LARIAT: Lincoln Adaptable Real-time Information Assurance Testbed[1][2]

Lee M. Rossey, Robert K. Cunningham, David J. Fried, Jesse C. Rabek, Richard P. Lippmann,
Joshua W. Haines, and Marc A. Zissman
Lincoln Laboratory, Massachusetts Institute of Technology
244 Wood Street
Lexington, Massachusetts 02420-9108
{lee,rkc}@sst.ll.mit.edu

Abstract—The Lincoln Adaptable Real-time Information Assurance Testbed, LARIAT, is an extension of the testbed created for DARPA 1998 and 1999 intrusion detection (ID) evaluations. LARIAT supports real-time, automated and quantitative evaluations of ID systems and other information assurance (IA) technologies. Components of LARIAT generate realistic background user traffic and real network attacks, verify attack success or failure, score ID system performance, and provide a graphical user interface for control and monitoring. Emphasis was placed on making LARIAT easy to adapt, configure and run without requiring a detailed understanding of the underlying complexity. LARIAT is currently being exercised at four sites and is undergoing continued development and refinement.

Keywords: Real-time, quantitative, repeatable, realistic, automated, testbed, evaluations, intrusion detection

TABLE OF CONTENTS

1. INTRODUCTION

DARPA off-line intrusion detection evaluations performed in 1998 and 1999 at MIT Lincoln Laboratory provided comprehensive technical evaluations of the accuracy of research intrusion detection systems [1-8]. These evaluations assessed the performance of DARPA-funded intrusion detection technology and supported 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. 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, authored 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 alternative technical approaches.

Off-line datasets are available from these evaluations. They contain extensive examples of normal and attack traffic run on a realistic testbed network. These datasets include network traces, Solaris BSM and Windows NT auditing logs, other log files, and file system information. They allow researchers to easily and quickly perform many identical trial runs with different intrusion detection techniques. These datasets are unique and have been used by many researchers. More than 160 sites have downloaded the data from these evaluations to test and develop intrusion detection systems.

As extensive as the datasets are, the DARPA evaluations were nevertheless limited. They used a reasonable, but not exhaustive, set of attacks with a limited set of actions performed as part of each attack. They also used a simple network topology, a non-restrictive security policy, a limited number of victim machines, probabilistic low-volume background traffic, and simple scoring. In addition, they evaluated ID systems that detected only atomic attack actions instead of correlating many component attacks to detect attack scenarios. The off-line format also limited the evaluations to passive intrusion detection systems that can operate in an off-line mode. This format is difficult to use with systems that query hosts or other network components, with systems that respond by changing network or host configurations, and with commercial systems that must be connected directly to an actual network or installed on a host. A separate real-time component of the evaluation [11] responded to some of these needs. However, it was time consuming to run since evaluators had to setup and run the

[1] 0-7803-7231-X/01/$10.00/© 2002 IEEE

[2] This work was sponsored by AF/ESC under Air Force Contract F19628-00-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

evaluation at one central location for each intrusion detection system that was evaluated.

The DARPA evaluations provided insight into desired characteristics of software that support IA evaluations. 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 attacks run successfully, cleaning up, and scoring putative detection alerts is more complex than expected. Complete automation of background traffic generation and attack launch coordination is essential, along with automated network and host initialization, attack verification and cleanup, and integrated scoring software. This software must enable short, repeatable test runs to separately evaluate different aspects of IA system performance.

The Lincoln Adaptable Real-time Information Assurance Testbed (LARIAT) was developed to address the most important limitations of the DARPA evaluations and form a next-generation evaluation testbed that builds on past experience. It supports both development and evaluation of IA systems, it can generate both component attacks and multi-component attacks, it automates many of the time-consuming components of past evaluations, it can be distributed and run at many sites, it can support complex hierarchical testbed networks that include defensive technologies such as firewalls, it can sustain high traffic rates, and test runs are repeatable and easily configured. We are aware of no other evaluation systems with such capabilities. Past evaluations reviewed in [1] have been difficult to configure and run. Recent commercial intrusion detection evaluations [2] have addressed some of the past limitations like the testbed complexity but still do not consider false alarm rates, and only use a limited range of attacks. The evaluations performed by Shipley [2] answer some important questions about the usability of available commercial technology, such as the ability to keep up with high traffic rates and the ability to remotely manage and control devices across an enterprise. Our work tends to focus on measuring the detection and false alarm rates of systems in a quantitative and repeatable manner with a goal of providing the technology and recorded network traffic to the community so that better intrusion detection technology can be developed. Although some systems provide components similar to those in LARIAT we are aware of no system with all the desired capabilities. The remainder of this paper describes the design of LARIAT, current uses, and future plans.

2. LARIAT

Two design goals were established for LARIAT: (1) support real-time evaluations and (2) create a deployable, configurable and easy-to-use testbed. Previous evaluations were time consuming and expensive. Our experience is that roughly four months are required to produce high quality evaluation results for every new ID system tested. Tasks performed during this time include learning about the new ID system, configuring it to work in an off-line mode, developing software to interpret the native output format, running the evaluation, and saving and interpreting the results. LARIAT eliminates the requirement for off-line operation, and greatly simplifies the setup and operation. We developed an architecture that ties all the software components together and provides complete automation of all evaluation phases. We have automated traffic generation, attack scheduling, system configuration, experiment runs and analysis.

The remaining sections explain LARIAT. Section 2.1 describes the phases required to perform an evaluation, and how LARIAT automates and controls these steps. Section 2.2 describes how the user configures the experiment and the background traffic configuration. Section 2.3 covers the attacks. Section 2.4 describes the software implementation and Section 2.5 describes the hardware used in the base testbed.

2.1 Experiment Steps—

An experiment is composed of the eight steps shown in Figure 1. These steps represent the flow of an experiment. Starting at the top of the figure, a user selects a profile that specifies the background traffic, the attacks to be run, and the experiment duration. The remaining seven steps are automated and controlled by the director software. The director allows the user to control testbed capabilities without having to interact with any of the individual hosts, thereby relieving the end-user from detailed knowledge of LARIAT. The automated phases are described below.

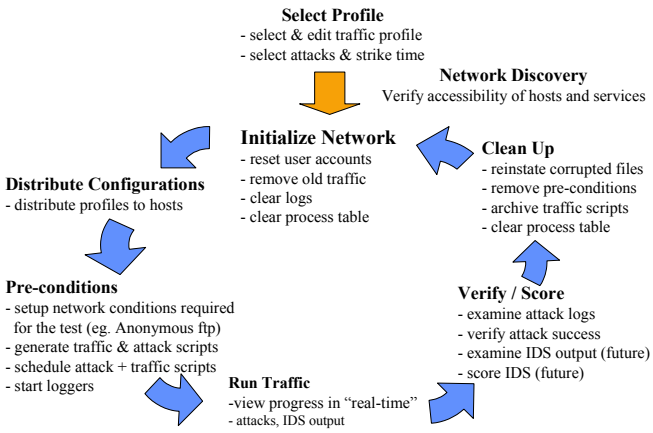

Figure 1 - Automated run sequence. After a user selects a profile LARIAT automates all the phases of an experiment.

Initialize Network—

During this step the test network is initialized. User accounts, log files and processes are all set to pre-specified configurations.

Distribute Configuration—

The traffic configuration and experiment details are distributed to each host.

Pre-Conditions—
Tasks are performed that prepare for the flow of synthetic traffic. The traffic generators synthesize user traffic by building scripts for each background traffic session. These scripts use a custom extension [6] to the Expect scripting language [18], [19] that was developed for the 1998 and 1999 evaluations. 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 is possible. Attack scripts are also prepared so that both the attack and background traffic scripts can be scheduled to run at the appropriate times on each traffic generator and attacker.

Run Traffic—
At this point the evaluation commences. During the run, the progress of background traffic and attacks can be viewed in real-time via the GUI.

Verify and Score—
Upon completion of the run, or at a specified interval after each attack, the system scans attacker logs and searches for evidence of the attack on the victim host, in an effort to verify the successful completion of each attack. Alerts from the ID systems are stored for analysis so that they can be compared against a verified list of attacks.
Generation of background traffic is important because it provides real user traffic on the network within which attacks will be injected. Ideally an ID system should only send alerts when it identifies real attacks (true detections) and not when a user performs a legitimate action (false alarm).
In future versions, LARIAT might be able to score an ID system by interpreting native ID alerts. Currently two problems exist. First each vendor uses a proprietary message alert format that would require a custom parser to interpret the message content. The IETF community is overcoming this problem by creating the IDMEF standard to standardize the ID alert format; though currently the message content format in not standardized and varies between users.

Clean Up—
After the verification and scoring, cleanup scripts, specific to each attack, remove evidence of that attack run, resetting any changes made by the attack script. A cleanup can range from reinstating a corrupted file to restoring an entire hard drive from a stored image. Hosts involved in background traffic are also re-initialized. Continuing 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. After this last step additional experiments can be run.

2.2 Background Traffic—

To start a test run, a user must first select a profile for the background traffic. A background traffic profile defines the operating environment to be simulated by the testbed and contains information such as the type of services (eg. http, ftp) that will be emulated, the statistical distribution of each service over the course of a day, and the traffic rate. The background traffic profile can be modified with respect to the content and distribution of services.
The traffic profiles were determined by recording and analyzing the network traffic distribution and composition from a US military base. New traffic profiles can be added and configured to suit a particular network environment.
Figure 2 shows how a user can setup an experiment. The top portion in the panel allows the experimenter to control experiment duration and select a previously defined background traffic profile. By editing these profiles a user can quickly setup and run an experiment. Finer control of traffic generation can be achieved by selecting the *Advanced* button.
Figure 3 shows a screen capture of the 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 3 shows how the user specifies the amount of FTP traffic to be generated, with the profile for the FTP session arrivals per 15-minute time interval, on a similar graph (plotted by itself). 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.

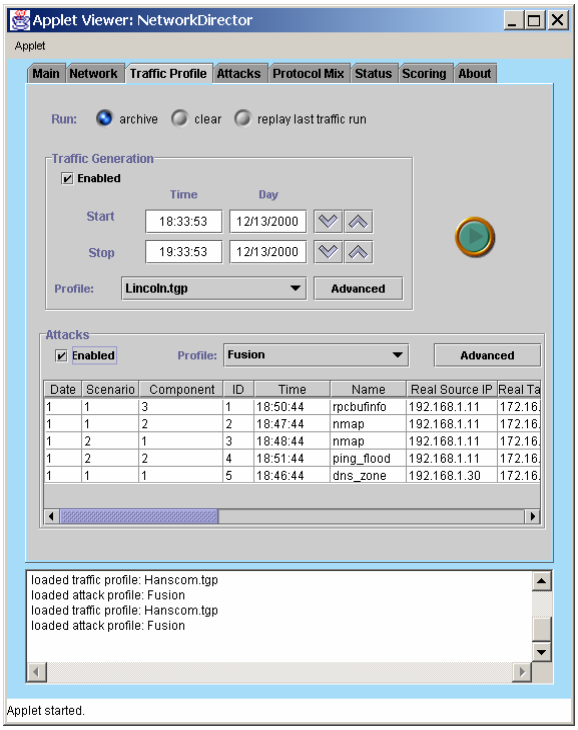

Figure 2 - The LARIAT GUI profile-selection panel.

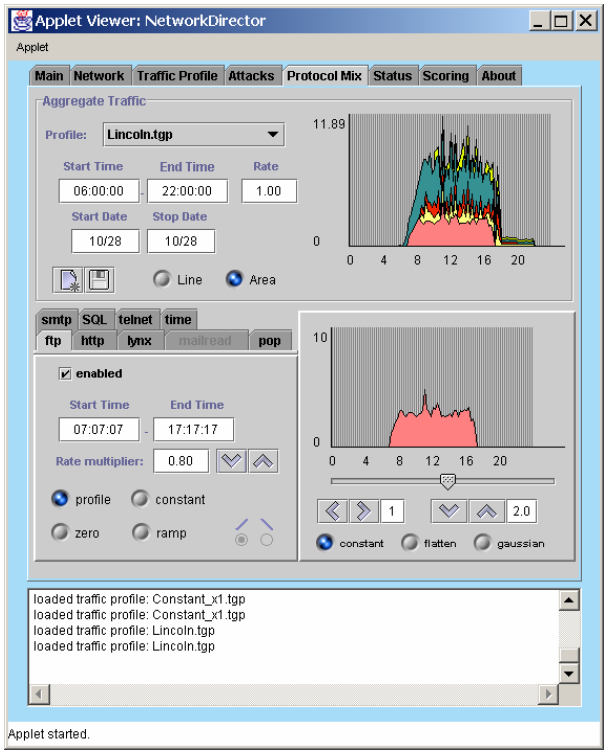

Figure 3 - The LARIAT GUI profile-editing panel. Here the user can modify the background traffic profile.

User sessions for each traffic type are constructed from probabilistic models of user actions in a manner similar to those of the 1999 evaluation data [6]. Although the interface controls the arrival of session start times, the system has internal models of managers, programmers, secretaries and system administrators.

Figure 4 show an example of the probability of executing commands for 5 programmers using telnet.

2.3 Attacks—

LARIAT manages attacks to be run during an experiment via the director GUI. This permits LARIAT to provide a consistent, simple interface to the experimenter while supporting the scheduling and execution of a large number of attack types against a variety of network hosts and across numerous operating systems releases.

2.3.1 Attack Framework—

The attack framework provides the following major capabilities: (1) attack component abstraction (2) an API to simplify adding and creating attacks (3) management and composition of attack components (4) an attack scenario model (5) an attacker's knowledge-base.

2.3.1.1 Attack Component Abstraction and Management—

Attack components developed for LARIAT do not reference the network configuration. As a result, we are able to reuse the same components with different variations or options to suit a particular attacker's goal. The abstraction also enables deployment to organizations with different network addresses and topologies. The attack components themselves are modified to accept all the relevant parameters as arguments, which are then supplied at runtime.

A separate XML file, used to describe all the properties of an attack, is created that accompanies the exploit code. Information contained within the description file includes the required parameters, information about what the attack requires to run, what it provides when complete, the skill level of the attacker, the visibility of the attack and other related information. To simplify the process of describing each attack for the user, some information is automatically extracted from the NIST ICAT meta-database [17] and incorporated into the attack component description file.

To organize and manage the attacks, each attack component

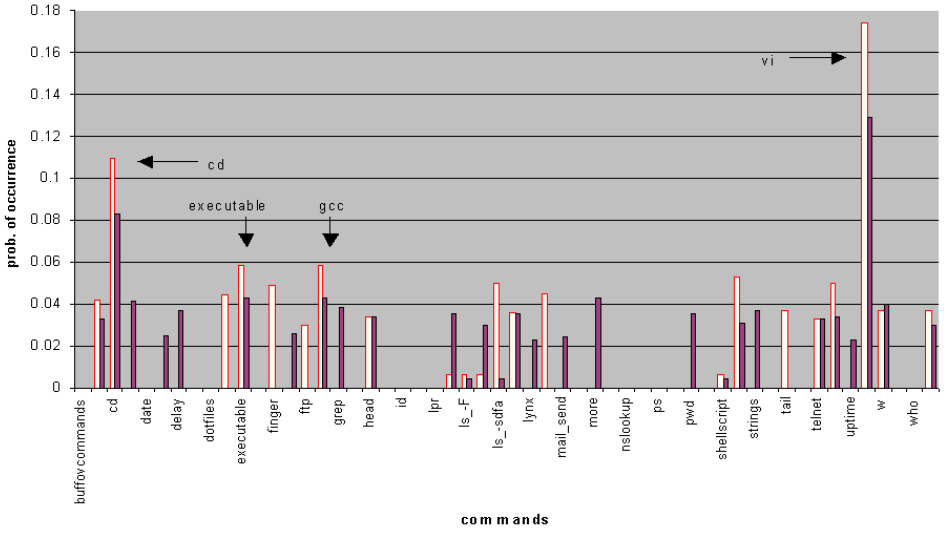

Figure 4 - Probability of executing commands for 5 programmers using telnet

is catalogued by it's CAN or CVE number [16] [17] and a short textual description. For example, the sadmind attack is stored in a directory called CVE_1999_0977_Buffer_Overflow_in_Solaris_sadmind. The directory with all the attack and supporting code is encrypted until required. The decryption, setup and re-encryption steps are performed during the run's "Pre-condition" and "Cleanup" phases shown in Figure 1. The attack description file is stored unencrypted so that it can be processed by the director or by a user who is interested in knowing about the attack without having to read the attack code. The director loads the attack component description files so that they can be searched, manipulated and displayed to the user via the GUI. Before a run is started the director will process the attack description files involved in an experiment and substitute the attack parameter variables with the specific testbed values.

2.3.1.2 Attack Framework API—

New attacks are incorporated into LARIAT using an attack component API. The API supports the capability to launch attacks in their native format (perl, shell script, binary, etc.) and can be used to store and retrieve information about attacks into a common data repository.

2.3.1.3 Attack Scenario Model—

Figure 5 shows an example of the attack model in which a series of attack components are linked together to exploit a Microsoft IIS and take control of the host platform. The attack model is similar to the requires/provides model [12] and has been implemented to provide the following capabilities. First it ensures that as attack components are linked together, that the information provided by one component is what the next one requires. This is especially helpful for a user that lacks detailed knowledge of the attack components. Second it is used to ensure that if a component fails during a run that any subsequent components that depended on the information provided are not executed. For example if an external scan used to identify a firewall fails to execute correctly then we should not launch any components that utilized information provided by that component such as an internal network scan or a remote-to-local exploit to a host protected by the firewall. Once the dependencies are specified for each component, the framework ensures the correct execution.

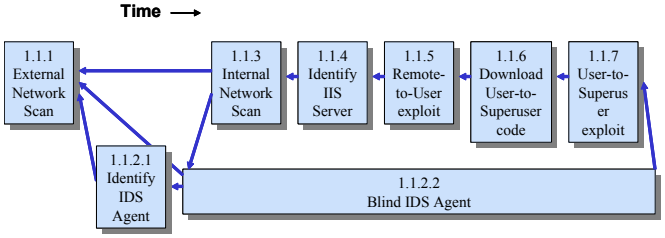

Figure 5 - Attack scenario using the LARIAT attack model

All scenario run-time information is stored in the attacker knowledge-base. Currently the knowledge-base informs the remote attacking hosts which attacks to launch and their parameters, records whether the attacks launched successfully, ensures that all the requirements for a particular attack are satisfied before running and stores the eventual success or failure of the attack. The knowledge base is also used by attack components to store and retrieve any information gained during a scenario. This provides a means to pass information gained from one component to the next. By using the API to store and retrieve information we can ensure interoperability of the attack components.
The attack knowledge-base is currently implemented as an XML file that is sent during the network initialization phase to every host involved with an attack. The knowledge-base is distributed to every host to ensure that there are no control traffic artifacts within the network traffic during a run. At the end of a run the knowledge-base is retrieved by the director to help determine ground-truth and display any information or error messages to the user.

2.3.2 Attack Profile—

Attack profiles abstract and manage the set of attack scenarios used in an experiment. An example of an attack scenario is the recent DDoS dataset created by Lincoln Laboratory [5]. In this scenario an attacker compromises several internal hosts protected by a firewall; installs attacker software on the remote hosts, and signals a coordinated denial of service attack against a site on the Internet.
An attack profile contains one or more attack scenarios of arbitrary length. The LARIAT director is used to load an attack profile, as shown in the lower portion of Figure 2. The attack profile can be interactively modified and saved from the director before it is executed. Attack profiles contain multiple attack scenarios to allow different scenarios or variations of a single scenario to be run. This feature allows the thorough testing of a particular hardware device, software system, or the scenario itself.

2.3.3 Attack Scenarios—

Attack scenarios are a temporal sequence of atomic attack components. Attack components can be either exploits, such as a buffer overflow, support functionality such as scans or code transport, or other techniques used by an attacker. Individual attack components are aggregated into attack scenarios with the intention of recreating an attacker's actions with the greatest possible fidelity. Scenarios typically begin with elements of network discovery, such as probing and scanning. Next, a victim host is compromised. Finally an attacker captures a "flag" and removes evidence of the security breech.
We have developed six scenarios that are representative of different attack classes; one is shown in Figure 6 and illustrates part of the current LARIAT testbed used at Lincoln Laboratory. Here LARIAT is configured as a stand-alone network with no external network connections. A firewall is used to separate the internal network from the

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.