Submitted to DISCEX-II

Extending the DARPA Off-Line Intrusion Detection Evaluations

Joshua W. Haines, Lee M. Rossey, and Richard P. Lippmann
Lincoln Laboratory, Massachusetts Institute of Techonology
244 Wood Street
Lexington, Massachusetts 02420-9108
{jhaines,lee,rpl}@sst.ll.mit.edu

Abstract

The 1998 and 1999 DARPA off-line intrusion detection evaluations assessed the performance of intrusion detection systems using realistic background traffic and many examples of realistic attacks. This paper discusses three extensions to these evaluations. First, the Lincoln Adaptable Real-time Information Assurance Testbed (LARIAT) has been developed to simplify intrusion detection development and evaluation. LARIAT allows researchers and operational users to rapidly configure and run real-time intrusion detection and correlation tests with robust background traffic and attacks in their laboratories. Second, "Scenario Datasets" have been crafted to provide examples of multiple component attack scenarios instead of the atomic attacks as found in past evaluations. Third, extensive analysis of the 1999 evaluation data and results has provided understanding of many attacks, their manifestations, and the features used to detect them. This analysis will be used to develop models of attacks, intrusion detection systems, and intrusion detection system alerts. Successful models could reduce the need for expensive experimentation, allow proof-of-concept analysis and simulations, and form the foundation of a theory of intrusion detection.

1. Introduction

The 1998 and 1999 DARPA off-line intrusion detection evaluations provided intrusion detection researchers with many examples of attacks and normal traffic. They also provided DARPA program managers and researchers with a thorough assessment of research intrusion detection system performance [1,2,5,13]. Eight research sites submitted host and network-based intrusion detections systems for evaluation. Off-line datasets supported the evaluation, and contained extensive examples of normal and attack traffic run on a realistic testbed network. This data includes network traces, Solaris BSM and Windows NT auditing logs, other log files, and file system information. It allows researchers to easily and quickly perform many identical trial runs with different intrusion detection techniques. More than 160 sites have downloaded the data from these evaluations to test and develop intrusion detection systems [6]. Section 2 of this paper reviews the design, results, and limitations of these evaluations.

Recent advances in intrusion detection technology require that the 1998 and 1999 evaluations be extended to more fully test new detection systems and techniques. In addition, the two completed evaluations can form the basis of new modeling and simulation efforts that can reduce the need for expensive experimentation. Three current efforts are planned and/or in progress to extend the earlier evaluations.

First, the Lincoln Adaptable Real-time Information Assurance Testbed (LARIAT) is a collection of software and tools that can be distributed to developers to provide an easily configurable real-time testbed for intrusion detection development and testing. LARIAT allows developers to simulate a wide range of operating environments by producing realistic background traffic on testbeds in their own labs. Attacks can be scripted and inserted into this traffic at pre-defined times. Researchers can test and train real-time detection techniques and can also run and re-run tests with varied background traffic and attacks without having to wait for a new off-line dataset to be published. LARIAT has been under development for roughly one year and is described in section 3.

Second, two short, focused datasets containing attack scenarios have been created and distributed. In each, a novice adversary attacks a naively defended network. These attacks and realistic background traffic were run on a network testbed that modeled the naively defended network. Network traces from the "Internal" and "DMZ" networks, and Solaris BSM host auditing data contain evidence of all of the phases of these attack scenarios.

This work was sponsored by DARPA under Air Force Contract F19628-95-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Air Force.

Labeling is provided in terms of textual descriptions, individual packets or Solaris BSM records, and alerts based on the IETF standard Intrusion Detection Message Exchange Format (IDMEF) [11]. These datasets are described in section 4.

Third, the 1999 Evaluation supported several major analyses that can be the basis for new modeling and simulation efforts. Initially, accurate labeling of the attacks in the data established evaluation "truth". Then, participants and evaluators analyzed initial evaluation results to agree on the scoring truth and to determine why attacks were missed and what caused false alarms. Also, participants and evaluators analyzed detection rates, false alarm rates, and identification scoring results in detail for presentation to the research community and to DARPA. These extensive efforts provide understanding of many attacks, their manifestations, and features used to detect attacks. They also show the importance of using realistic and detailed adversary models, such as those described in [7,8]. Planned modeling and simulation efforts are based on these evaluation results and are described in section 5.

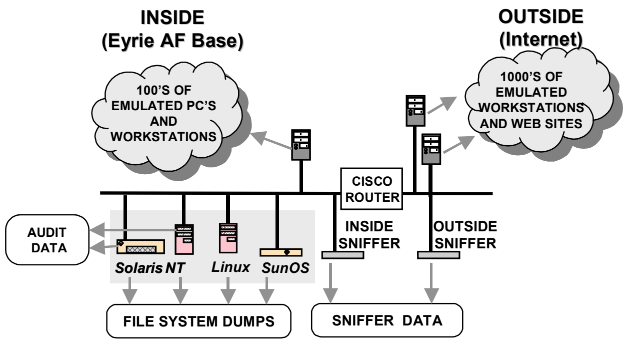


Figure 1 Block diagram of 1999 test bed.

2. Review of the 1999 evaluation

The DARPA 1998 and 1999 Intrusion Detection Evaluations consisted of comprehensive technical evaluations of research intrusion detection systems [1,2,5,13]. These evaluations had two main objectives: to evaluate the performance of DARPA-funded intrusion detection technology; and to support the reseachers developing that technology. Intrusion detection systems were evaluated for detection accuracy by providing both realistic background traffic and attacks to allow measurement of false alarm and attack detection rates. The evaluation assisted research and development by providing extensive examples of realistic background traffic based on observations made at an operational site in 1998. Usage patterns of a wide variety of common services were modeled and these models were the basis for the synthesis of realistic user-sessions using real services and protocols. Synthetic users surfed the web, sent, read, and replied to email, transferred files with FTP, logged into hosts with Telnet and SSH, edited documents, and edited and compiled code. Many examples of a wide range of attacks were also provided. These evaluations were not designed to evaluate complete, deployable intrusion detection systems or commercial systems, but rather to evaluate the accuracy of alternate technical approaches.

Figure 1 shows the isolated test bed network that supported the 1999 evaluation. Scripting techniques that extend the approaches used in [3,4] are used to generate live background traffic that is similar to traffic that flows between the inside of one Air Force base and the Internet. This approach was selected for the evaluation because hosts can be attacked without degrading operational systems and because corpora containing background traffic and attacks can be widely distributed without security or privacy concerns. A wide variety of background traffic is generated in the test bed that appears to have been initiated by hundreds of users, sourced from tens of hosts, and destined to thousands of hosts. The left side of Figure 1 represents the inside of the fictional Eyrie Air Force base and the right side represents the Internet. Automated attacks were launched against four inside victim machines (SunOS, Solaris, Linux, and Windows NT) and the router from both inside and outside hosts. Not shown in this figure are the Windows NT and UNIX workstations from which the attacks were launched. More than 200 instances of 58 different attacks were embedded in three weeks of training data and two weeks of test data.

Machines labeled "sniffer", in Figure 1, capture packets transmitted over the attached network segment using tcpdump [18]. In addition, Windows NT audit events and Solaris BSM audit events are collected from victim hosts. File system listings and dumps of selected files are also collected from each of the victims.

The 1999 evaluation focused on measuring the ability of systems to detect new attacks without first training on instances of these attacks. Many new attacks were developed and examples of only a few of these were provided in training data. Some attacks were launched from within the Air Force Base network, to simulate the dangers posed by insider attacks. Inside sniffer data was provided to researchers to enable detection of these attacks. Stealthy attacks were included to simulate sophisticated attackers who can carefully craft attacks to look like normal traffic [15,16]. Attacks against Windows NT were included to simulate the increased use of such systems in operational settings. Occasional attack and normal traffic sessions were fragmented using techniques similar to those described in [12]. Two types of analyses were performed in addition to attack detection and false alarm analysis. First, participants optionally submitted attack forensic information that could help a security analyst identify the important characteristics of the attack and formulate responses. This identification information included the attack category, the attack name for old attacks (those seen in training data), the ports or protocols used, and the IP addresses used by the attacker. Second, an analysis of misses and high-scoring false alarms was performed for each system to determine why systems miss specific attacks and what causes false alarms.

The evaluation results[1,2] showed that the best overall performance would have been provided by a combined intrusion detection system that used both host- and network-based intrusion detection. Detection accuracy was poor for new (not previously seen), stealthy, and Windows NT attacks. Ten of the 58 attack types were completely missed by all systems. Systems missed attacks because protocols and TCP services were not analyzed at all or to the depth required, because signatures for old attacks did not generalize to new attacks, and because auditing was not available on all hosts. Promising capabilities were demonstrated by host-based systems, by anomaly detection systems, and by a system that performs forensic analysis on file system data.

The Results of the 1999 evaluation should be interpreted within the context of the test bed, background traffic, attacks, and scoring procedures used. The evaluation used a reasonable, but not exhaustive, set of attacks with a limited set of actions performed as part of each attack. It also used a simple network topology, a non-restrictive security policy, a limited number of victim machines, probabilistic low-volume background traffic, simple scoring, and extensive instrumentation to provide inputs to intrusion detection systems. Future evaluation efforts could improve in many of these areas including testing systems in a range of environments with differing background and attack characteristics.

The off-line evaluation format limited the 1998 and 1999 evaluations to passive intrusion detection systems that can operate in an off-line mode. The off-line format is difficult to use with systems that query hosts or other network components, or with systems that respond by changing network or host configurations. Queries can probe the network or request information from an application or operating system in response to some possible attack detection. Although not impossible, it is quite difficult to record all state information that might be required by a query, and thus produce an off-line data stream to take the place of a real-time implementation. It is also difficult to evaluate systems that respond to attacks by changing host or network configurations since the effects of changes on the attack or background traffic are difficult to predict and include in a static, off-line dataset. The DARPA real-time evaluation [22] responded to some of these needs but was time consuming to run since evaluators had to setup and run the evaluation for each intrusion detection system to be evaluated.

Additionally, the 1999 evaluation primarily supported intrusion detection systems that detect atomic attack actions using raw event streams as input. Currently, correlation and fusion systems are being developed that operate at a higher level and use the outputs from low-level intrusion detection systems as input. These systems attempt to recognize related events to find attack scenarios, increase understanding of individual activity, or reduce the number of false alarms. The 1999 evaluation datasets can support some correlation systems that attempt to better identify atomic attack instances or reduce false alarm rates using one or more lower-level detectors. However, correlation systems that recognize multiple component attack scenarios with one or more detection alert streams will need datasets containing realistically written, multiple-component attack scenarios.

Many other lessons were learned from the 1999 evaluation. It is expensive and time consuming to setup and run background traffic and attacks for days and weeks. Attacks fail and machines crash, often for reasons that are difficult to predict and avoid. Verifying that an attack ran successfully, cleaning up after it, and scoring putative detection alerts was much more complex than expected. Better automation of background traffic generation and attack launch coordination can help, along with automated network and host initialization, attack verification and cleanup, and integrated scoring software.

3. LARIAT

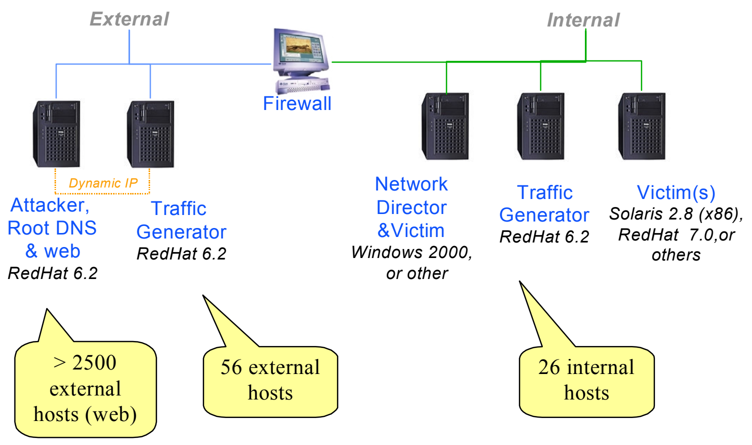
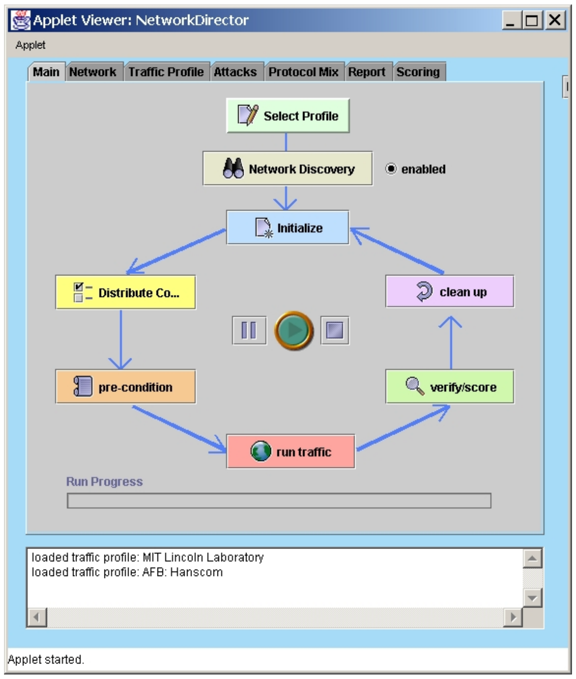


Figure 2 LARIAT network diagram. Victim hosts and Traffic Generators can be added as needed.

The Lincoln Adaptable Real-Time Information Assurance Testbed, LARIAT, overcomes many of the limitations of the 1998 and 1999 off-line evaluations by giving users and developers a framework in which to perform real-time tests of intrusion detection systems and other information assurance tools in their own laboratories. LARIAT provides an easily configured test environment for intrusion detection systems that query hosts and networks or that invoke responses. LARIAT abstracts the underlying complexity of traffic generation, to allow users to quickly and easily perform test runs at a wide range of traffic rates. It also allows users to customize traffic load and content, add or remove hosts, and even integrate LARIAT into an existing testbed network. This section describes how a LARIAT test run proceeds and how LARIAT can be configured.

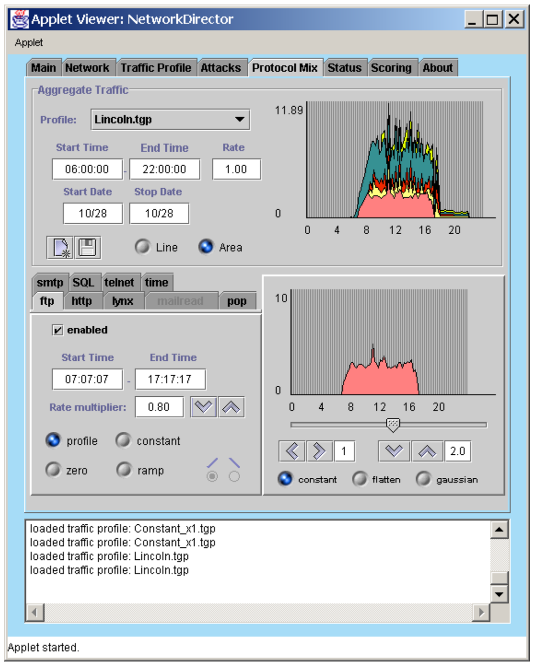
LARIAT provides easily configured versions of the background traffic generation and attack scripting software used in the 1999 Evaluation and a graphical user interface (GUI) for configuration and control. Figure 2 shows the base LARIAT network. Similar to the 1999 evaluation testbed, the left side of the figure simulates the external Internet while the right side simulates the Intranet being protected and against which attacks are launched. The *network director* is a Java applet that can be run from within a web browser of any host on the network. It controls the testbed and allows the user to setup test runs via the GUI. The *traffic generators* run Red Hat Linux and a special version of the Linux kernel that allows one *traffic generator* to look like many hosts on the internal network, and the other to look like many hosts around the Internet [19,22]. Background traffic sessions are sourced from and destined to both virtual hosts and real victim hosts that the user places on the network and configures into the generation software. The *Root DNS & Web* machine serves as the root Domain Name Service (DNS) server for the testbed and mirrors web content from thousands of real websites. It uses the same Linux kernel. Content can be retrieved via an existing Internet connection if desired. Attacks are launched from the *attacker*. The *attacker's* IP address can be selected at attack time from IP addresses used by the traffic generator. The victims shown are Solaris 2.8 (x86), Linux, and Windows 2000 as these are the platforms for which attacks have thus far been incorporated, however any platform could be added as a victim. The *network director* allows the user to add new victim hosts to the background traffic as necessary so that normal use traffic impinges on machines being attacked. Additional *traffic generators*, on additional subnets, with more virtual hosts, can be added to model more complex operating environments.

Each test run consists of a few steps, displayed on the LARIAT GUI main panel in Figure 3. To start a test run, the user selects a profile for the background traffic. A profile defines the operating environment to be simulated on the testbed. These profiles can be modified with respect to the content and distribution of services, attacks, and attack-launch times. Profiles can be saved and loaded by the GUI for later use. Figure 4 shows a screen capture of the GUI panel that allows background traffic modification. The upper part of the panel shows the aggregate traffic to be generated, including the start and end times, a global rate modifier, and profiles of the arrival rates of the user sessions of each traffic type. The traffic profile graph gives the expected number of session arrivals (y-axis) for each 15-minute interval throughout a 24-hour day (x-axis). The lower part of Figure 4 shows how the user specifies the amount of FTP traffic to be generated, with the profile for FTP traffic on a similar graph (plotted by itself). User sessions for each traffic type are similar to those in the 1999 evaluation data. Arrival rate and distribution of sessions of each type can be adjusted to specify aggregate content of background traffic. This allows testing of an intrusion detection system in a range of operating environments or testing of system throughput with high traffic rates. The user can also select and schedule attacks that will be used during a test run. Attacks are scripted; some configurable details, such as the victim host or username involved, can be modified via the GUI.



loaded traffic profile: MIT Lincoln Laboratory
loaded traffic profile: AFB: Hanscom

Figure 3 The LARIAT GUI's main control panel. Here the user can view the status of a test run.



Profile: Lincoln.tgp
Start Time 06:00:00 End Time 22:00:00 Rate 1.00
Start Date 10/28 Stop Date 10/28
[✓] enabled
Start Time 07:07:07 End Time 17:17:17
Rate multiplier: 0.80

loaded traffic profile: Constant_x1.tgp
loaded traffic profile: Constant_x1.tgp
loaded traffic profile: Lincoln.tgp
loaded traffic profile: Lincoln.tgp

Figure 4 The LARIAT GUI profile-editing panel. Here the user can modify the background traffic profile.

"Network Discovery" is a placeholder for future functionality in which LARIAT will be able to "discover" the hosts, users and capabilities of the network into which it is installed, rather than having to be user-configured. For example, basic reachability checks could verify which network services are allowed from and to the configured *traffic generators*.

During the "Initialize Network" stage, the control software initializes the network and hosts by removing artifacts from previous runs (e.g., hung user sessions and processes) and initializing logging mechanisms. Hosts involved in background traffic are also initialized to ensure a common starting point for each run. During the "Distribute Configuration" stage, the *network director* sends the user-specified profiles to each *traffic generator*. Then during the "Pre-Conditions" stage, the *traffic generators* synthesize scripts for each background traffic session. These scripts use a custom extension to the Expect scripting language that was developed for the 1998 and 1999 evaluations. It allows fine-grained control of how long simulated human users wait between entering commands in interactive sessions and how long it takes to type commands using probabilistic inter-character delays. Attack scripts and these background traffic scripts are scheduled to run at the appropriate times by a batch script on each *traffic generator* and the *attacker*. Finally, the test run is started.

During the run the user can view the progress of background traffic and attacks in real-time via the GUI. Upon completion of the run, or a specified interval after each attack, special scripts verify that attacks ran successfully by scanning attacker logs and searching for evidence of the attack on the victim host. After the run, cleanup scripts, specific to each attack, remove evidence of that attack run, resetting any changes made by the attack script. Hosts involved in background traffic are also re-initialized. Hung user sessions are killed, as in the earlier "Initialization" phase, to ensure that test runs start from common system state, even if a test run is terminated before completing.

In future versions, the *network director* will also collect alerts from the intrusion detection system being tested. The *network director* will then perform first-order "scoring", denoting alerts as "hits" or "false alarms" and presenting them to the user via the GUI. This alert classification process will be a guide for further analysis by the user or researcher.

LARIAT provides many ways to configure the background traffic and attack generation for each run or test environment. Beyond adjusting traffic rate, profile, and start/stop times by service, the user can add or remove zones. Zones can correspond to Internet domains or sub-domains and contain real and virtual hosts to be sources and destinations of traffic sessions. Two zones are configured by default. Zones can be supported with additional *traffic generators*. Real and virtual hosts can also be added to existing zones. *Taffic generators* are supplied with information about each zone and the hosts within them. To integrate new hosts, scripts are provided to configure the new hosts with the system state (users, passwords, files, directories) required for the background traffic destined to the host. Alternatively, for some services, it is possible to update the traffic generation software with the existing system state of new hosts. Users, via the *network director* GUI, tell the background traffic generators how much and what types of traffic from that zone are to be destined to other zones. Thus, real victim hosts can be added or removed and background traffic can be configured to impinge on the new hosts, or LARIAT can be integrated into an existing testbed. LARIAT can also archive all background traffic scripts from an entire run to replay them at a later time.

LARIAT currently generates traditional UNIX background traffic sessions (FTP, Telnet, Web-Surfing, SMTP mail traffic, PoP, etc.) and some sessions to and from Windows NT hosts (Telnet, FTP, Mail, Web-surfing), similar to those described in Section 7 of [13]. New services and traffic are being developed to match a wider variety of current operating environments. A set of attacks is provided with the tool, some of which are specific to the victim hardware and software. Typically attacks need to be selected for specific victims and network configurations. Attacks can be scripted in LARIAT's Expect-based scripting language or can be automated by incorporating the attack in its native form (perl, shell script, binary, etc.) and running it via an automatically generated wrapper script. Attacks can also be launched manually, if desired.

4. Scenario datasets

The 1999 Evaluation showed that it is difficult to design and create a single large dataset to satisfy the needs of many intrusion detection researchers. Such a dataset needs to have many examples of a wide range of attacks; the 1998 and 1999 evaluation datasets used this approach. Shorter datasets that focus on one particular type of attack are easier to develop and can be more useful to a researcher. For example, a four-hour test run could focus on a User-to-Root attack and provide many examples of that attack, or could focus on the Nmap [21] network-probing tool and provide examples of all of the command-line options of that tool. These datasets would provide many examples of individual attack components for lower-level intrusion detection research. A short, focused data set could also address the needs of intrusion correlation researchers by focusing on a single attack scenario, and providing all of the steps of the scenario.

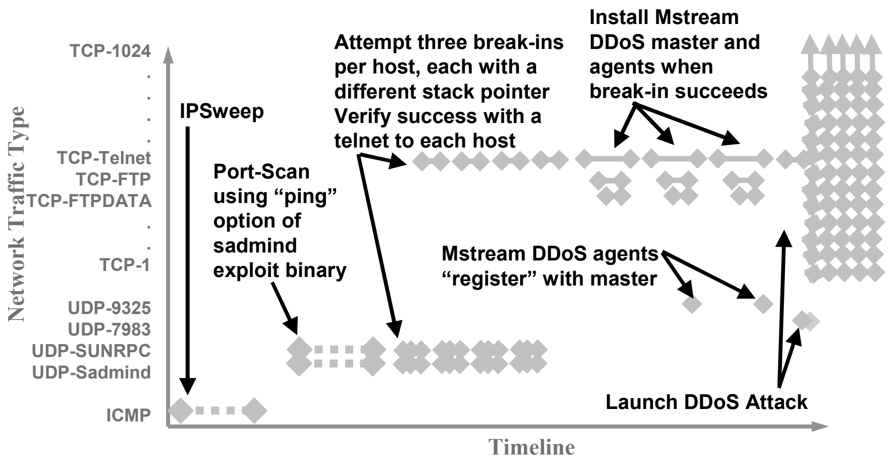


Figure 5 A plot of network traffic generated by LLS_DDOS_1.0 attack scenarios. Gray marks or lines indicate traffic, the X-axis gives the notional time at which it occurred and the Y-axis gives the type of traffic.

Two recently created scenario datasets address the needs of mid-level correlation systems. Each includes several hours of background traffic and a complete attack scenario. Attacks and background traffic were run on the same testbed used in the 1999 evaluation, but with the addition of a commercial-off-the-shelf firewall and de-militarized zone (DMZ) network separating the Internal and Internet networks and a Solaris 2.7 victim host. The adversary, adversary's goal, defender and defender capabilities are modeled by selecting the attack actions and network defenses employed. In both datasets, a novice adversary attacks a naively defended network with the goal of installing the mstream [10] Distributed Denial of Service (DDoS) software and launching a DDoS attack against another military target. The firewall was configured to allow many services, modeling the naive defender, while the adversary carried out fairly blatant attack actions, modeling the novice attacker. Datasets were created with input from Sandia National Laboratories and intrusion detection and correlation researchers.

In the first dataset, LLS_DDOS_1.0, the attack is almost entirely scripted and happens in several phases. Figure 5 summarizes the network traffic generated by the attacker's actions. First, a sweep of every IP address at the fictitious Eyrie Air Force Base discovers hosts. Live hosts are probed for the Solaris Admin Suite (sadmind) by attempting to connect to this RPC managed service. Hosts running this service are attacked using the sadmind exploit, a buffer-overflow yielding remote root-level access [9]. For the attack to succeed, the attacker must specify the correct stack pointer, which can vary from architecture to architecture. The adversary has verified, with independent testing, three stack pointers as possibilities, and the script tries each pointer on all potential victims. The attack attempts to create a root-level user on the victim. Success is verified with a Telnet to the victim. Three hosts are compromised. With this access, the script uses Telnet and FTP to install an mstream DDoS attack client on each victim and the mstream server on one victim. At a later point, the attacker manually logs into the mstream server and launches the DDoS attack.

The second dataset, LLS_DDOS_2.0.2, is stealthier than LLS_DDOS_1.0 and is also almost entirely scripted. It conducts the initial scan using the DNS HINFO query. If an administrator has put operating system information into the HINFO records, these queries can indicate which hosts are Solaris. The attack then directs the sadmind probe at only those hosts that are reported to be Solaris. This version of the scenario initially compromises only one host at the base from the Internet, and then launches all subsequent break-in attempts via a script from that host, rather than performing all actions from the remote host.

Network traces from outside the network gateway and from the internal network are provided. Solaris BSM auditing streams are provided from two of the three primary victim hosts: mill.eyrie.af.mil (Solaris 2.7) and pascal.eyrie.af.mil (Solaris 2.5.1). Table 1 shows these data streams and indicates what attack evidence exists in each. It is described in more detail in section 6. Three types of truth labeling are provided. High-level labeling is a paragraph describing each attack phase. Low-level labeling specifies all packets or audit records from sessions that correspond to the attack and helps researchers find evidence left by these attacks. Mid-level labeling contains IDMEF-style [11] XML alerts for each network session and for each "exec" BSM audit record that was related to the attack. Mid-level labeling shows the alerts that one possible intrusion detection system might produce. This data and labeling is posted on a password protected website for use by researchers [6].

5. Modeling and simulation

1999 Evaluation results and analysis provided understanding of how and where evidence of exploits is manifest in data streams used by intrusion detection systems. Further, the evaluation provided understanding of how intrusion detection systems work in terms of input features and the type of analysis they perform. This section presents a "Feature-Analysis" model for modeling attacks and intrusion detection systems. Attack modeling is discussed in Section 5.1 and intrusion detection system modeling is described in Section 5.2. These models could be a basis for proof-of-concept and probabilistic experiments that could help reduce the need for costly experimentation and foster rapid advances in the areas of intrusion detection and information assurance. Several specific applications for these models are described in Section 5.3.

5.1 Attack models

Feature-Analysis modeling for attacks requires an understanding of the evidence left by an attack. This enables one to delimit the capabilities required by an intrusion detection system to detect that attack. For example, Table 1 shows data streams that must be analyzed to find each action of the attack scenario used in Lincoln Scenario Dataset LLS_DDOS_1.0. Ten attacker actions that summarize the attack scenario are given, each listed on a separate line of the table. Each square of the grid shows a binary analysis of whether that action left evidence in the data stream. For example, detection of the IPsweep using Solaris BSM data might prove difficult,

Where the Exploit is Manifest	Inside TCPdump	Outside TCPdump	Mill Solaris BSM	Pascal Solaris BSM
IPSweep	YES	YES		
Probe for sadmind, Host mill	YES	YES	YES	
Probe for sadmind, Host pascal	YES	YES		YES
Probe sadmind, 8 non-Solaris hosts	YES	YES		
Breakin via sadmindex, host mill	YES	YES	YES	
Breakin via sadmindex, host pascal	YES	YES		YES
Install/Run master/client on mill	YES	YES	YES	
Install/Run client on pascal	YES	YES		YES
Launch DDoS	YES	YES	YES	YES
DDoS	YES	YES	YES	YES

Table 1 Data streams that contain evidence of each attacker action in LLS_DDoS_1.0.

but the Network traces contain sufficient evidence for detection. Subdividing the cells of the table could yield a more realistic analysis, called "Feature-Analysis" modeling in this paper. For example, a TCPdump data stream could be subdivided to indicate what data and understanding must be retrieved from the network data to detect the attack. The breakdown could be by the traffic type (ARP, ICMP, UDP[DNS, Syslog, ...], TCP[telnet, http, ftp, …], etc.) and the type of analysis performed (IP packet header fields, TCP header fields, TCP Session, etc.). For example, to best detect the IPsweep, the intrusion detection system needs to look at one or more ICMP packets within the network data stream.

5.2 Intrusion detection system models

Feature-Analysis modeling of intrusion detection systems requires an understanding of how these systems work. A thorough understanding of the input data streams, features, detection algorithms, and output helps in understanding what evidence must exist for an intrusion detection system to detect an attack and helps define the content of the resulting alert.

Table 2 shows a Feature-Analysis model for Network-IDS-1, a fictitious, network based intrusion detection system. The columns denote the data feature input, and the rows give the level of analysis performed by the intrusion detection system. As shown, "Network-IDS-1" extracts and analyzes ICMP traffic, TCP/IP headers (for common services), TCP/IP session setup (for common services), and the content of HTTP sessions. This detection system should be able to detect attacks where evidence manifests itself in ICMP packets, TCP/IP header fields and handshake, and in HTTP session content.

This model in Figure 2 is certainly not complete or ideal. Service and protocol categories could be subdivided; for example, different types of ICMP packets or DNS queries could be listed. Also, more specific analysis techniques can be listed and more details of the analysis could be provided. For modeling purposes, real-time queries for information can be considered "data streams" and a similar model of those sources of input data can be created. The features extracted and analysis techniques applied can be represented with the added notion of the timing of the query(s) required for detection. The Features and Analyses should represent the actual intrusion detection system under consideration as much as possible. Feature-Analysis models cannot be static; they must be updated as more is learned about attacks and as new intrusion detection techniques are developed. Rows or columns must be added or sub-divided for each new analysis technique or feature to maintain the realism of the model.

Network IDS-1	Ethernet	ARP	ICMP	UDP	DNS	Syslog	TCP/IP	Telnet	HTTP	SSH	SMTP	RPC	32773/u
Header			X				X	X	X	X	X	X	
Multiple Headers			X				X	X	X	X	X	X	
Protocol "Handshake"							X	X	X	X	X	X	
Packets			X										
Multiple Packet			X										
Session Negotiation									X				
Session Content									X				
Session Actions													

Table 2 One possible characterization of the fictitious "Network-IDS-1" showing the features extracted and depth of analysis performed.

The Feature-Analysis technique can be extended to model Solaris-BSM based intrusion detection systems by altering the row/column heading to match the features and analysis techniques of that data source. For example, the columns would list the programs analyzed (inetd, named, httpd, tcsh, etc.) and/or the type of record analyzed (exec, open/close, etc.) The rows would list analysis techniques specific to BSM-based detection (user profiling, program-profiling, etc.)

As described earlier, Feature-Analysis modeling can also apply to attacks. For each attack action and stream of data to be considered, a table similar to Table 2 can be created. The X-axis could show the ways in which evidence of that attack manifests itself and the Y-axis could show the types of analysis that might detect an attack. Each cell of Table 1 could be replaced with a complete breakdown similar to that of Table 2.

5.3 Uses of models

Feature-Analysis models of computer attacks and intrusion detection systems could be used as a basis for modeling and simulation efforts that could promote rapid advances in intrusion detection. The following four research efforts based on modeling and simulation could help address current research needs and advance intrusion detection and correlation research:

1. **Predict attack detection coverage and alert content**. Models of attacks and intrusion detection systems could be analyzed to see where they overlap to determine which attacks can be detected. This comparison is a basis for proof-of-concept and probabilistic analyses to assess attack coverage of intrusion detection systems. For example, one or more attacks could be compared to the model of Network-IDS-1 shown in Table 2. The model would give a "first-order" idea of which attacks Network-IDS-1 can be expected to detect. A similar rule-based characterization and evaluation of detection systems is proposed by [20]. An alert content model, based on expert knowledge of the detection system and content of alerts resulting from different attacks, could be added to postulate alert content in response to various modeled attacks. The costly process of actually developing the attacks, creating background traffic, and developing a deployable detection system could be used more sparingly to verify results of the model-based analysis.
2. **Plan future evaluations.** Feature-Analysis modeling can help understand what background traffic and attacks are necessary to fully stress a particular intrusion detection system. Background traffic is a necessary part of an evaluation to prevent trivial detection of attacks (i.e., everything that happens is an attack) and to allow measurement of false alarm rates in different operating environments. For background traffic to suit these purposes, it must both be realistic and it must match the detection algorithm that is being tested. If a network-based system does not analyze ICMP traffic, then there is no need to model realistic ICMP traffic in the background environment. However, if a detection system keeps statistics on connections to port 80 of a server, then web traffic should be included with a realistic arrival rate that matches the real environment. Similarily, there is no need to evaluate performance of an intrusion detection systems with an attack when none of the evidence of the attack is analyzed by the system.

3. **Synthesize realistic alert streams.** Models of real alert content could be used to synthesize realistic alert streams for use by researchers working on correlation and higher-level detection. These synthetic alert streams would have the "look and feel" of streams of operational alerts from real intrusion detection systems installed in operational environments, but could be more widely distributed since they would not contain operationally sensitive information. To create a synthetic alert stream like this, an intrusion detection system could be run on operational data in a secure facility. Alerts could be collected and statistically modeled with respect to format and content including rate and distribution of alert type, source and destination address, etc. Statistical models could drive synthesis of alert streams with similar characteristics.
4. **Plan large-scale experiments**. Detailed attack and detection models could support proof-of-concept experiments analyzing the placement of intrusion detection technologies and for synthesis of alert streams to be input for modeled or real detection and correlation systems. A large-scale, theatre-wide scenario would consist of detailed descriptions of the networks and hosts, the cyber-mission supported by these assets, realistic adversary(s), attacks, and scenarios. Event-timelines seen at different parts of the network or computer system would support the scenario and could be created by hand or synthesized by a software simulation. Events would correspond to background and adversary attack actions instantiated in synthetic intrusion detection system alerts, network or system events, and application and operating system logs. Modeled intrusion detection and correlation systems could use one or more of these event streams as input and could be "placed" within this framework. Background traffic and modeled attacks could be "launched" by replaying or synthesizing those event streams. One result could be postulated alerts that could be assessed. This would help understand how the placement and operation of detection systems affects overall ability to detect and recognize an adversary's attacks at the correlation or higher level. With sufficiently realistic underlying models, these simulations could provide synthetic alert streams as testing input for real correlation and higher-level cyber-security tools in limited experiments.

6. Summary

Tools and techniques from the 1999 Evaluation are being extended in three ways. First, the Lincoln Adaptable Information Assurance Real-time Testbed, LARIAT, is a testbed and collection of GUI-based software tools that allows intrusion detection and correlation researchers to run configurable, real-time test runs in their own labs. LARIAT provides mechanisms to setup and control background traffic generation software and to configure and launch scripted attacks during a test run. Second, two "Lincoln Scenario" datasets have been generated and distributed. These short focused datasets each contain realistic background traffic and all steps of a single attack scenario: scanning, probing, break-in, installation and launching of a Distributed Denial of Service attack. Network traces and Solaris BSM data from two hosts are available with three types of labeling. Finally, the 1999 evaluation provides a basis for modeling attacks and intrusion detection systems. Feature-Analysis models are proposed that characterize intrusion detection systems and attacks with regard to the evidence left by an attack or the input features extracted and analysis performed by an intrusion detection system. These models could support several research efforts including proof-of-concept and probabilistic simulations of intrusion detection systems to study intrusion detection system coverage and alert content. Such simulations could reduce the need for costly evaluations and can help design evaluations that best suit the systems being evaluated. Alerts from operational intrusion detection systems could be modeled and synthesized to yield realistic alert streams for researchers. Large-scale experiments based on these models could help determine where to place intrusion detection and correlation systems for best performance. Finally, models and simulations with sufficient realism could provide streams of synthetic alerts to be used to support development of detection and correlation systems.

7. Acknowledgements

We would like to thank Cathy McCollum, DARPA, and the Air Force for supporting these efforts. We would also like to thank Sandia National Laboratories, researchers from Honeywell, UCSB, and SRI, and participants in the May 2000 Evaluation Re-Think meeting for their contributions to the scenario datasets and for their extensive analysis efforts that are the basis for attack and intrusion detection system modeling. Finally, we would like to thank Scott Lewandowski for providing extensive comments on drafts of this paper and Dave Fried, Steve Goulet and Oliver Dain for their work on the testbeds.

8. References

1. R. Lippmann, J. Haines, D. Fried, J. Korba, K. Das, “The 1999 DARPA off-line intrusion detection evaluation,” *Computer Networks*, 34, 2000, 579-595.
2. R. P. Lippmann 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, (Eds.) Springer Verlag. Lectures in Computer Science Vol. 1907, 2000, pp. 162-182.
3. N. Puketza, K. Zhang, M. Chung, B. Mukherjee, and R. A. Olsson, “A methodology for testing intrusion detection systems,” *IEEE Transactions on Software Engineering*, 22, 1996, pp. 719-729.
4. N. Puketza, M. Chung, R. A. Olsson, and B. Mukherjee, “A Software Platform for Testing Intrusion Detection Systems,” *IEEE Software*, September/October, 1997, pp 43-51.
5. Richard P. Lippmann, David J. Fried, Isaac Graf, Joshua W. Haines, Kristopher R. Kendall, David McClung, Dan Weber, Seth E. Webster, Dan Wyschogrod, Robert K. Cunningham, and Marc A. Zissman, “Evaluating Intrusion Detection Systems: the 1998 DARPA Off-Line Intrusion Detection Evaluation”, in Proceedings of the 2000 DARPA Information Survivability Conference and Exposition (DISCEX), Vol. 2, January 2000, IEEE Press.
6. A public web site at http://www.ll.mit.edu/IST/ideval/index.html contains information on the 1998 and 1999 evaluations. Follow instructions on this web site or send email to the authors (rpl or jhaines@sst.ll.mit.edu) to obtain access to a password-protected site with complete up-to-date information on these evaluations and results.
7. B. Wood and R. Duggan, “Red-Teaming of Advanced Information Assurance Concepts,” Proceedings of DARPA Information Survivability Conference and Exposition, Hilton Head, SC, Jan. 25-27, 2000
8. G. Schudel and B. Wood, “Modeling Behavior of the Cyber-Terrorist,” submitted for consideration by the 2000 IEEE Symposium on Security and Privacy.
9. Solaris sadmind vulnerability: http://www.securityfocus.com/frames/?content=/vdb/bottom.html%3Fvid%3D866
10. Mstream DDoS tool: http://www.securityfocus.com/archive/82/57690
11. http://www.ietf.org/html.charters/idwg-charter.html
12. T. H. Ptacek and T. N. Newsham, “Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection”, Secure Networks, Inc. Report, January 1998.
13. J. Haines, R. Lippmann, D. Fried, J. Korba, and K. Das, “Design and Procedures of the 1999 DARPA Off-Line Intrusion Detection Evaluation”, MIT Lincoln Laboratory Technical Report, In Press, December 2000.
14. J. Korba, “Windows NT Attacks for the Evaluation of Intrusion Detection Systems”, S.M. Thesis, MIT Department of Electrical Engineering and Computer Science, June 2000.
15. K. Das, “The Development of Stealthy Attacks to Evaluate Intrusion Detection Systems”, S.M. Thesis, MIT Department of Electrical Engineering and Computer Science, June 2000.
16. R. P. Lippmann and R. K. Cunningham, “Guide to Creating Stealthy Attacks for the 1999 DARPA Off-Line Intrusion Detection Evaluation”, MIT Lincoln Laboratory Project Report IDDE-1, June 1999.
17. K. Kendall, “A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems”, S.M. Thesis, MIT Department of Electrical Engineering and Computer Science, June 1999.
18. The Lawrence Berkeley National Laboratory Network Research Group provides tcpdump at http://www-nrg.ee.lbl.gov/.
19. Skaion Corporation: terry@skaion.com or steve@skaion.com.
20. D. Alessandri, “Using Rule-Based Activity Descriptions to Evaluate Intrusion Detection Systems,” in *Recent Advances in Intrusion Detection, Third International Workshop, RAID 2000 Toulouse, France, October 2000, Proceedings*. H. Debar, L. Me and S. F. Wu, (Eds.) Springer Verlag. Lectures in Computer Science Vol. 1907, 2000, pp. 183-196.
21. http://www.nmap.org
22. R. Durst, T. Champion, B. Witten, E. Miller, and L. Spagnuolo, “Testing and evaluating computer intrusion detection systems”, *Communications of the ACM*, 42(7), pp. 53-61 (1999)