Our study concerns the suitability of the IDEVAL background network traffic, which was synthesized, for evaluating anomaly detection systems. This background traffic is used to build a model of "normal" behavior, so that deviations can be reported as suspicious. McHugh notes that some aspects of the process of generating the background traffic were described only superficially, that there was no validation of the synthesized traffic in comparison to real traffic with respect to false alarms that an anomaly detection system might generate, and no procedure for inserting badly behaved data which is known to exist in real traffic. Our more detailed analysis of the data confirms McHugh's speculation that this data have statistically different characteristics from real traffic, and we suggest an approach to mitigate the problem.

Our key contributions are as follows:

- We demonstrate that simulation artifacts can easily be exploited to detect attacks in IDEVAL (Section 3).
- We compare the IDEVAL background traffic to real traffic, and find that the IDEVAL data is too predictable and "clean" (Section 4).
- We propose a strategy for making the IDEVAL background more realistic by injecting real traffic (Section 5).
- We evaluate five anomaly detection algorithms (SPADE [5], and four that we developed), and show that most questionable detections in IDEVAL do not appear when using mixed traffic (Section 6).

2. Related Work

The DARPA/MIT Lincoln Lab evaluation (IDEVAL) data set has been used to test a large number of intrusion detection systems [6-22], both in the 1998 and 1999 blind evaluations, and in subsequent development work after the evaluation data was released. The data can be used to test both host based and network based systems, and both signature and anomaly detection systems. The 1998 network data has also been used to develop the 1999 KDD cup machine learning competition [23], which had 25 participants, and which continues to be used to test intrusion detection methods [24-25].

McHugh [4] criticizes many aspects of the DARPA/LL evaluation, which includes questionable collected traffic data, attack taxonomy, and evaluation criteria. With respect to the collected traffic data, he criticizes the lack of statistical evidence of similarity to typical Air Force network traffic, low traffic rates, lack of “crud” (idiosyncrasies), relative uniform distribution of the four major attack categories, skewed distribution of victim hosts, and flat

network topology. The critique is not accompanied with answers to many of the questions it raised. This paper investigates some of the evidence that indicates the IDEVAL dataset is statistically different from the real data we collected and proposes a procedure to help reduce the problem.

In our prior work with network anomaly detection [11-14], we discovered that some attacks appeared to be detectable in IDEVAL due to simulation artifacts. In particular:

- PHAD [12] detects many attacks because the hostile IP packets have a TTL value which is lower by 1 than the background traffic. We believe that this is due to the underlying configuration of the simulation, in which attacking traffic and background traffic (even with the same IP address) were synthesized on different physical machines [3].
- ALAD [14], LERAD [13], and NETAD [11] detect a large number of attacks by anomalous source addresses, including attacks on DNS, web and mail servers, where previously unseen addresses should be the norm.
- NETAD detects several attacks by anomalies in the TCP window size field, without a reasonable explanation for why these anomalies should occur.

Unfortunately, we lack alternative data sets for our work. Public collections of real traffic such as the Internet Traffic Archive [26], and the University of New Mexico data set [27], are stripped of much of the data useful to anomaly detection, including all of the application payload data. Payload data, which could include private email and passwords, is always stripped off to protect the privacy of real users.

IDEVAL uses simulated traffic for this reason. Unfortunately, Internet traffic is very hard to simulate correctly [28]. There are a great many sources of network behavior (hardware, software, and people), all with their idiosyncrasies. Some types of events display long range dependencies, making it difficult to reliably sample their average rates no matter how long the sample period [29-30]. Data does not always conform to protocols, resulting in unexplained malformed packets or "crud" [31]. Still, the IDEVAL developers went to great lengths to produce an accurate simulation, such deliberately injecting errors, synthesizing email messages and data files using English word bigram statistics, and using public mailing lists and websites. The mix of protocols and hourly traffic rates are based on traffic collected from a real Air Force base network in 1998 [2-3].

3. Simulation Artifact Detection

To motivate our investigation of simulation artifacts in the IDEVAL data, we develop a very simple anomaly detection system that could not possibly work. We call our system SAD (Simple Anomaly Detector, available at [32]). We evaluate SAD using the 1999 IDEVAL data set. This set includes 5 weeks of sniffed traffic (*tcpdump* files) from two points in a simulated network, one "inside" sniffer, between the gateway router and four "victim" machines, one "outside" sniffer between the gateway and the simulated Internet, and host based audit data collected nightly from the four victims. The five weeks are as follows:

- Weeks 1 and 3 – no attacks (for training anomaly detection systems).

- Week 2 – 43 instances of 18 labeled attacks used for system development.
- Weeks 4 and 5 – 201 instances of 58 attacks (40 new), used for evaluation.

In 1999, 8 organizations submitting 18 systems were provided with data from weeks 1-3 to develop their systems. Three months later, the organizations were given weeks 4-5 and asked to provide a list of alarms. Each alarm should identify the date and time of an attack, the victim IP address, and a numerical score indicating the relative certainty of an attack. Systems were ranked by the percentage of attacks detected out of those the system was designed to detect (based on the data examined and the type of attack) at a score threshold allowing no more than 100 false alarms (10 per day). An attack is counted as detected if any alarm correctly identifies the target (or any target if more than one) and the time of any portion of the attack with 60 seconds leeway. Duplicate detections of the same attack are ignored, as are detections of "out-of-spec" attacks that the system is not intended to detect. Any other alarm is a false alarm. The top four systems in 1999 detected between 40% and 55% of the attacks by this criteria. The best systems generally used both signature and anomaly detection on both host and network data.

Our system, SAD, examines only inbound TCP SYN network packets (destination address 172.16.x.x in IDEVAL). It looks at just one byte of the packet, specified as a parameter, for example the TTL field (time to live – an 8-bit counter used to expire packets caught in misconfigured router loops). During training, SAD records which of the 256 possible values are seen at least once and which are not. During testing, it detects an anomaly if this byte has a value that was never seen in training. If there have been no other anomalies in the last 60 seconds, then it outputs an alarm with a score of 1 warning that the packet is hostile.

We train SAD on the IDEVAL inside sniffer traffic from week 1 and test it on week 2 to identify which packet bytes might be useful for anomaly detection. We evaluate SAD (using EVAL [32], our implementation of the IDEVAL detection criteria) and identify several promising SAD variations, which we define as any variation that detects at least one of the 43 attacks for every 10 false alarms (Table 1, second column). Then we evaluate these variants on the actual test data by training them on inside week 3, and testing on weeks 4 and 5, which contain evidence of 177 of the 201 attacks used in the published evaluation results. We consider these 177 attacks to be in-spec, consistent with the rules of the 1999 evaluation. Almost all of these variations would have done well in this evaluation (Table 1, third column). The best SAD variant, which examines the third byte of the source IP address, detects 79 of 177 attacks (45%), with 43 false alarms. This result is competitive with the top systems in the original evaluation.

However, these results are misleading. To test how SAD might behave in a real network, we mix the 146 hours of training traffic from week 3 and 197 hours of test traffic from weeks 4-5 with equal durations of (presumably attack-free) traffic collected from a university departmental server on a large network. We mix the traffic by shifting the timestamps to make it appear as if the web server is part of the home network. No other fields (e.g. IP addresses) are changed. The mixed traffic contains 154,057 simulated and 125,834 real inbound TCP SYN packets.

As we should expect, the results are quite poor (Table 1, last column). Very few attacks are detected, and the false alarm rate is much higher. But a more detailed analysis shows that these results make more sense. For example, on the simulated data SAD detects

source address anomalies in attacks on public web servers (*apache2, back, crashiis*, and *phf*), mail servers (*mailbomb*, *sendmail*),. DNS (*ls_domain*), and anonymous FTP (*guessftp, warez*), where novel addresses should be normal. However, on the mixed data, the only attack detected is *neptune*, which spoofs the source address with an unassigned portion of the IP address space (10 or 11 in the first byte). Likewise, most of the other packet header anomalies detect only attacks that require the attacker to write (usually arbitrary) values into those fields.

But why did SAD do so well in the first place? In the next section, we compare the simulated training traffic with our real traffic to shed some light on this question.

Table 1. SAD detections and false alarms for variants that do well on the 43 attacks in weeks 1-2 of the 1999 IDEVAL IDS evaluation inside sniffer traffic. Detections and false alarms are shown for weeks 1-2 (43 attacks), weeks 3-5 (177 attacks) and for weeks 3-5 mixed with real traffic

SAD Byte Detections/False Alarms	**Weeks 1-2**	**Weeks 3-5**	**Mixed**
IP packet size, low byte	4/0	15/2	0/1
TTL	25/36	24/4	5/43
Source IP address, 1st byte	13/7	64/41	4/0
Source IP address, 2nd byte	13/7	67/42	0/0
Source IP address, 3rd byte	16/15	79/43	0/0
Source IP address, 4th byte	17/14	71/16	0/0
Source port, high byte	2/0	13/0	0/0
Destination port, high byte	4/24	4/0	4/1664
Destination port, low byte	5/6	0/0	0/0
TCP header size	4/0	15/2	0/5
TCP window size high byte	5/1	15/2	7/112
TCP window size, low byte	3/1	7/1	4/29
TCP options, bytes 1, 2, 3	4/4	15/2	0/1
TCP options, byte 4	4/4	15/2	0/255

4. Simulated vs. Real Traffic

In this section, we compare the IDEVAL training data (inside sniffer weeks 1 and 3) with real traffic collected from a similar environment, a university Solaris machine which is the main server for the CS department (www.cs.fit.edu), with several faculty user accounts and serving several thousand web pages. We collected 50 traces of 2,000,000 packets each on weekdays (Monday through Friday) over a 10 week period from September through December 2002. Each trace was started at midnight and lasts 10 to 15 hours. Packets were truncated to 200 bytes. We also collected two smaller traces of 1,000,000 packets each in November, 2001. Our environment differs in the following respects:

- The Ethernet traffic is switched, so only traffic to and from the host (and multicast traffic) is visible.

- Some hosts on the local network use dynamic IP addresses which can change daily.
- The real local network is protected by a firewall.
- There is no *telnet* traffic and very little FTP traffic. The real host instead supports secure shell and secure FTP. Non secure FTP data is not transported on the standard port (20) normally used for this purpose.

However, most of the other protocols found in the IDEVAL background data are found in the real traffic. These include Ethernet, ARP, IP, TCP, UDP, ICMP, HTTP, SMTP, POP3, IMAP, *nbname*, *nbdatagram*, DNS, NTP, *auth* and *printer*. The real traffic has many protocols not found in IDEVAL: IGMP, OSPFIGP, PIM, NFS, RMI, *portmap*, and some obscure and undocumented protocols.

Although the real traffic is protected by a firewall, it is not free of attacks. We manually identified 33 suspicious HTTP web server requests. Here are two examples:

```
GET /MSADC/root.exe?/c+dir HTTP/1.0
```

This is a probe for a backdoor dropped by the *code red* worm.

```
GET /scripts/..%255c%255c../winnt/system32/cmd.exe?/c+dir
```

This appears to probe for an IIS vulnerability. It uses double URL encoding to mask the suspicious string "`/../`" in the URL. The string "`%25`" encodes the "`%`" character, and "`%5c`" encodes a backslash, "`\`". Windows accepts either "`\`" or "`\\`" in place of "`/`" in a file path. Neither attack succeeded because the target is a UNIX machine.

We also identified one suspicious SMTP request, "`EXPN root`". EXPN (request user information) is disabled on our server.

4.1. Analysis

We compared our real traffic with IDEVAL attack-free traffic (inside sniffer weeks 1 and 3) from the standpoint of events that might be meaningful to a network anomaly detection system. Systems such as SPADE [5] and ADAM [33] examine the IP addresses and port numbers of inbound TCP client requests to local servers, and flag unusual values, based on their average frequency in training. *eBayes* [20] also models time-aggregate attributes such as error intensity or number of different ports seen during a time window. In our own work, we also examine Ethernet IP, TCP, UDP, and ICMP packet header fields (e.g. PHAD [12]), application payload bytes (NETAD [11]), and keywords of reassembled TCP streams in inbound client traffic (ALAD [14] and LERAD [13]).

Our systems flag values which have never been seen in training, especially when the attribute proved to be highly predictable during the past. Specifically, we assign a score of tn/r, where t is the time since the last anomaly, n is the number of training instances, and r is the number of different values seen in training. For example, given a training sequence "AABBB", and the anomalous value "C", we have $t = 3$ (the last anomaly was the first B), $n = 5$ (there are 5 training instances), and $r = 2$ (A and B). A system based on event

frequency, such as SPADE, would assign an anomaly score of $1/p$, where p is the probability based on past history (e.g. 1/6 because C occurs 1 out of 6 times). In either case, the novel value "C" generates a high anomaly score because it is seen for the first time. Thus, whichever model we use, we are interested in the frequency of novel values, r and the rate at which it increases over time, because these events would generate false alarms. Also, a higher value of r in real traffic could mean that some attacks detected in IDEVAL would be missed in real traffic because the anomalous values they generate are actually values that occur naturally, i.e. the IDEVAL detections were due to simulation artifacts.

4.2. Findings

We compared inbound client traffic from IDEVAL inside sniffer traffic from weeks 1 and 3 with the real traffic and found that many attributes have a higher value of r in real traffic, and furthermore, that this value tends to level off in IDEVAL but continue to grow in real traffic. This means that the attribute would generate more false alarms in real traffic, or could not be used. The following is a summary of our findings.

TCP SYN Regularity. In all 50,650 inbound TCP SYN packets in IDEVAL weeks 1 and 3 (the initial packet to the server), there are always exactly 4 option bytes (MSS = 1500). In the real traffic (210,297 inbound TCP SYN packets, not including 6 TCP checksum errors), we found 103 different values for the first 4 option bytes alone. The number of option bytes varies from 0 to 28. Also the simulated window size field always has one of 7 values (from 512 to 32,120) in IDEVAL, but we found 523 different values in real traffic, covering the full range from 0 to 65,535.

Source Address Predictability. There are only $r = 29$ distinct remote TCP client source addresses in IDEVAL weeks 1 and 3. Half of these are seen in the first 0.1% of the traffic. Our real data has $r = 24{,}924$ unique addresses, of which 53% are seen only in the second half of the data, suggesting that r increases at a steady rate. Furthermore, 45% of these appear in only a single TCP session, compared to none in IDEVAL. These statistics are consistent with the power law distribution normally found in Internet addresses [34-35] in the real traffic only.

Checksums Errors. The 12 million packets of IDEVAL traffic from inside week 3 have no IP, TCP, UDP, or ICMP checksums. (We did not test week 1). About 0.02% of our real TCP and ICMP packets have checksum errors. This estimate is probably low because we could not compute checksums for truncated packets. We did not find any real UDP or IP checksum errors.

Packet Header Fields. Only 9 of the possible 256 TTL values were observed in IDEVAL. We observed 177 different values in the real traffic. For TOS, we observed 4 values in IDEVAL and 44 values in real traffic.

IP Fragmentation. Fragments were found only in real traffic (0.45% of packets). The DF (don't fragment) flag was set in all of these packets.

"Crud". The following events were observed in the real traffic (in packets with good checksums), but not in IDEVAL.

- Nonzero values in the TCP ACK field when the ACK flag is not set (0.02%).
- Nonzero values in the urgent pointer field when the URG flag is not set (0.02%).
- Nonzero values in the two TCP reserved flags (0.09%).
- Nonzero values in the 4 reserved bits of the TCP header size field (0.006%).

HTTP Requests. In IDEVAL, the 16,089 HTTP requests are highly regular, and have the form "GET *url* HTTP/1.0" followed by optional commands of the form "*Keyword: values*". There are 6 possible keywords (within the first 200 bytes of the first data packet, which is all we can compare). The keywords are always capitalized with a space after the colon and not before. In the "User-Agent" field, there are 5 possible values, all versions of *Mozilla* (Netscape or Internet Explorer).

In the 82,013 real HTTP requests, the version may be 1.0 or 1.1. (The data is newer). There are 8 commands: GET (99% of requests), HEAD, POST, OPTIONS, PROPFIND, LINK, and two malformed commands "No" and "tcp_close". There are 72 different keywords. Keywords are usually not always capitalized, and the spacing around the colon is occasionally inconsistent. Some keywords are misspelled (*Connnection* with 3 n's, or the correctly spelled *Referrer* instead of the usual *Referer*). Some keywords are malformed ("XXXXXXX:" or "~~~~~~~:"). A few request lines are missing a carriage return before the linefeed. There are 807 user agents, of which 44% appear only once. The top five are:

- *Scooter/3.2*
- *googlebot/2.1*
- *ia_archiver*
- *Mozilla/3.01*
- *http://www.almaden.ibm.com/cs/crawler*.

SMTP Requests. In IDEVAL, the 18,241 SMTP mail requests are always initiated with a HELO or EHLO (echo hello) command. There are 3 different HELO arguments (identifying the local sender host name) and 24 different EHLO arguments (identifying the remote sender host name), all but one of which appear at least twice. In the 12,911 real SMTP requests there are 1839 different HELO arguments (69%) appearing only once, and 1461 different EHLO arguments (58% appearing only once). In addition, 3% of real SMTP requests do not start with a HELO/EHLO handshake. (SMTP does not require it). Instead, they start with RSET, QUIT, EXPN, NOOP, or CONNECT. In none of the IDEVAL requests but 0.05% of the real requests, the SMTP command is lower case. Also, 0.1% of the real requests are malformed with binary data in the argument.

SSH Requests. An SSH request initiates with a string identifying the client version. In the 214 IDEVAL requests, the string is always "SSH-1.5-1.2.22". In the 666 real requests, there were 32 versions, of which 36% appeared only once.

4.3. Implications

Our findings suggest that many attributes of network traffic that appear to be predictable in IDEVAL are less predictable in real traffic. An anomaly detection rule based on one of these attributes should generate more false alarms in real traffic than in IDEVAL. The extent to which this leads to fewer detections depends on whether these attributes are important to anomaly detection, and if so, whether the values that indicate an attack in IDEVAL also occur naturally in real traffic. In examining the test data, we found examples of all three cases.

Detections Masked in Real Traffic. In IDEVAL, the unrelated attacks *back, dosnuke, neptune, netbus, netcat, ntinfoscan,* and *queso* have TTL values of 253 or 126, which never appear in the training traffic. In real traffic, these and many other values are likely to appear. Even if these attacks appear exactly as they do in IDEVAL, an anomaly detection system trained on real traffic is likely to miss them.

In IDEVAL, the only SMTP connections that do not start with HELO or EHLO are the two attacks *mailbomb* (a denial of service flood) and *sendmail* (a root shell buffer overflow). A rule requiring HELO/EHLO would generate false alarms in real traffic. Although it would be simple to conceal the attacks by adding a HELO/EHLO handshake (which is unrelated to the attack), we find that many other attacks have similar idiosyncrasies that make them stand out, at least in IDEVAL.

Detections Buried in False Alarms. Attacks on public services such as HTTP (*apache2, back, crashiis, phf*), SMTP (*mailbomb, sendmail*), or DNS (*ls_domain*) ought not be detectable by source address anomalies in real traffic. In IDEVAL, all of these attacks have source IP addresses that never appear in the training data. While this may be true in real attacks, this fact is hardly useful.

Attributes with No Effect. Some attributes that have higher *r* in IDEVAL, such as TOS, are not significant, as they do not distinguish any IDEVAL attacks.

Incorrectly Simulated Attacks. In IDEVAL, the *smurf* attack can be detected by ICMP checksum errors. *Smurf* is one of the few attacks that is simulated and does not use published exploit code. It is a denial of service network flood, in which the attacker sends ECHO REQUEST packets to a broadcast address with the spoofed source address of the victim. In a real network, this attack should be missed, both because ICMP checksum errors occur normally, and because in a real *smurf* attack, the packets originate from neutral hosts that should not generate checksum errors.

5. Evaluation with Mixed Traffic

The evidence presented in Sections 3 and 4 suggest a problem with the IDEVAL data. However, this data was generated at great expense and could not easily be replaced. We

would prefer a solution that fixes the data as it exists now, rather than require that a new set be created.

We believe it is impractical to synthesize Internet traffic accurately due to its vast complexity. However, we believe that attacks can be – and for the most part were – simulated correctly. Thus we propose to use real traffic (with all the usual implications about privacy and security) as background and training data, but to continue to use the labeled, simulated attacks as before.

Our proposal is to add real traffic to the IDEVAL data to make it appear as if it were being sent and received during the simulation. We believe it is not necessary to remove the simulated background traffic because the combination should be similar (in the statistical sense of Section 4) to the real traffic alone. To see this, let A_S be the set of values of attribute A seen in simulation up to the present time, and let A_R be the corresponding set of values seen in real traffic. Then the set of values A_M seen in merged traffic would be at all times:

$$A_M = A_S \cup A_R \; . \tag{1}$$

Note that the *r* statistic for attribute A_S, which we denote r_S is simply $|A_S|$. Likewise, we define $r_R = |A_R|$ and $r_M = |A_M|$. Therefore, we have at all times:

$$\max(r_S, r_R) \le r_M \le r_S + r_R \; . \tag{2}$$

In cases where we suspect r is an artifact, we have $r_S << r_R$, and therefore $r_M \approx r_R$, so removing the simulated traffic would have little effect. Furthermore, because this is true at all times, r_M and r_R would have similar growth rates.

A problem can occur when A_R is too small or empty, i.e. there is little or no real traffic of types where A is defined to mix with the simulation. In this case, $r_M \approx r_S$, and the artifact, if there is one, would not be removed. One such example is the destination address of incoming traffic, where there are $r_S = 16$ simulated hosts and $r_R = 1$ real host. We are unable to test whether the destination address is an artifact in the simulation (although we have no reason to believe that it would be). Other untestable attributes are those of FTP and telnet payloads, because there is little FTP and no telnet traffic in our real data.

We wish to evaluate network anomaly detection systems on mixed data. Our approach is as follows. First, we analyze the system to determine which attributes are monitored. Then we test the simulated and real data to determine which attributes are present in the simulation, but absent or rare in the real data. Then we either modify the system to ignore these attributes (e.g. remove rules for FTP and telnet), or we modify the real data to remove the discrepancy (e.g. modify destination IP addresses in the real traffic). We illustrate the process with two systems, SPADE and LERAD, then present results for three of our other systems.

5.1. Data Preparation

For our mixed traffic, we use the same large data set as described in Section 4. We have 579 hours of traffic, which is more than enough to mix into the 146 hours of traffic in inside sniffer week 3 plus the 198 hours in weeks 4 and 5. We mix the traffic in a 1:1 ratio, i.e. one hour of simulated traffic is mixed with one hour of real traffic. Other ratios would be possible by stretching or compressing the real traffic, but we do not do this because the traffic rates are similar.

We mix the traffic to make it appear as if all of the collected data occurs during the simulation. We do this by adjusting the time stamp of the first real packet to match the time stamp of the first simulated packet, then maintain the relative times of the other real packets, excluding gaps in the two collections. This is illustrated in Figure 1. Time reads from left to right.

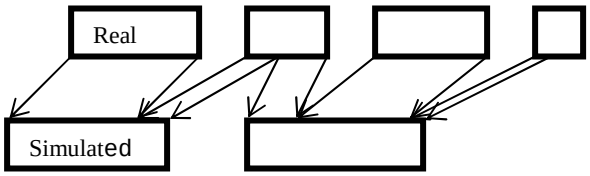

Figure 1. Mapping real time into simulation time when there are gaps in collection in both data sets

To reduce the traffic load on the test anomaly detection systems, we filter both the simulated and real network traffic using the TF prefiltering stage of NETAD [11, 32]. This filter passes only the traffic that would most likely be of interest to an intrusion detection system: the initial packets of inbound client traffic. Specifically, the following is removed:

- All non-IP packets.
- All outbound packets. A packet is inbound if the destination address is 172.16.x.x or 192.168.x.x (simulated eyrie.af.mil) or exactly the IP address of the real server (163.118.135.1).
- UDP packets to high numbered ports (over 1023), which are normally server responses back to clients.
- TCP ACK, SYN-ACK, FIN-ACK, and PSH-ACK packets unless within the first 100 payload bytes of a SYN packet (i.e. only the first 100 bytes of an inbound client request are passed, and none of the server's response).
- Any packet where more than 16 have been passed to the same IP address/port/protocol (TCP/UDP/ICMP) combination in the last 60 seconds. A 4K hash table is used, so there are a small number of drops due to collisions.

After filtering, there are 1,101,653 IDEVAL packets from inside weeks 3-5 (about 3% of original, 0.9 packets per second), and 1,663,608 real packets (1.6% of original, 0.8 packets per second).

The real traffic consists of 50 traces, divided into 10 weeks. We mix these into weeks 3 (training), 4, and 5 (test) of the inside sniffer data to prepare three mixed data sets, which we label A, B, and C as shown in Table 10. We denote the unmixed data (after filtering) as set S.

Table 2. Mixed data sets used for evaluation. All data is filtered

Set	**Training data**	**Test data**
S	IDEVAL inside week 3	IDEVAL inside weeks 4-5
A	S + real weeks 1-3	S + real weeks 4-7
B	S + real weeks 4-6	S + real weeks 7-10
C	S + real weeks 7-9	S + real weeks 1-4

The results for SAD in Section 3 were obtained with sets S and C. However sets A and B give results similar to C. In this case, filtering has little effect because most inbound TCP SYN are passed through.

5.2. Evaluation Criteria

We evaluate each IDS on the IDEVAL data with and without real traffic mixed in. This serves two purposes. First, we wish to know if we successfully removed the artifacts. Second, we wish to predict how these systems would work on real traffic. Although the rate of attacks in the IDEVAL data is artificially high (except possibly for probes), we can still use these results to estimate the probability of detecting an attack given any false alarm rate (e.g. 10 per day), on the type of traffic that we add to the data.

To test whether artifacts are removed, we look at each attack and the attributes that lead to its detection with and without mixing. If, based on the attack's description, the detection is suspect, then we would expect it to be missed when real traffic is added. For example, we would expect that HTTP or SMTP attacks detected by source address in simulated traffic would be missed in mixed traffic. However, if the feature is genuine, for example, *neptune's* forged source address, then the attack would still be detected, although it could still be missed due to a higher false alarm rate. In general, we will use the following somewhat subjective guidelines to determine whether a detection is legitimate.

- Source address is legitimate for denial of service (DOS) attacks that spoof it, or if the attack is on an authenticated service (e.g. telnet, auth, SSH, POP3, IMAP, SNMP, syslog, etc), and the system makes such distinctions. FTP is anonymous in the LL data, so we consider it public.
- Destination address is legitimate for probes that scan addresses, e.g. *ipsweep*.
- Destination port is legitimate for probes that scan or access unused ports, e.g. *portsweep*, *mscan*, *satan*. It is debatable whether it is legitimate for attacks on a single port, but we will allow them.
- TCP state anomalies (flags, duration) are legitimate for DOS attacks that disrupt traffic (*arppoison*, *tcpreset*), or crash the target (*ntfsdos*, *dosnuke*).
- IP fragmentation is legitimate in attacks that generate fragments (*teardrop*, *pod*).

- Packet header anomalies other than addresses and ports are legitimate if a probe or DOS attack requires raw socket programming, where the attacker must put arbitrary values in these fields.
- Application payload anomalies are legitimate in attacks on servers (usually R2L (remote to local) attacks, but may be probes or DOS).
- TCP stream length is legitimate for buffer overflows.
- No feature should legitimately detect a U2R (user to root) or Data attack (security policy violation).

We evaluate each IDS using EVAL [32], our implementation of the 1999 IDEVAL detection criteria. We consider all 177 attacks for which evidence is visible in the inside sniffer traffic to be in-spec. Although U2R and Data attacks would not normally be in-spec, we will consider them to test whether they are detected in real traffic.

In order to reduce alarm floods, we also consolidate duplicate alarms (using AFIL [32]) that identify the same target within a 60 second window prior to evaluation. After consolidation, the group of alarms is replaced with the single alarm with the highest anomaly score.

6. Experimental Procedure and Results

In this section, we evaluate five network anomaly detection systems on mixed traffic. When necessary, we modify the systems or the input data so that the IDS cannot distinguish real and simulated traffic and model them separately. We compare the results using simulated and mixed traffic, and evaluate the legitimacy of the detections for each set. We describe the procedure and evaluation in detail for two representative systems, SPADE and LERAD, and summarize results for three others.

6.1. SPADE

SPADE [5] models various joint probabilities of address/port combinations in inbound TCP SYN packets based on their observed frequency. It assigns an anomaly score of $1/p$, where p is the probability of the current values, calculated by n_v/n, where the current combination of values was observed n_v times out of n total packets (including the current packet to avoid division by 0). There is no specific training period. All packets are added to the training model after evaluation.

There are four user selectable probability modes, as shown in Table 3. The default mode is P(DA, DP) (destination address and port). However, DA is untestable with our data (since there is only one new destination address), and we did not find any evidence that DP might be unrealistic. However, in mode P(DA, DP, SA) (destination address and port, and source address), the addition of real traffic should mask detections if SA is unrealistic, as we suspect. Two other modes include the source port (SP), which does not normally contain useful information.

Because DA carries information that can distinguish real and simulated traffic, we modify the real destination address by replacing it randomly with one of the four IDEVAL victim hosts. This makes it appear to SPADE as if there were four copies of each real server on the four victims, each receiving one fourth of the real inbound traffic. Simulated addresses were not modified.

We tested SPADE version v092200.1, which is built into SNORT 1.7 Win32 [36]. We used sets S, A, B, and C. All SPADE options were set to their default values, and all SNORT rules other than SPADE were turned off. SPADE does not have separate training and test modes, so we ran it on weeks 3 through 5 continuously, discarding all alarms in week 3. SPADE uses an adaptive threshold with various parameters to control alarm reporting. However we used the raw score reported by SPADE instead. The default threshold allows thousands of false alarms so we do not believe that any were lost.

Results are shown in Table 3 for each of the four probability modes. We used a threshold of 200 false alarms rather than 100 because the numbers are low. SPADE detects about half as many attacks at 100 false alarms.

Table 3. Attacks detected by SPADE at 200 false alarms according to EVAL on filtered inside sniffer weeks 3-5 (S) and when mixed with real traffic (A, B, C) in each probability mode

SPADE detections at 200 false alarms	**S**	**A, B, C**
0: P(SA, SP, DA)P(SA, SP, DP)/P(SA, SP)	6	6, 6, 7
1: P(DA, DP, SA, SP)	1	0, 0, 0
2: P(DA, DP, SA)	6	2, 1, 1
3: P(DA, DP) (default)	8	9, 8, 7

Probability modes 0 and 1 include the source port (SP), which is normally picked randomly by the client and would not be expected to yield useful information. The six attacks detected in mode 0 on set S are *insidesniffer*, *syslogd*, *mscan*, *tcpreset*, *arppoison*, and *smurf*. All but *mscan* are probably coincidental because the others generate no TCP SYN packets. However, all but *syslogd* target multiple hosts, increasing the likelihood of a coincidental alarm for one of the targets.

However modes 2 and 3 show the effects of mixing clearly. We have previously identified source address (SA) as an artifact. We now find that adding real data removes most of the detections from mode 2, which uses SA, but not from mode 3, which does not. On S, mode 2 detects *guest*, *syslogd*, *insidesniffer*, *perl*, *mscan*, and *crashiis*. By our previously mentioned criteria, *guest* (telnet password guessing) could legitimately be detected by SA, and *mscan* (a probe for multiple vulnerabilities on a range of hosts) by destination address or port (DA or DP). We do not count *syslogd* or *insidesniffer* (no TCP traffic), *perl* (a U2R attack), or *crashiis* (an HTTP attack).

SPADE detects only *mscan* on all three mixed sets. On A, it also detects *portsweep*, which also can legitimately be detected by DP. Thus, our results are consistent with the claim that SA (but not DP) is an artifact and that the artifact is removed in mixed traffic. When real traffic is added, the fraction of legitimate detections goes from 2/6 to 1/1 (or 2/2).

6.2. LERAD

LERAD [13] models inbound client TCP streams. It learns conditional rules of the form "if $A_1 = v_1$ and $A_k = v_k$ and ... then $A \in V = \{v_{k+1}, ... v_{k+r}\}$, where the A_i are nominal attributes and the v_i are values. If LERAD observes a test stream which satisfies a rule antecedent but the value is not in V, then it assigns a score of tn/r where t is the time since the last anomaly for that rule, n is the number of training streams satisfying the antecedent, and $r = |V|$, the size of the set of values observed at least once in training. The attributes are the individual bytes of the source and destination address, the source and destination ports, stream length and duration (quantized log base 2), the TCP flags of the first and last two packets (as 3 attributes), and the first 8 words (separated by white space) in the application payload. LERAD uses a randomized learning algorithm to generate candidate rules with high n/r. It then discards rules likely to generate false alarms by using the last 10% of the training data as a validation set, discarding any rule that generates an alarm (known to be false) during this time. A typical run results in 60-80 rules after discarding 10-20% of them during validation.

To make sure that the rules generated by LERAD do not distinguish between the simulated and real data, we modified LERAD to weight rules by the fraction of real traffic (identified by destination address) satisfying the antecedent in training. However, we find in practice that this weighting has very little effect. Almost all of the rules are satisfied by a significant fraction of real traffic, and the effect is to decrease the number of detections by less than 3%.

We also modified the TCP stream reassembly algorithm to handle truncated and filtered traffic. The IDEVAL data contains complete packets, allowing streams to be reassembled completely. However, truncated TCP packets would leave gaps. Thus, we truncate the stream after the first 134 bytes of the first payload packet to match the maximum payload size of the filtered traffic. Our filter removes closing TCP flags. Therefore we also modify the TCP flag attributes to be the flags of the last three packets up through the payload, instead of the first and last two packets in the completely reassembled stream. The modified reassembly is also applied to the simulated traffic.

Table 4. Legitimate and total number of attacks detected by LERAD at 100 false alarms on sets S and C

Attribute	**Legitimate/Detected in S**	**Legitimate/Detected in C**
Source address	8/26: dict, guesstelnet, guest, sshprocesstable, sshtrojan / casesen, crashiis, fdformat, ffbconfig, guessftp, netbus, netcat_setup, perl, ps, sechole, sqlattack, xterm, warezclient, warezmaster	0/0
Destination address	1/6: mscan / ncftp, guesstelnet	1/6: mscan / ncftp, guesstelnet
Destination port	14/14: ftpwrite, guesspop, imap, ls_domain, satan, named, neptune, netcat, netcat_breakin	11/11: ftpwrite, ls_domain, satan, named, netcat, netcat_breakin,

Payload	22/29: apache2, back, crashiis, imap, mailbomb, ntinfoscan, phf, satan, sendmail / guesstelnet, portsweep, yaga	8/8: back, imap, ntinfoscan, phf, satan, sendmail
Duration	0/1: insidesniffer	0/0
Length	0/2: netbus, ppmacro	1/1: sendmail
TCP flags	4/9: dosnuke / back, loadmodule, sendmail	4/4: dosnuke
Total	**49/87 (56%)**	**25/30 (83%)**

In 5 runs on set S, EVAL detects 87, 88, 80, 85, and 91 detections. (The loss of detections, compared to about 114 on unfiltered traffic, is due mostly to the loss of TCP flags and some of the payload). On one run each on sets A, B, and C, the modified LERAD detects 29, 30, and 30 in-spec attacks. A typical run on these data sets now results in 30-40 rules after removing 50% of the rules in the validation phase.

In Table 4, we list attacks detected in sets S and C categorized by the attribute that contributes the most to the anomaly score. In each category we list the legitimate detections first, separated from the non-legitimate detections by a slash. By our criteria in Section 5.2, 56% of the attacks detected in S are legitimate, compared to 83% in set C. All of the non-legitimate detections in C are due to destination address anomalies, which we cannot remove because the real traffic introduces only one new address. Sets A and B give results similar to C.

6.3. Results Summary

We tested SPADE and LERAD on mixed data, and found by an analysis of the detected attacks that suspected simulation artifacts were removed. These were primarily source address anomalies, but also include some application payload and TCP flag anomalies in LERAD. Destination address anomalies could not be removed in either system.

We also reached similar conclusions in tests on three of our other systems, PHAD, ALAD and NETAD. We modified all programs so that no rule depends exclusively on simulated data. We present only a summary of the results here. Details are given in [37].

PHAD models Ethernet, IP, TCP, UDP, and ICMP packet header fields without regard to direction (inbound or outbound). No modification was needed because all of these packet types occur in the real traffic. In the results presented in Table 5, most of the non-legitimate detections are due to destination address. PHAD does not include the TTL field, because it is a known artifact, but in another experiment when we added it back in, we found that mixing real traffic removed the artifact.

ALAD models TCP streams like LERAD, but uses fixed rather than learned rules. We modified these rules to remove dependencies on the destination address, which would distinguish the real traffic. We also removed rules for application payloads other than HTTP, SMTP, and SSH. We used LERAD's modified TCP stream reassembly algorithm to handle truncated and filtered packets. The result of injecting real traffic was to increase the fraction of legitimate detections, mostly by removing detections by source address.

NETAD models packets, like PHAD, but for several types such as inbound TCP, inbound TCP SYN, HTTP, SMTP, telnet, and FTP. We removed the telnet and FTP rules.

Again, the fraction of legitimate detections was increased, mostly by removing source address and TCP window size anomalies, which were the dominant means of detecting attacks in IDEVAL.

The results are summarized in Table 5. The original number of detections is the number reported in the literature at 100 false alarms when trained on inside week 3 and tested on weeks 4-5 before the data is filtered or the algorithm is modified. For sets S and C we show the number of legitimate and total detections, and the percentage legitimate. Set C generally resulted in a number of detections between those of sets A and B, and is therefore the most representative of the mixed results. In every case, the fraction of legitimate detections increases when mixed data is used.

Table 5. Legitimate and total detections at 100 false alarms on sets S and C. The original results are the published results based on the unmodified algorithm on unfiltered data (inside weeks 3-5).

System	Original	Legit/S (pct)	Legit/C (pct)
SPADE, mode 2		2/6 (33%)	1/1 (100%)
PHAD	54	31/51 (61%)	19/23 (83%)
ALAD	59	16/47 (34%)	10/12 (83%)
LERAD (avg.)	114	49/87 (56%)	25/30 (83%)
NETAD	132	61/128 (48%)	27/41 (67%)

7. Concluding Remarks

We analyzed the attack-free portions of the 1999 IDEVAL inside sniffer traffic by comparing it with one source of real traffic from the point of view of attributes that are important to anomaly detection. We discovered that many attributes that have a small, fixed range in simulation, but a large and growing range in real traffic, in particular, remote client addresses, TTL, TCP options and TCP window size. The simulated traffic also lacks "crud", such as IP fragments, garbage data in unused TCP fields, bad checksums, and malformed application commands and arguments.

While these differences may be of little consequence to a signature detection system, they are likely to be important to an anomaly detection system, which is designed to be sensitive to unusual events based on a learned model of normalcy. The IDEVAL background underestimates the frequency of novel or unusual events, which would otherwise generate false alarms in many systems. It also allows the systems to detect attacks based on idiosyncrasies that would normally be masked.

We propose solving these problems by adding real traffic to the simulation. This requires careful analysis to ensure that the system cannot distinguish the real traffic from the simulation, and that it does not monitor traffic types for which no real data is available. We did this for five systems and showed that many attacks that appear to be detected by suspected simulation artifacts are no longer detected when real traffic is added.

Although we tested only the 1999 inside sniffer traffic, we believe the problem exists in the outside traffic and the 1998 traffic because they were generated by the same methodology. However our analysis does not apply to the host based data (BSM, audit logs, etc.). This data set was generated by real hardware and software, rather than simulated, so it may not be affected. Also, our analysis assumes that the real traffic is attack-free, but we know that this is not the case. Finding and removing (or labeling) all of the hostile traffic would be difficult. The effect of the presence of these attacks could mask some of the IDEVAL attacks and lower the number of detections we reported. That is, the number of detections reported in this study could be higher, to our advantage, if the real traffic used for training does not have any attacks.

We believe that injecting real traffic into the IDEVAL data is useful because it eliminates the need to simulate (and label) a large number of attacks. Though evaluations using real traffic are still not repeatable because privacy and security concerns usually prohibit the release of this data off-line, our approach allows partial reproducibility based on a publicly available benchmark.

Our analysis is based on a single source of real network traffic. Obviously, every environment is different, so we cannot make general conclusions over different environments. We plan to confirm our results using other sources of real traffic. We have investigated some publicly available datasets, e.g., Internet Traffic Archive [26], however, they are not entirely suitable for our analyses mainly due to the absence of application payload. Also, we also plan to study the effects of using different proportions of simulated and real traffic, that is, at least how much real traffic we need. As hardware gets faster, software gets more complex, and new protocols are introduced, real traffic will look less and less like the IDEVAL data and a new benchmark would be needed. Also, it is more common now for the application payload to be encrypted and hence anomaly detection that involves application payload need to be closer to the application, and away from the network, after the payload has been decrypted.

Acknowledgments

This research is partially supported by DARPA (F30602-00-1-0603).

References

1 R. Lippmann, et al., "The 1999 DARPA Off-Line Intrusion Detection Evaluation", Computer Networks 34(4) 579-595, 2000. Data is available at http://www.ll.mit.edu/IST/ideval/

2 Lippmann, R.P. and J. Haines, Analysis and Results of the 1999 DARPA Off-Line Intrusion Detection Evaluation, in Recent Advances in Intrusion Detection, Third International Workshop, RAID 2000 Toulouse, France, October 2000 Proceedings, H. Debar, L. Me, and S.F. Wu, Editors. 2000, Springer Verlag. p. 162-182.

3 J. W. Haines, R.P. Lippmann, D.J. Fried, M.A. Zissman, E. Tran, and S.B. Boswell, 1999 DARPA Intrusion Detection Evaluation: Design and Procedures. MIT Lincoln Laboratory: Lexington, MA, 2001.
4 J. McHugh, "Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory", Proc. ACM TISSEC 3(4) 262-294, 2000.
5 J. Hoagland, SPADE, Silicon Defense, http://www.silicondefense.com/software/spice/
6 D. Barbara, Wu, S. Jajodia, "Detecting Novel Network Attacks using Bayes Estimators", SIAM Intl. Data Mining Conference, 2001.
7 E. Eskin, "Anomaly Detection over Noisy Data using Learned Probability Distributions", Intl. Conf. Machine Learning, 2000.
8 E. Eskin, A. Arnold, M, Prerau, L. Portnoy & S. Stolfo. "A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data", In D. Barbara and S. Jajodia (editors), *Applications of Data Mining in Computer Security*, Kluwer, 2002.
9 A. K. Ghosh, A. Schwartzbard, "A Study in Using Neural Networks for Anomaly and Misuse Detection", Proc. 8'th USENIX Security Symposium, Aug. 26-29 1999, Washington DC.
10 Y. Liao and V. R. Vemuri, "Use of Text Categorization Techniques for Intrusion Detection", Proc. 11th USENIX Security Symposium, 51-59, 2002.
11 M. Mahoney, "Network Traffic Anomaly Detection Based on Packet Bytes", Proc. ACM-SAC, Melbourne FL, 2003.
12 M. Mahoney, P. K. Chan, "PHAD: Packet Header Anomaly Detection for Identifying Hostile Network Traffic", Florida Tech. technical report 2001-04, http://cs.fit.edu/~tr/
13 M. Mahoney, P. K. Chan, "Learning Models of Network Traffic for Detecting Novel Attacks", Florida Tech. technical report 2002-08, http://cs.fit.edu/~tr/
14 M. Mahoney, P. K. Chan, "Learning Nonstationary Models of Normal Network Traffic for Detecting Novel Attacks ", Edmonton, Alberta: Proc. SIGKDD, 2002, 376-385.
15 P. G. Neumann, P. A. Porras, "Experience with EMERALD to DATE", Proc. 1st USENIX Workshop on Intrusion Detection and Network Monitoring, Santa Clara, CA, 1999, pp. 73-80
16 A. Schwartzbard and A.K. Ghosh, "A Study in the Feasibility of Performing Host-based Anomaly Detection on Windows NT", Proc. 2nd Recent Advances in Intrusion Detection (RAID 1999) Workshop, West Lafayette, IN, September 7-9, 1999.
17 R. Sekar, A. Gupta, J. Frullo, T. Shanbhag, S. Zhou, A. Tiwari and H. Yang, "Specification Based Anomaly Detection: A New Approach for Detecting Network Intrusions", ACM CCS, 2002.
18 R. Sekar and P. Uppuluri, "Synthesizing Fast Intrusion Prevention/Detection Systems from High-Level Specifications", Proc. 8th USENIX Security Symposium, Washington DC, Aug. 1999,
19 M. Tyson, P. Berry, N. Williams, D. Moran, D. Blei, "DERBI: Diagnosis, Explanation and Recovery from computer Break-Ins", http://www.ai.sri.com/~derbi/, April. 2000.
20 A. Valdes, K. Skinner, "Adaptive, Model-based Monitoring for Cyber Attack Detection", RAID 2000, LNCS 1907, p80-92, Springer Verlag, 2000.

21 G. Vigna, S.T. Eckmann, and R.A. Kemmerer, "The STAT Tool Suite", Proc. 2000 DARPA Information Survivability Conference and Exposition (DISCEX), IEEE Press, January 2000.
22 G. Vigna and R. Kemmerer, "NetSTAT: A Network-based Intrusion Detection System", Journal of Computer Security, 7(1), IOS Press, 1999.
23 C. Elkan, "Results of the KDD'99 Classifier Learning Contest", http://www.cs.ucsd.edu/users/elkan/clresults.html (1999)
24 L. Portnoy, "Intrusion Detection with Unlabeled Data Using Clustering", Undergraduate Thesis, Columbia University, 2000
25 K. Yamanishi, J. Takeuchi & G. Williams, "On-line Unsupervised Outlier Detection Using Finite Mixtures with Discounting Learning Algorithms", KDD, p.320-324, 2000.
26 V. Paxson, The Internet Traffic Archive, http://ita.ee.lbl.gov/ (2002).
27 S. Forrest, Computer Immune Systems, Data Sets and Software, http://www.cs.unm.edu/~immsec/data-sets.htm (2002).
28 S. Floyd, V. Paxson, "Difficulties in Simulating the Internet", IEEE/ACM Transactions on Networking, 2001.
29 W. E. Leland,M. S. Taqqu, W. Willinger, D. W. Wilson, "On the Self-Similar Nature of Ethernet Traffic", ACM SIGComm '93, San Francisco, 1993.
30 V. Paxson, S. Floyd, "The Failure of Poisson Modeling", IEEE/ACM Transactions on Networking (3) 226-244, 1995.
31 V. Paxson, "Bro: A System for Detecting Network Intruders in Real-Time", Lawrence Berkeley National Laboratory Proceedings, 7'th USENIX Security Symposium, Jan. 26-29, 1998, San Antonio TX,
32 M. Mahoney, Source code for PHAD, ALAD, LERAD, NETAD, SAD, EVAL and AFIL.PL is available at http://cs.fit.edu/~mmahoney/dist/
33 R. Sekar, M. Bendre, D. Dhurjati, P. Bollineni, "A Fast Automaton-based Method for Detecting Anomalous Program Behaviors". Proceedings of the 2001 IEEE Symposium on Security and Privacy.
34 L. A. Adamic, "Zipf, Power-laws, and Pareto - A Ranking Tutorial", http://ginger.hpl.hp.com/shl/papers/ranking/ranking.html (2002)
35 B. A. Huberman, L. A. Adamic, "The Nature of Markets in the World Wide Web", http://ideas.uqam.ca/ideas/data/Papers/scescecf9521.html (1999)
36 M. Roesch, "Snort - Lightweight Intrusion Detection for Networks", Proc. USENIX Lisa '99, Seattle: Nov. 7-12, 1999.
37 M. Mahoney, "A Machine Learning Approach to Detecting Attacks by Identifying Anomalies in Network Traffic", Ph.D. dissertation, Florida Institute of Technology, 2003.