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

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. The DARPA/MIT Lincoln Laboratory off-line intrusion detection evaluation data set is the most widely used public benchmark for testing intrusion detection systems. Our investigation of the 1999 background network traffic suggests the presence of simulation artifacts that would lead to overoptimistic evaluation of network anomaly detection systems. The effect can be mitigated without knowledge of specific artifacts by mixing real traffic into the simulation, although the method requires that both the system and the real traffic be analyzed and possibly modified to ensure that the system does not model the simulated traffic independently of the real traffic.

1. Introduction

One of the most important data sets for testing intrusion detection systems (IDS) is the DARPA/Lincoln Laboratory off-line evaluation data set, or IDEVAL [1-3]. Since the two evaluations in 1998 and 1999, at least 17 papers have been written on intrusion detection systems tested on this benchmark [4-20]. The network traffic from the 1998 data set was also used to develop the 1999 KDD cup machine learning competition [21], which had 25 participants, and which continues to be used to test intrusion detection methods [22-23].

IDEVAL is the most comprehensive test set available today. The 1999 evaluation data allows testing of 244 labeled instances of 58 different attacks on four operating systems (SunOS, Solaris, Linux, and Windows NT). It supports both host based methods, using audit logs, file system dumps, and system call traces, and network based methods using captured traffic. Attack-free data is available for training anomaly detection systems.

Prior to IDEVAL, researchers usually had to test their systems using data collected from a live environment in which they simulated attacks. In order to protect users' privacy, this data would typically not be released, making it impossible for other researchers to independently confirm published results or to evaluate competing intrusion detection systems in any meaningful way. When live data is released, it is often limited in scope to protect user privacy. For example, the Internet Traffic Archive [24] traces are stripped of application payload data. Forrest's *s-tide* data set [25] would only be useful for testing the detection of UNIX based attacks by analyzing system call sequences without arguments.

IDEVAL solved the user privacy problem by using custom software to synthesize typical user behavior and network traffic to make a small, isolated network (four "victim" machines on an Ethernet) appear as if it were part of a large Air Force network with hundreds of hosts, thousands of users, and an Internet gateway. This was a significant undertaking, considering the complexity of such a system. McHugh [26] criticized many aspects of IDEVAL, including questionable collected traffic data, attack taxonomy, distribution of attacks, and evaluation criteria. With respect to the collected traffic data, he criticized the lack of statistical evidence of similarity to typical Air Force network traffic (especially with respect to false alarms), and low traffic rates. His critique is based on the procedure to generate the data set, rather than on an analysis of the data. To our knowledge, our analysis of the IDEVAL background network traffic is the first of its kind. We do not address issues related to the host based data.

We also do not address attack simulation. Most of the attacks used in IDEVAL are derived from published sources such as the Bugtraq mailing list. We believe that these were simulated correctly. Rather, we question the accuracy of the network traffic portion of the background data. If this is the case, then any simulation errors should affect anomaly detection systems, which model normal behavior to detect novel attacks, and not signature detection systems, which model previously known attacks.

In this paper, we analyze the IDEVAL background network traffic and find evidence of simulation artifacts that could result in an overestimation of the performance of some anomaly detection algorithms, including some trivial ones. Fortunately, all is not lost. We propose that the artifacts can be removed by mixing IDEVAL with real network traffic, in effect simulating the labeled IDEVAL attacks in a real environment. The method does not require any specific knowledge of simulation artifacts, but does require that the real data and the IDS be analyzed and possibly modified to prevent the simulated traffic from being modeled independently of the real traffic.

This paper is organized as follows. In Section 2, we describe related work in anomaly detection and previously known artifacts in IDEVAL. In Section 3, we compare traffic in IDEVAL to real traffic and identify attributes that meet two of three preconditions for simulation artifacts. In Section 4, we propose a method of mixing IDEVAL and real traffic to remove those preconditions wherever they exist. In Sections 5 we find that many attacks can be non-legitimately or trivially detected by some of the attributes that met our preconditions, but that these detections are removed when IDEVAL is mixed with real traffic. In Section 6, we conclude.

2. Related Work

An anomaly detection system builds a model of "normal" behavior (e.g. network client behavior) in order to flag deviations as possible novel attacks. We briefly describe several network based systems.

SPADE [27] models IP addresses and ports of inbound TCP SYN packets, the first packet from a client to a local server. Depending on its configuration, it models the joint probability of the destination address (DA) and port number (DP), and possibly the source address (SA) and source port number (SP), e.g. $p = P(\text{DA, DP, SA})$. The probability is estimated by counting all TCP SYN packets from the time it was turned on until the current packet. It assigns an anomaly score of 1/*p*, and outputs an alarm if this score is above a threshold. SPADE can detect port scans if they probe rarely used ports (i.e. *P*(DA, DP) is low). If SA is included, then SPADE can detect novel (possibly unauthorized) users on restricted ports. For example, if DP = 22 (*ssh*), then *P*(DA, DP, SA) will be low whenever SA is not the IP address of a regular user on server DA.

ADAM [4], like SPADE, monitors client TCP addresses and ports, in addition to address subnets, weekday and hour. It uses market basket analysis to find rules among these attributes with high support and confidence. ADAM combines signature and anomaly detection by passing low probability events to a classifier (a decision tree) trained on labeled attacks. It classifies each session as normal, known attack or unknown attack. Unlike SPADE, ADAM has specific training and test phases.

eBayes [5] models aggregate attributes over a short time window, such as number of open connections, ICMP error rate, number of ports accessed, and so on. It maintains a set of conditional probabilities of the form *P*(attribute = value | category) and uses naive Bayesian inference to classify events as normal, known attack or unknown attack. It includes mechanisms to adjust probabilities to reinforce existing categories and to add new categories when an event does not easily fit the existing categories.

Our prior work includes four systems, PHAD [6], ALAD [7], LERAD [8], and NETAD [9]. These systems have distinct training and test phases, and output an alarm only when a test attribute has a value *never* seen in training. The alarm score is not based on training frequency. Rather it is highest when the time since the previous anomaly is large, the training support is large, and the size of the set of allowed values is small. PHAD and NETAD model network packets including the complete headers, allowing them to detect exploits of low level protocols, such as IP fragmentation vulnerabilities. ALAD and LERAD model TCP streams including words in the application payload, allowing them to detect attacks on open servers (e.g. HTTP or SMTP) that address/port analysis would miss. LERAD uses a market basket approach to finding associations between attributes to synthesize rules. The others use fixed rule sets. All of the systems except PHAD model only inbound client traffic.

Our systems perform competitively on IDEVAL by our own evaluation. In [8] we identify five types of anomalies leading to attack detection.

- Address and port anomalies (traditional firewall model), such as scanning closed ports, or unauthorized users attempting to connect to password protected services identified by previously unseen client IP addresses.
- Use of legal but seldom used protocol features, for example, IP fragmentation. These features are more likely to contain vulnerabilities because they are poorly field tested during ordinary usage.
- Idiosyncratic (but legal) differences in protocol implementations, such as using lower case SMTP mail commands when upper case is usual.
- Manipulation of low-level protocols to hide attack signatures in higher level protocols, for example, FIN scanning to prevent TCP sessions from being logged.
- Anomalies in the victim' s output inresponse to a compromise, for example, a root shell prompt emitted by an SMTP server compromised by a buffer overflow.

However, analysis of the attacks that we detect reveals that many do not fit into any of these categories. We believe these are simulation artifacts. In particular:

- Our published results for PHAD exclude the TTL (time to live) field because over half of the attacks it detects would otherwise be due to anomalies in this field. TTL is an 8 bit field which is decremented after each router hop to prevent infinite loops. In IDEVAL, the values 126 and 253 appear only in hostile traffic, whereas in most background traffic the value is 127 or 254. Most of those attacks exploit application protocols and would have no reason to manipulate TTL. This is also not an attempt to conceal the attacks because the values are too large [28]. We believe these anomalies are 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, LERAD, and NETAD detect a large number of attacks (about half) by anomalous source addresses, including attacks on DNS, web and mail servers, where previously unseen addresses should be the norm.
- NETAD detects several attacks that exploit application protocols by anomalies in the TCP window size field.

3. IDEVAL Background Traffic Analysis

All of the suspected simulation artifacts we observed occur in well behaved attributes. We say that an attribute is *well behaved* if the set of observed values is small and if any two samples collected over different, short time intervals result in the same set of observed values. A well behaved attribute modeled in an anomaly detection system ordinarily will not generate false alarms after a short training period. On the other hand, the set of values observed in a poorly behaved attribute would grow over time. Often, the values have a power law distribution: a few values might appear frequently, while a large number of values would occur only once or a few times.

In IDEVAL, the TTL, client IP address and TCP window size attributes are well behaved. However, this alone does not establish the presence of a simulation artifact. Three preconditions must exist:

1. The attribute is well behaved in simulated traffic without attacks.
2. The attribute is not well behaved in real traffic without attacks.
3. The attribute is not well behaved in traffic containing simulated attacks.

The first condition is required because otherwise the attribute would generate false alarms and not be useful for anomaly detection. (Our systems detect this case and either assign lower anomaly scores or discard the attribute). The second condition is necessary or else the simulation is accurate. The third condition is necessary or else there would be no anomaly. We have not established the second condition, that the attribute is not well behaved in real traffic.

In this section we compare the IDEVAL background traffic with one source of real traffic to test the first two preconditions for simulation artifacts. We test whether an attribute is well behaved in IDEVAL but not in real traffic. We analyze the attributes that are most commonly modeled by network anomaly detection systems: addresses, ports, other packet header fields, and words (delimited by white space) in application protocols (e.g. SMTP, HTTP) in attack-free inbound client traffic.

Lacking published sources of real traffic, we collect our own. We caution that our comparison is based on just once source. However it represents samples from over 24,000 sources on the Internet over most of the protocols used by IDEVAL.

3.1. FIT Data Set

We compare the 256 hours of IDEVAL attack-free training data from inside sniffer weeks 1 and 3 with 623 hours of traffic which we denote as FIT. FIT was collected on a university Solaris machine which is the main server for our CS department (cs.fit.edu). It has several faculty user accounts and serves several thousand web pages. We collected 50 traces of 2,000,000 packets each on weekdays (Monday through Friday) over 10 weeks from September through December 2002. Each trace was started at midnight and lasted 10 to 15 hours. Packets were truncated to 200 bytes. FIT differs from IDEVAL in the following respects:

- The Ethernet traffic is switched, so only traffic originating from or addressed to the host (or multicast) is visible.
- Some hosts on the local network use dynamic IP addresses which can change daily.
- FIT 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 FIT. These include Ethernet, ARP, IP, TCP, UDP, ICMP, HTTP, SMTP, POP3, IMAP, *nbname*, *nbdatagram*, DNS, NTP, *auth* and *printer*. The FIT traffic also has many protocols not found in IDEVAL: IGMP, OSPFIGP, PIM, NFS, RMI, *portmap*, and some obscure and undocumented protocols.

Although FIT is protected by a firewall, it is not free of attacks. We examined the traffic manually and with SNORT [29] and we are aware of at least four attacks from 17 sources: a port/security scan from a host inside the firewall similar to *mscan* or *satan* (although we did not identify the tool used), and multiple probes from three HTTP worms: *Code Red II*, *Nimda*, and *Scalper*. In addition, we are aware of a large number of lower risk probes on open ports: an HTTP proxy scan, a DNS version probe, and numerous ICMP echo probes from scanning tools identified by SNORT as *L3-retriever*, *CyberKit*, *Speedera*, *SuperScan*, and *nmap*.

We filtered both IDEVAL and FIT using TF [30], which passes only inbound client IP traffic prior to analysis. TF also truncates TCP streams by passing only packets containing the first 100 payload bytes. Filtering reduces IDEVAL to 3% of its original number of packets, and FIT to 1.6%.

3.2. Findings

We compared filtered attack-free inbound client traffic from IDEVAL inside sniffer traffic from weeks 1 and 3 with FIT. To test whether an attribute is well behaved, we used the size of the set of the allowed values and its rate of growth. If an attribute is well behaved, then the set will be the same size in the first half of the data as in the full set. If the attribute is poorly behaved, then the first half will contain about half as many values.

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 FIT (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 IDEVAL window size field always has one of 7 values (from 512 to 32,120), but we found 523 different values in FIT, covering the full range from 0 to 65,535.

Source Address Predictability. There are only 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. FIT has 24,924 unique addresses, of which 53% are seen only in the second half of the data, suggesting that the number increases at a steady rate with no upper bound. 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 [31-32] in FIT only.

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

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

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

"Crud". The following events were observed FIT (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 FIT 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 but 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 FIT SMTP requests there are 1839 different HELO arguments, of which 69% appear only once, and 1461 different EHLO arguments (58% appearing only once). In addition, 3% of FIT 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 FIT requests, the SMTP command is lower case. Also, 0.1% of the FIT 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 FIT requests, there are 32 versions, of which 36% appear only once.

We are aware that a few unusual values are due to attacks, in particular, the unusual *Connnection* appears only in the *Nimda* worm, and *EXPN* appears in the security scan (*EXPN root*). However, the great majority of these values appear in what we believe is non-hostile traffic. Also, we do not claim that FIT statistics generalize to all sites. Nevertheless, the FIT traffic contains packets from tens of thousands of Internet sources with a plausible distribution. It would be difficult to suggest that the wide differences between FIT and IDEVAL are due to unsolicited inbound FIT traffic being atypical at the protocol levels we examined.

4. Evaluation with Mixed Traffic

We stated that two of the three preconditions for a simulation artifact are that the attribute must be well behaved in simulation, but poorly behaved in real traffic. Note that if we take the union of a well behaved and poorly behaved attribute, that the result is poorly behaved, i.e., the set of values is large and growing. This suggests that if we mix the IDEVAL traffic with real traffic, then simulation artifacts (if there are any) would be removed. This result is independent of whether the third precondition is met (that simulated attacks make the attribute poorly behaved). Also, if either the first or second precondition is not met, then we are not changing the statistics of the attribute (since they are the same in both traffic sources). This allows us to remove simulation artifacts even if we don' t know where they are.

One difficulty with this approach is ensuring that the IDS cannot model the two traffic sources independently. For example, IDEVAL and FIT can be distinguished by the destination IP address, timestamp, the presence of some protocols, and probably many other attributes. If an IDS models the two sources independently, as in *P*(TTL | destination address), then we have two independent attributes, TTL | IDEVAL (which is well behaved and detects attacks), and TTL | FIT (which is poorly behaved and has no effect). We must analyze the IDS for such dependencies and then remove them either by modifying the IDS or the input data. For example, we could modify the IDS to model TTL unconditionally by ignoring the destination address, or we could map FIT addresses to IDEVAL before input to the IDS.

A second difficulty is that some attributes present in IDEVAL may be missing in the real traffic. For example, FIT has no *telnet* traffic, so any *telnet* model will be unaffected by mixing. We did not identify any simulation artifacts in *telnet*, but we cannot assume that none exist. To be sure, we must either modify IDEVAL to remove *telnet* traffic or modify the IDS to ignore it.

A third difficulty is that both traffic sources will generate false alarms, resulting in a higher rate than either source by itself. This is important because an IDS may be evaluated at a fixed false alarm rate.

A fourth difficulty is that mixed traffic has all of the problems of real traffic with regard to privacy, reproducibility, and independent comparison. Essentially we are testing an IDS on live traffic into which the labeled IDEVAL attacks have been injected. Although we are no longer testing on a standard benchmark, we are spared the effort of having to simulate and/or label the hostile traffic.

With these caveats, we propose to add real traffic to the IDEVAL data to make it appear as if it were being sent and received during the simulation. We do this by setting back the packet timestamps of the real traffic to correspond to the collection periods used in IDEVAL. We use this method rather than setting forward the IDEVAL timestamps because it simplifies the evaluation procedure. We do not have to adjust the timestamps of the IDEVAL alarms during evaluation.

Both the IDEVAL and real traffic sources may contain gaps in collection. The mixing procedure is to adjust the timestamps of the real traffic to remove the gaps, resulting in a contiguous trace, then adjust them again to insert gaps where they occur in IDEVAL (e.g. nights and weekends). Then the two traces are interleaved to create a single trace maintaining the chronological order of the new timestamps. The procedure is illustrated in Figure 1. It would also be possible to stretch or compress the real data to match the packet rate or duration of IDEVAL, although in our experiments the rates are already similar so we just discard any leftover real traffic.

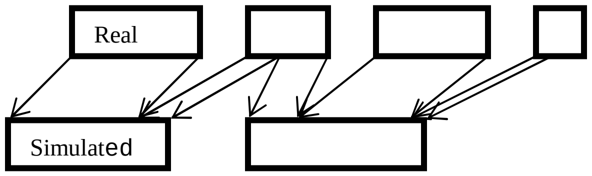

Figure 1. Mapping real time into simulation time when there are gaps in collection in both data sets. Time reads from left to right

5. Experimental Procedure

We wish to test whether the mixing procedure we propose actually removes unexplained detections, i.e. possible simulation artifacts. To do this, we test six anomaly detection systems on IDEVAL and on mixed data sets at fixed false alarm rates. When necessary, we modify the IDS or input data to avoid modeling attributes whose values originate solely from IDEVAL. We establish a criteria for detection legitimacy, essentially asking whether the IDS detects a feature of the attack. If mixing removes artifacts, then the fraction of detections judged legitimate should be higher when tested on mixed traffic.

We test six anomaly detection systems: SAD, SPADE, PHAD, ALAD, LERAD, and NETAD. SAD is a trivially simple anomaly detector that exploits the artifacts described in Section 2 to achieve results competitive with the 1999 blind evaluation. The others are "real" systems as described previously. Space limitations prevent us from describing all of the experiments in detail, so we describe SAD, SPADE, and LERAD, then summarize the results for the others.

5.1. Mixed Data Preparation

We prepare three mixed data sets by mixing the 146 hours of filtered IDEVAL inside sniffer traffic from training week 3 and 197 hours of test weeks 4-5 with different portions of the 623 hours of filtered FIT traffic described in Section 3. We denote the filtered IDEVAL traffic as set S, and the three mixed sets as A, B, and C (Table 1). Each mixed set is constructed by mixing three of the ten weeks of FIT traffic into week 3, and 4 weeks into weeks 4-5. The traces are not compressed or stretched. Rather, leftover FIT traffic from the end of the 3 or 4 week period is discarded.

The traffic rates for IDEVAL and FIT are similar. 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 FIT packets (1.6% of original, 0.8 packets per second). The mixing program, TM, is available at [30].

Table 1. 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 + FIT weeks 1-3	S + FIT weeks 4-7
B	S + FIT weeks 4-6	S + FIT weeks 7-10
C	S + FIT weeks 7-9	S + FIT weeks 1-4

5.2. Evaluation Procedure

We evaluate each IDS by training on week 3 and testing on weeks 4 and 5 for sets S, A, B, and C. We use EVAL [30], our implementation of the 1999 IDEVAL detection criteria. IDEVAL requires that the IDS identify the victim IP address (or at least one if there are multiple targets) and the time of any portion of the attack with 60 seconds leeway, and assign a score for ranking alarms. Only attacks identified as "in-spec" are counted. Alarms identifying out of spec attacks are counted neither as detections nor false alarms.

An attack is in-spec if there is evidence for it in the data examined, and it is in one of the categories appropriate for the technique used. In the inside sniffer traffic, there is evidence for 177 attacks according to the IDEVAL truth labels. The most appropriate categories for network based systems are probes, denial of service (DOS) and remote to local (R2L). However we will consider all 177 attacks as in-spec, including user to root (U2R) and data (secrecy policy violation) attacks because they might be detectable due to artifacts. Normally such attacks would be difficult to detect because they are launched from a shell, which could be hidden from the IDS by encryption, e.g. *ssh*.

We usually set EVAL to report attacks detected at a threshold allowing 100 false alarms (10 per day), consistent with the 1999 evaluation.

In order to reduce alarm floods, we evaluate all systems after consolidating duplicate alarms (using AFIL [30]) 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.

5.3. Criteria for Legitimate Detection

Most of the systems we examine identify the anomaly that is responsible (or most responsible) for each alarm. To answer the question of whether mixed traffic removes simulation artifacts, we must decide whether the anomaly was a "legitimate" feature of the attack, or of the IDEVAL simulation. Because this is a subjective judgment, we establish the following criteria:

- 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 IDEVAL, 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 attacks, but may be probes or DOS).
- TCP stream length is legitimate for buffer overflows.
- No network feature should legitimately detect a U2R or Data attack.

5.4. Experimental Results

5.4.1. SAD

SAD [30] is a trivially Simple Anomaly Detector which we developed for this paper to illustrate how simulation artifacts can be exploited in IDEVAL. SAD examines one byte at a fixed offset (specified as a parameter, e.g. TTL) of inbound TCP SYN packets. During training, SAD records which of the 256 possible values are observed at least once. During testing, it detects an anomaly if the value was never observed in training. If there have been no other anomalies in the last 60 seconds, it outputs an alarm with a score of 1. (The alarms are not ranked).

To find good parameters to SAD, we evaluate it on IDEVAL weeks 1 and 2, which was available in advance to the original 1999 participants for development. Week 1 is attack-free training data, and week 2 contains 43 labeled instances of 18 of the 58 attacks from the test set. We identify good parameters to SAD, defined as any that detect at least 2 attacks with no more than 50 false alarms (10 per day). We find 16 parameters that meet this requirement and submit them for testing.

Before any system can be tested on mixed traffic, we must answer the two questions posed in Section 4: are there any IDEVAL attributes missing from the FIT data, and can SAD model any aspect of the two traffic sources independently? The answer to both questions is no, so no modifications to SAD or the input data are needed. To answer the first question, there are 154,057 IDEVAL and 125,834 FIT inbound TCP SYN packets in set C, regardless of the parameter used, with similar numbers for sets A and B. To answer the second question, the model makes no distinction between the two traffic sources. We focus on set C throughout this paper because in most of the systems we test, the number of detections is intermediate between A and B.

The results for the 16 SAD parameters is shown in Table 2 for weeks 1-2 and for sets S and C. (Results for A and B are similar to C). The results for S are competitive with the top four systems in the 1999 evaluation, which detected 40% to 55% in-spec attacks at 100 false alarms [1]. We make two important observations. First, results from the development set (weeks 1-2) are a good predictor of results in the test set (weeks 3-5). Second, results from IDEVAL are a poor predictor of performance when FIT background traffic is mixed in.

Table 2. Number and percentage of attacks detected (Det) and number of false alarms (FA) by SAD in weeks 1-2 (out of 43 attacks) and in sets S and C (out of 177 attacks)

SAD Byte	**Weeks 1-2**		**Weeks 3-5 (S)**		**Mixed Set C**	
	Det/43	**FA**	**Det/177**	**FA**	**Det/177**	**FA**
IP packet size, 2nd byte	4 (9%)	0	15 (8%)	2	0	1
TTL	25 (58%)	36	24 (14%)	4	5 (3%)	43
Source address, 1st byte	13 (30%)	7	64 (36%)	41	4 (2%)	0
Source address, 2nd byte	13 (30%)	7	67 (38%)	43	0	0
Source address, 3rd byte	16 (37%)	15	79 (45%)	43	0	0
Source address, 4th byte	17 (40%)	14	71 (40%)	16	0	0
Source port, 1st byte	2 (5%)	0	13 (7%)	0	0	0
Dest. port, 1st byte	4 (9%)	24	4 (2%)	0	4 (2%)	1664
Dest. port, 2nd byte	5 (12%)	6	0	0	0	0
TCP header size	4 (9%)	0	15 (8%)	2	0	5
TCP window size, 1st	5 (12%)	1	15 (8%)	2	7 (4%)	112
TCP window size, 2nd	3 (7%)	1	7 (4%)	1	4 (2%)	29
TCP options, 1st-3rd	4 (9%)	4	15 (8%)	2	0	1
TCP options, 4th byte	4 (9%)	4	15 (8%)	2	0	255

We will not discuss the attacks detected in set S in detail except to say that most fail the criteria for legitimacy described in Section 5.3. For example, SAD detects many attacks on public services such as HTTP, SMTP, and DNS by source address anomalies. On set C, SAD detects only *neptune* (excluding coincidental detections) by an anomaly (10 or 11) in the high byte. This attack (a SYN flood) forges the source address with random values, including invalid addresses such as 10.x.x.x. The TTL detections in C (*land, portsweep, queso*) are all for attacks that manipulate low level protocols through raw socket programming, which again passes our legitimacy criteria.

5.4.2. SPADE

SPADE [27] 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 is less well behaved in FIT than in IDEVAL. However, in mode P(DA, DP, SA) (destination address and port, and source address), the addition of FIT 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 FIT and IDEVAL traffic, we modify the FIT traffic so that SPADE cannot make this distinction directly. SPADE monitors only TCP SYN packets, so our input to SPADE consists solely of TCP SYN packets from the two data sets. For each packet from the time-shifted FIT data, we randomly and independently replace the destination IP address with the IP address of one of the four IDEVAL victims (pascal, zeno, marx, or hume). This makes it appear to SPADE as if there are four copies of each FIT server on the four victims, each receiving one fourth of the FIT inbound traffic. IDEVAL addresses were not modified.

We tested SPADE version v092200.1, which is built into SNORT 1.7 Win32 [29]. 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 FIT 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 FIT 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) as legitimate.

SPADE in mode 2 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 FIT traffic is added, the fraction of legitimate detections goes from 2/6 to 1/1 (or 2/2 on set A).

5.4.3. LERAD

LERAD [8] models inbound client TCP streams. It learns conditional rules of the form "if $A_1 = v_1$ and $A_2 = v_2$ and ... and $A_k = v_k$ 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 an anomaly 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 IDEVAL and FIT data, we modified LERAD to weight rules by the fraction of FIT 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 FIT traffic. The effect of weighting is to decrease the number of detections in mixed traffic 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. The TF filter also 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 applied to both the IDEVAL and FIT traffic.

Table 4. Legitimate and total number of attacks detected by LERAD at 100 false alarms on sets S and C. The notation *l/d: l_1, l_2.../n_1, n_2...* means that *l* attack instances were legitimately detected out of *d* total detections. The legitimate attack types are l_1, l_2... and the non-legitimate attack types are n_1, n_2, ...

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 attacks. (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 attack instances 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 FIT traffic introduces only one new address. Sets A and B give results similar to C.

5.5. Results Summary

We tested SAD, SPADE and LERAD on mixed data, and found by an analysis of the detected attacks that suspected simulation artifacts in IDEVAL were removed. These are 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 [33].

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 FIT 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). The notation *l/d* means that *l* detections are legitimate out of *d* total detections.

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%)

6. Concluding Remarks

We analyzed attack-free inbound client background traffic in the DARPA/Lincoln Laboratory IDS evaluation and compared it with Internet traffic from thousands of clients collected in one environment. Based on this comparison, we concluded that many attributes meet two of the three conditions required for simulation artifacts: they are well behaved in simulation but poorly behaved in our sample of real traffic. Some of these attributes meet the third condition by differing systematically in the simulated attacks, as demonstrated by a trivially simple network anomaly detector that competes with the top systems in the 1999 evaluation. The suspected artifacts are the client source IP address, TTL, TCP window size, and TCP options. We also suspect that most of the application protocols contain artifacts, as they satisfy the first two conditions and many attacks exploit these protocols. We have enough data to test only HTTP, SMTP, and *ssh*, and found that all three protocols meet the first two requirements.

We propose that mixing real traffic into the DARPA set will remove artifacts because it turns well behaved attributes into poorly behaved ones, so that the first condition is no longer satisfied whenever the second one is. This procedure does not require that we know where the artifacts are. However, it will fail if the IDS is able to model any attribute in the two traffic sources independently or if any attributes modeled by the IDS are missing in the real traffic. We must analyze and possibly modify both the IDS and the real traffic to ensure that this does not happen. We also lose the advantage of independent testing and comparison due to privacy issues inherent with real traffic. These costs must be weighed against the advantage of using the rich set of labeled attacks available in the DARPA set.

Our work is based on comparison and mixing with real traffic collected from only one environment. However, as we used unsolicited inbound traffic from tens of thousands of sources on the Internet, we believe it is representative with regard to one important characteristic: that poorly behaved attributes with power law distributions are prevalent in real traffic and are underrepresented in the DARPA set. Furthermore, our finding of non-legitimate detections in the DARPA set (e.g. source address anomalies in web server attacks) is independent of our source of real traffic.

We do not make any claims about how the six anomaly detection systems we tested would perform on real traffic. Any results we presented on mixed traffic would apply only to the one real source of traffic that we used, are not reproducible, and would have to be adjusted to account for the two sources of false alarms. The important result is that the fraction of legitimate detection is higher on (our) mixed traffic in all six cases.

We analyzed only the inside sniffer traffic in the 1999 evaluation. It is likely that simulation artifacts exist in the outside sniffer traffic, and also in the 1998 evaluation (from which the 1999 KDD cup data was generated), because the same methodology was used to generate this data. We did not address whether the attacks were simulated correctly, although we have no reason to believe that they were not, as they were taken mostly from published sources. Also, we did not address issues related to any host based data. We indicated a problem (and a solution) with regard to the evaluation of network anomaly detection systems using the DARPA set, but we cannot make any claims with regard to network signature detection or any type of host based detection.

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, Proc. RAID 2000, 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", Lexington MA: MIT Lincoln Laboratory, 2001.

4 D. Barbara, Wu, S. Jajodia, "Detecting Novel Network Attacks using Bayes Estimators", Proc. SIAM Intl. Data Mining Conference, 2001.

5 A. Valdes, K. Skinner, "Adaptive, Model-based Monitoring for Cyber Attack Detection", Proc. RAID 2000, 80-92.

6 M. Mahoney, P. K. Chan, "PHAD: Packet Header Anomaly Detection for Identifying Hostile Network Traffic", Florida Tech. technical report CS-2001-04, http://cs.fit.edu/~tr/

7 M. Mahoney, P. K. Chan, "Learning Nonstationary Models of Normal Network Traffic for Detecting Novel Attacks ", Proc. SIGKDD 2002, 376-385.

8 M. Mahoney, P. K. Chan, "Learning Models of Network Traffic for Detecting Novel Attacks", Florida Tech. technical report CS-2002-08, http://cs.fit.edu/~tr/

9 M. Mahoney, "Network Traffic Anomaly Detection Based on Packet Bytes", Proc. ACM-SAC 2003.

10 E. Eskin, "Anomaly Detection over Noisy Data using Learned Probability Distributions", Proc. Intl. Conf. Machine Learning, 2000.

11 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.

12 A. K. Ghosh, A. Schwartzbard, "A Study in Using Neural Networks for Anomaly and Misuse Detection", Proc. 8' th USENIX Security Symposium 1999.

13 Y. Liao and V. R. Vemuri, "Use of Text Categorization Techniques for Intrusion Detection", Proc. 11th USENIX Security Symposium 2002, 51-59.

14 P. G. Neumann, P. A. Porras, "Experience with EMERALD to DATE", Proc. 1st USENIX Workshop on Intrusion Detection and Network Monitoring 1999, 73-80.

15 A. Schwartzbard and A.K. Ghosh, "A Study in the Feasibility of Performing Host-based Anomaly Detection on Windows NT", Proc. RAID 1999.

16 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", Proc. ACM CCS, 2002.

17 R. Sekar and P. Uppuluri, "Synthesizing Fast Intrusion Prevention/Detection Systems from High-Level Specifications", Proc. 8th USENIX Security Symposium 1999.

18 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/, 2000.

19 G. Vigna, S.T. Eckmann, and R.A. Kemmerer, "The STAT Tool Suite", Proc. 2000 DARPA Information Survivability Conference and Exposition (DISCEX), IEEE Press, 2000.

20 G. Vigna and R. Kemmerer, "NetSTAT: A Network-based Intrusion Detection System", Journal of Computer Security, 7(1), IOS Press, 1999.

21 C. Elkan, "Results of the KDD' 99 Classifier Learning Contest", http://www.cs.ucsd.edu/users/elkan/clresults.html (1999)

22 L. Portnoy, "Intrusion Detection with Unlabeled Data Using Clustering", Undergraduate Thesis, Columbia University, 2000

23 K. Yamanishi, J. Takeuchi & G. Williams, "On-line Unsupervised Outlier Detection Using Finite Mixtures with Discounting Learning Algorithms", Proc. KDD 2000, 320-324.

24 V. Paxson, The Internet Traffic Archive, http://ita.ee.lbl.gov/ (2002).

25 S. Forrest, Computer Immune Systems, Data Sets and Software, http://www.cs.unm.edu/~immsec/data-sets.htm (2002).

26 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) 2000, 262-294.

27 J. Hoagland, SPADE, Silicon Defense, http://www.silicondefense.com/software/spice/ (2000).

28 T. H. Ptacek & T. N. Newsham (1998), Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection, http://www.robertgraham.com/mirror/Ptacek-Newsham-Evasion-98.html

29 M. Roesch, "Snort - Lightweight Intrusion Detection for Networks", Proc. USENIX Lisa 1999.

30 M. Mahoney, Source code for PHAD, ALAD, LERAD, NETAD, SAD, EVAL, TF, TM, and AFIL is available at http://cs.fit.edu/~mmahoney/dist/

31 L. A. Adamic, "Zipf, Power-laws, and Pareto - A Ranking Tutorial", http://ginger.hpl.hp.com/shl/papers/ranking/ranking.html (2002).

32 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).

33 M. Mahoney, "A Machine Learning Approach to Detecting Attacks by Identifying Anomalies in Network Traffic", Ph.D. dissertation, Florida Institute of Technology, 2003.