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

[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.

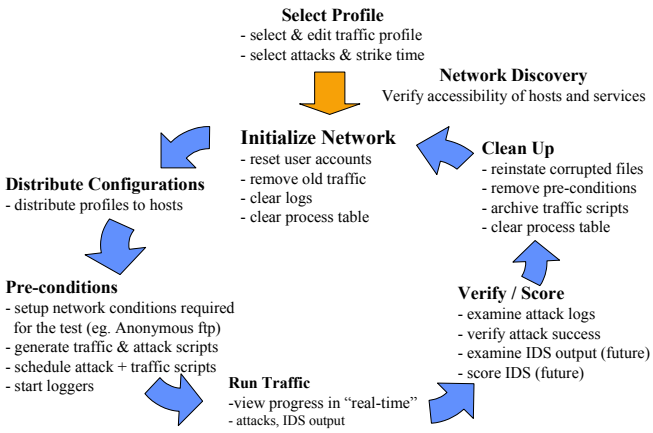

Figure 1 - Automated run sequence. After a user selects a profile LARIAT automates all the phases of an experiment.

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.

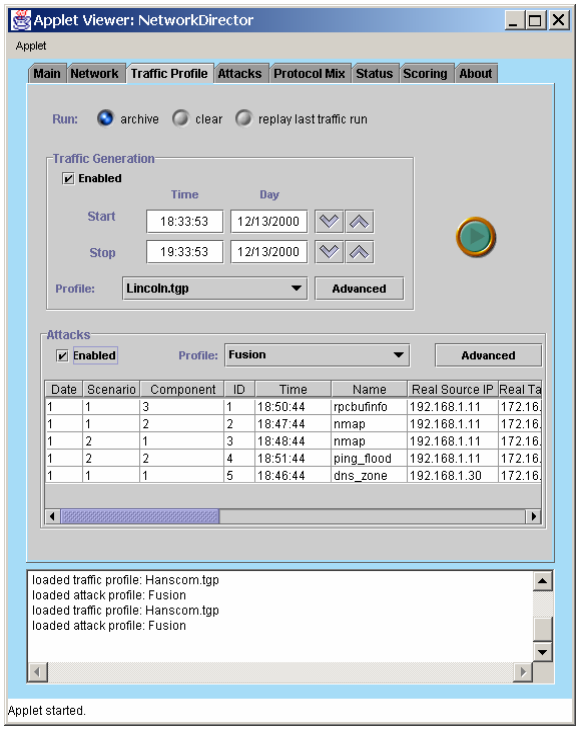

Figure 2 - The LARIAT GUI profile-selection panel.

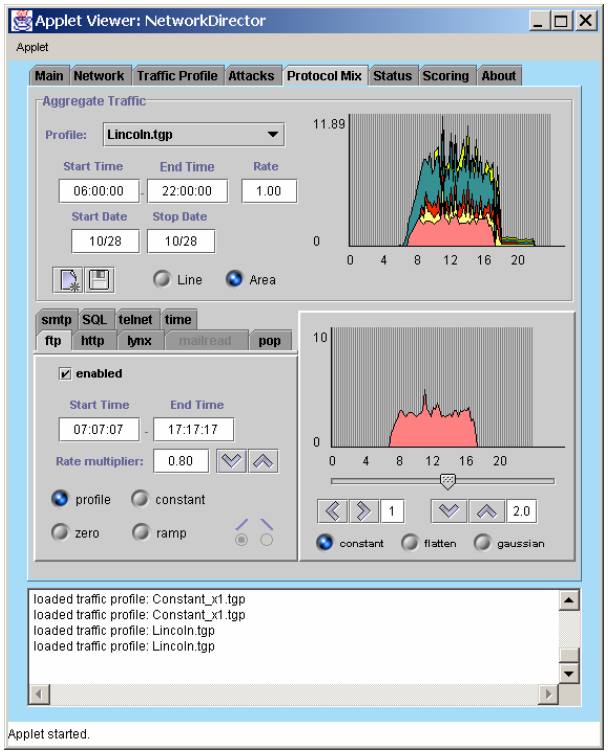

Figure 3 - The LARIAT GUI profile-editing panel. Here the user can modify the background traffic profile.

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

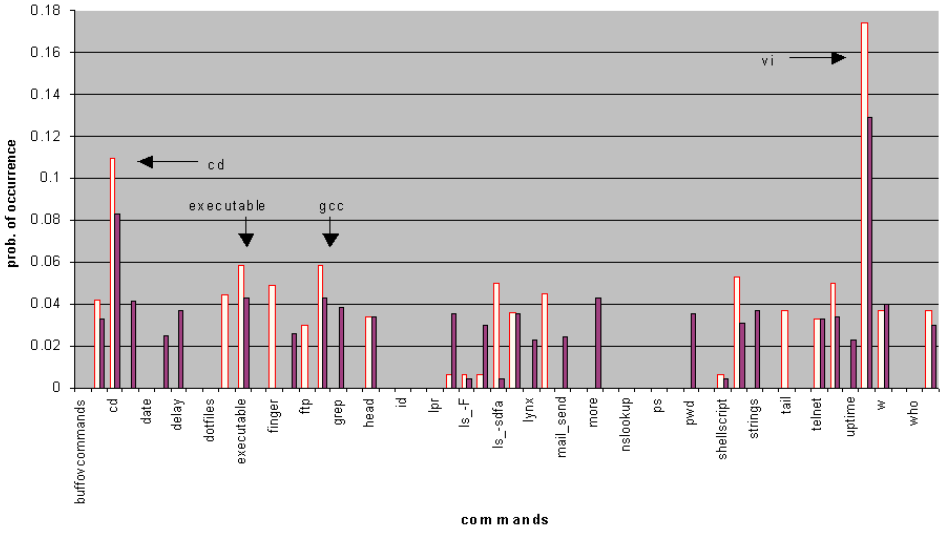

Figure 4 - Probability of executing commands for 5 programmers using telnet

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.

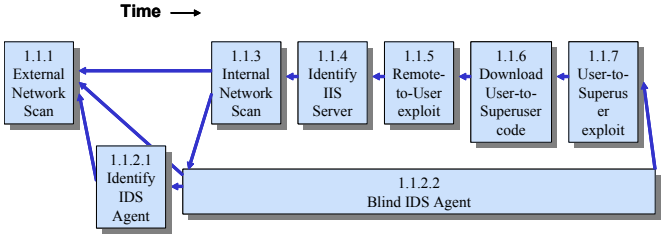

Figure 5 - Attack scenario using the LARIAT attack model

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

emulated Internet and is running NAT on the internal addresses. The only ports allowed inbound are http to an Apache web server running on a host on the DMZ, ssh to an internal Solaris 8 server and mail to another internal server. Only TCP ports above 1024 and a few privileged services are allowed outbound.

Some example run-time parameters specified in the knowledge-base include the date and time the attack is scheduled to execute; any attack components that must have either started, completed or be running; the real host that will launch the attack and the target hosts IP address. Note that these are all parameters that can be adjusted. For example, the date and time values specifying when an attack should be launched can be specified in the attack profile as either an absolute time e.g. (fourth day of the experiment at 13:34:3) or relative time e.g. (7 minutes after some other experiment event). The actual times are supplied at run time. Additionally any insider knowledge or a priori information can be incorporated into the attack scenario by pre-populating the attacker knowledge-base.

2.3.4 Old Attack Components—

The LARIAT attack corpus benefited from the reuse of attack components developed for the 1998 and 1999 DARPA evaluations. These attack components tend to exploit flaws in older versions of operating systems, applications and networking infrastructure components. Some of these attack components are still viable, although some are no longer effective against the newer operating systems and applications in the base LARIAT network. These attack components add richness to the attack corpus by allowing for a greater range of operating system versions and provide a source of realistic unsuccessful attacks.

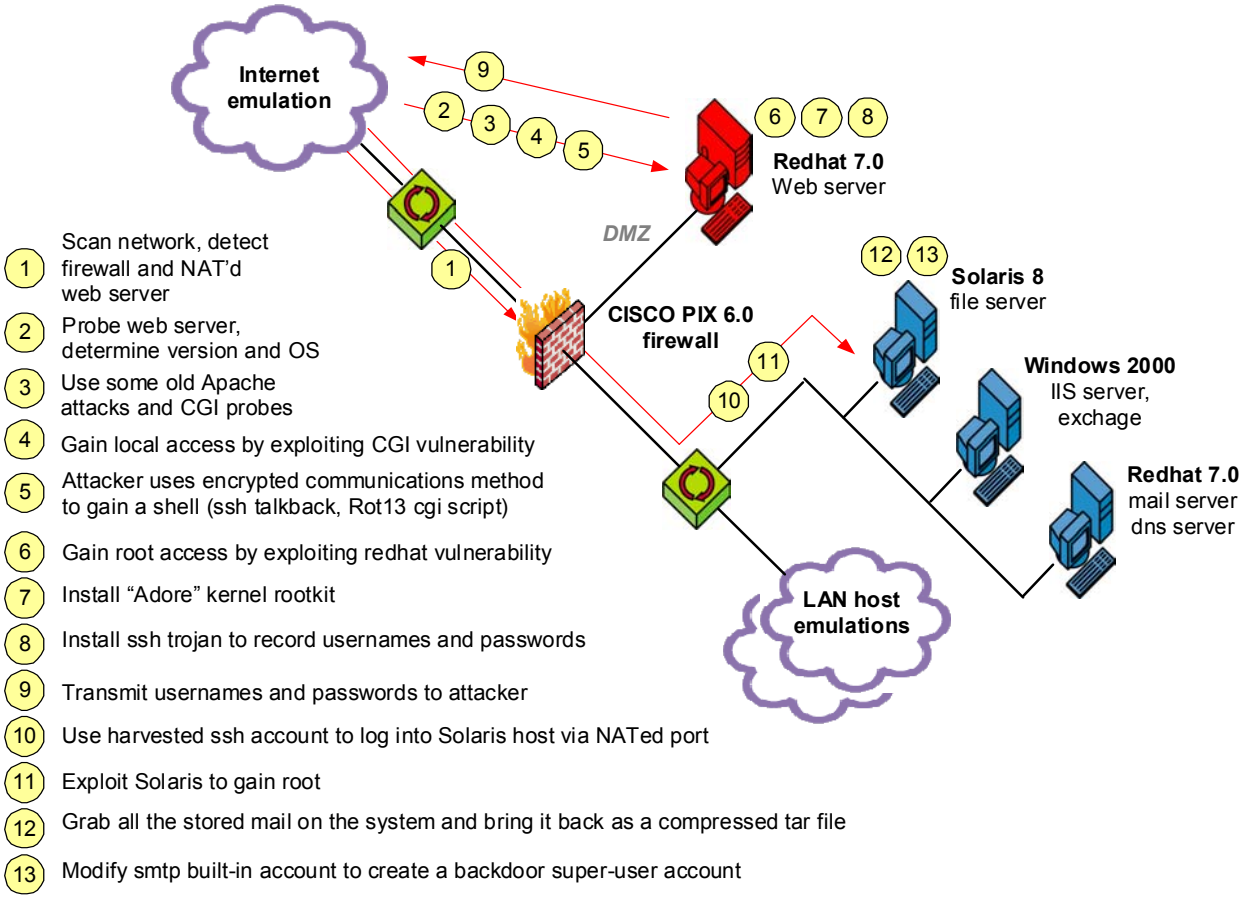

Figure 6 - A sample attack scenario available with LARIAT

2.3.5 New Attack Components—

Newer attack components, which have been developed since the DARPA evaluations, are being incorporated into the LARIAT attack corpus to support attacks against current operating systems. The new attack components can be combined in attack scenarios with the older attack components to provide a greater fidelity to an attackers action. For example, an attack scenario might include an attacker determining the type of operating system a victim host is running. The attacker might then launch an older exploit. Once it fails, a newer exploit might successfully compromise the victim host. The new attacks provided in addition to those available in the 1998 and 1999 datasets [6], [7], [8], are targeted towards Solaris 2.7 and 2.8 for both SPARC and x86, Windows 2000 and Linux hosts.

2.3.5 Attack Helpers—

LARIAT includes tools to support launching attacks and transferring files. The "Attack Helper" module integrates attack information into the LARIAT system and permits wrappers, automatically generated by the LARIAT framework, to launch the attack. The only information required by the "Attack Helper" is the name of the attack, command line arguments, and a unique numeric identifier (supplied by the director). The "Xfer" module is provided to facilitate the transfer of files using various services (e.g. FTP, TFTP, telnet, rcp, scp, smtp) and encodings (e.g. Uuencoded, encrypted, ROT13, gzip) while offering a consistent interface to expand the range of attacks. Typical uses of the "Xfer" module include transporting exploit code to a victim host and retrieving a "flag" file from a victim host.

2.4 Software Implementation—

Figure 7 represents the major software components present in LARIAT. The director is implemented as a signed Java applet. It can be run in any recent web browser. To ensure portability of the data, all the experiment and network-specific details are captured in an XML configuration file. XML was also chosen because both the Java director and all the perl scripts and modules resident on the individual hosts can interpret the same configuration data. This abstraction of the experiment information is important because all the traffic generators can have identical software and settings, while the traffic produced is based solely on the configuration data. This also allows for reconfiguration of the hosts/network as well for scaling of traffic quantity.

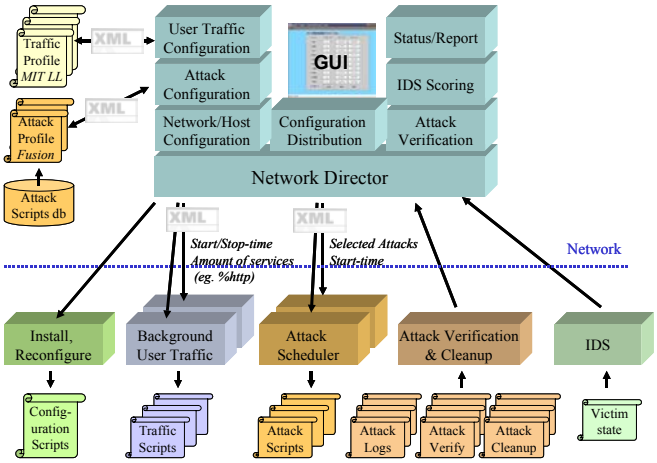

Figure 7 - Software components

To support scaling of traffic volume and richer network configurations LARIAT uses user groups. Each traffic generator creates one user group that includes a configurable number of users, each with a different IP address covering one or more subnets. A traffic generator can currently support about 500 concurrent user sessions. Two user groups (an inside and an outside) are configured by default. Additional user groups can be supported by adding traffic generators. Our experience has shown that three traffic generators are sufficient to saturate a 100 Mbps network. Real hosts are necessary because these are the platforms that host ID systems. 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. Users, via the director, can configure the XML configuration file that tells the background traffic generators how much and what types of traffic from that user group are directed towards other user groups.

The testbed provides scripts to facilitate changes to network topology. These scripts rebuild Domain Name Service (DNS) tables for both the real and virtual hosts in each domain, custom telnet logins and message-of-the-day banners for every virtual host, and user profiles to match the updated network.

2.5 Supporting Hardware and Software—

Figure 8 shows the actual hosts used to implement the network shown in Figure 6. The left side of the figure simulates the Internet while the right side simulates the Intranet being protected and against which attacks are launched. The network director (GUI) controls the flow of the experiment and provides the interface to the user. In Figure 8 it is running on the Windows 2000 server on the internal network. The two traffic generators run Redhat Linux and a special version of the Linux kernel, provided by Skaion Corporation [10], which allows a single host to emulate multiple IP addresses. Custom software written for the traffic generators produces user traffic that allow the inside traffic generator to look like many hosts on the internal network, and the other outside traffic generator to

look like many hosts around the Internet [11]. 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 traffic generation software. The Internet Server on the far left of Figure 8, serves as the root Domain Name Service (DNS) server for the testbed and mirrors web and ftp server content from thousands of real sites. It uses the same Linux kernel provided by Skaion to emulate the site's IP addresses. Web and FTP content can also be retrieved via an existing Internet connection if desired. Attacks can be launched from any host but they mostly originate from a single attacker machine. The attacker's IP address can be selected at attack time from IP addresses used by the traffic generator. The hosts exploited by the attacks are real unmodified servers running popular services such as http, ftp and smtp. These servers respond to the requests from the virtual users on the traffic generators to recreate the legitimate use of network services. The servers exploited by the attackers in this configuration are a Solaris 8 (x86), Linux RedHat 7.0, and Windows 2000 as these are the new platforms for which attacks have thus far been incorporated; however, any platform could be added as a target server. The real servers on the right side of Figure 8 can run host-based IDS if desired.

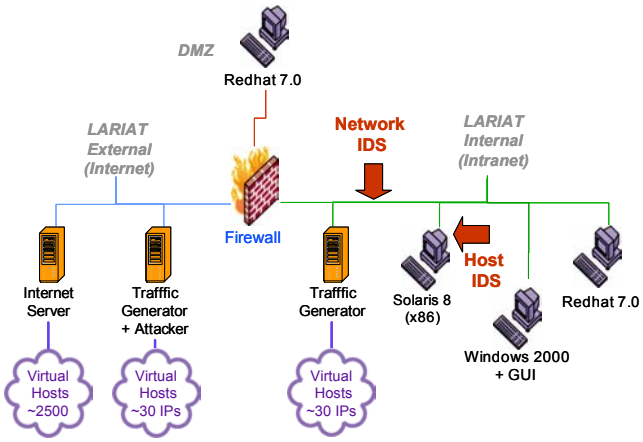

Figure 8 - LARIAT minimal hardware configuration. Victim hosts and traffic generators can be added as needed

3. USES OF LARIAT

LARIAT is being used by a variety of communities. Researchers use LARIAT to develop new technologies; independent laboratories use LARIAT to measure ID system strengths and weaknesses; and enterprise users use LARIAT to make purchasing decisions. The following sections describe an example of each.

3.1 IDS Development—

We at Lincoln Laboratory have been using LARIAT for development and verification of information assurance technology such as host-based intrusion detection [13] and intrusion alert fusion [14]. The network that we use is similar to that shown in Figure 6. This is a stand-alone configuration with no outside network connectivity. The external network simulates the Internet with about 2600 addressable virtual hosts, and the internal network simulates a small Air Force base with about 60 active hosts. The two domains are separated by a firewall. For fusion research, we install both real-time ID sensors and real-time fusion prototypes on the testbed. The LARIAT traffic causes ID system alerts, which we process by the fusion system.

Two other sites are also using the testbed for research focused on applying data-mining techniques to intrusion detection on conventional networks.

3.2 IDS Evaluation for Product Improvement—

Together with the National Institute of Standards and Technology (NIST), we have participated in discussions with commercial vendors regarding the possibility of using LARIAT to improve and evaluate commercial systems.

Two areas of concern from commercial vendors of ID systems are the relatively low traffic volume and limited amount of Windows traffic. To address the traffic volume we have made a number of enhancements to increase the 400 kbps average traffic throughput present in the 1999 dataset. It is now possible to generate a load capable of saturating a 100 Mbps network. We are also addressing the lack of Windows traffic. In the 1999 evaluation we did generate remote login sessions to a Windows NT host, but these sessions were limited and did not capture the unique characteristics of traffic typical of a Windows host logged into a Windows domain. Some of the traffic we need are Active Directory, direct hosting, file sharing and Microsoft Exchange. We are currently working on modifying a Windows host to create the appearance of a much larger set of hosts. This would be similar to what has been done on the Linux traffic generators. Once complete, we should be able to create the appearance of many Windows hosts logged onto a Windows domain and running traffic typical of a Windows host.

3.3 IDS Evaluation for Enterprise Use—

The third example of an organization using LARIAT is an enterprise laboratory tasked with evaluating and recommending network devices for use throughout their organization. This laboratory performs their own evaluations of candidate intrusion detection systems and tests many other types of IA components such as firewalls, routers and VPNs prior to deployment. The laboratory is equipped with representative components of their networking infrastructure. This allows them to run tests without affecting real users.

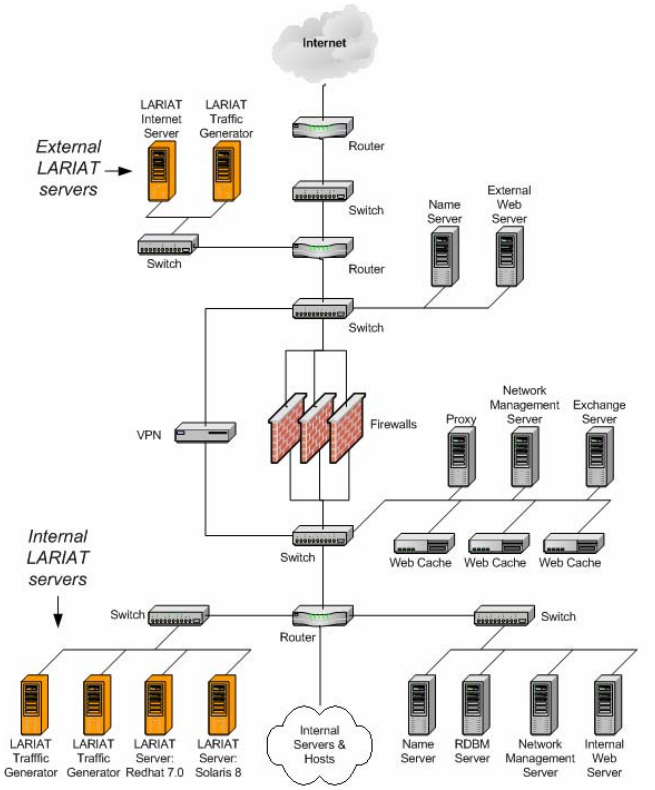

Figure 9 - Integration into an enterprise test network

LARIAT is integrated into their test network as shown in Figure 9. The external LARIAT network integrates into the external enterprise network consisting of external routers, a DMZ, and external servers; the internal LARIAT network becomes a component of the internal enterprise. This setup is different from the standalone version used in the past because (1) it integrates with an existing network and ties into DNS server configuration and naming scheme; (2) it is also connected to the Internet; (3) it must integrate with other networked devices and (4) it runs in a fully-switched environment. In this environment LARIAT has been used to test the capabilities of several high speed firewalls under consideration for use in their organization to sustain a large number of user connections.

4. Current Work

LARIAT is undergoing continuous evolution. For example, as more organizations migrate to Microsoft operating systems we have generated more Windows traffic types. Many attacks exploit specific characteristics of Windows based systems and to effectively evaluate ID systems we need to generate the appropriate type of background user traffic as well as the attacks themselves. The transition from an off-line to a real-time testbed also requires that we extend our ability to automatically verify the successful completion of attacks so that the output of an ID system can be accurately scored. We are also continually enhancing the background traffic and the attack components to ensure that they are relevant to newer operating systems and network environments.

5. Summary

To our knowledge, LARIAT provides the most comprehensive collection of tools available to recreate a network environment suitable to perform development and evaluations of ID and IA technologies. The emphasis placed on automation, configurability and ease-of-use allows LARIAT to meet the needs of a range of users. The collection of integrated and configurable attacks and realistic background traffic allow a novice operator to quickly run experiments. On the other hand, the configurability and adaptability of both the network traffic and of the testbed allow more experienced users to fine-tune experiments to suit their particular research. The open and extensible nature of the testbed also allows groups to enhance the testbed with additional traffic types, attacks, and capabilities that could benefit the whole community.

6. Acknowledgements

We would like to thank Col Howard Borst, Carmen Paludi, Lt Keith Piwowarski and their staff at the Air Force Electronic Systems Command (AF/ESC) for their sponsorship and support. We would also like to thank Steve Durst and Terry Champion from Skaion Corporation for their help enhancing their software to meet our needs for higher traffic rates.

References

[1] R. Lippmann, J. Haines, D. Fried, J. Korba, and K. Das, "The 1999 DARPA off-line intrusion detection evaluation,"*Computer Networks*, 34, pp.579-595, 2000.

[2] Greg Shipley and Patrick Mueller, "Dragon Claws its Way to the Top," *Network Computing,* August 20, 2001.

[3] R. Lippmann and J. Haines, "Analysis and Results of the 1999 DARPA Off-Line Intrusion Detection Evaluation," Debar, H., Me, L., Wu, S. F. (eds.): *Recent Advances in Intrusion Detection*, Third International Workshop, RAID 2000 Toulouse, France, October 2000, Proceedings, Lecture Notes in Computer Science, Vol. 1907. Springer-Verlag, Berlin Heidelgerg New York, 2000, pp. 162-182.

[4] R. Lippmann, D. Fried, I. Graf, J. Haines, K. Kendall, D. McClung, D. Weber, S. Webster, D. Wyschogrod, R. Cunningham and M. Zissman, "Evaluating Intrusion Detection Systems: the 1998 DARPA Off-Line Intrusion Detection Evaluation," *Proceedings of the 2000 DARPA Information Survivability Conference and Exposition (DISCEX)*, Vol. 2, IEEE Press, 2000.

[5] A public web site at http://www.ll.mit.edu/IST/ideval/index.html contains information on the 1998 and 1999 evaluations and the DDOS datasets. 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.

[6] J. Haines, R. Lippmann, D. Fried, J. Korba, and K. Das, *Design and Procedures of the 1999 DARPA Intrusion Detection Evaluation: Design and Procedures*, Technical Report 1062, MIT Lincoln Laboratory, February 26, 2001.

[7] J. Korba, *Windows NT Attacks for the Evaluation of Intrusion Detection Systems*, S.M. Thesis, Department of Electrical Engineering and Computer Science, MIT, June 2000.

[8] K. Das, *The Development of Stealthy Attacks to Evaluate Intrusion Detection Systems*, S.M. Thesis, Department of Electrical Engineering and Computer Science, MIT, June 2000.

[9] K. Kendall, *A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems,* S.M. Thesis, Department of Electrical Engineering and Computer Science, MIT, June 1999.

[10] Skaion Corporation, www.skaion.com 2001.

[11] R. Durst, T. Champion, B. Witten, E. Miller and L. Spagnuolo, "Testing and evaluating computer intrusion detection systems," *Communications of the ACM*, Vol. 42(7), 1999, pp.53-61.

[12] Steven J. Templeton and Karl Levitt, "A requies/provides model for computer attacks," *Proceedings of the 2000 New security paradigm workshop,* September 18-21, Ballycott Co. Cork Ireland, 2000.

[13] R. Cunningham, R. Lippmann, et al. "Host-based Bottleneck Verification efficiently detects novel computer attacks," *IEEE Military Communications Conference Proceedings*, Atlantic City, NJ, USA, 1999.

[14] Oliver M. Dain and Robert K. Cunningham, "Building Scenarios from a Heterogeneous Alert Stream," *2001 Proceedings of the IEEE SMC Information Assurance Workshop*, West Point, NY, June 2001.

[15] Internet Engineering Task Force. Intrusion Detection Message Exchange Format (IDMEF). www.ietf.org/html.charters/idwg-charter.html, 2001.

[16] MITRE Common Vulnerabilities and Exposures (CVE). http://www.cve.mitre.org/, 2001.

[17] National Institute of Standards and Technology (NIST) ICAT metadatabase, http://icat.nist.gov/, 2001.

[18] Don Libes, *Exploring Expect: A Tcl-Based Toolkit for Automating Interactive Programs*, O'Reilly & Associates, ISBN: 1565920902, December 1994.

[19] National Institute of Standards and Technology (NIST) Expect. http://expect.nist.gov, 2001.

Lee M. Rossey *is an Associate Staff Member in the Information Systems Technology Group at MIT Lincoln Laboratory . He previously worked at Lockheed-Martin/L-3 Communication in the Information Security program area. He has a BS in electrical and computer engineering and BA from SUNY Buffalo and an MS in electrical and computer engineering from the University of Florida.*

Robert K. Cunningham *received the Sc.B. degree in computer engineering from Brown University in 1985, the M.S. degree in electrical engineering from Boston University in 1988, and the Ph.D. degree in cognitive and neural systems from Boston University in 1998. From 1985 to 1987 he worked at Raytheon, designing and developing a parallel and distributed operating system for a next generation weather radar system. After completing his masters degree in 1988, he became a staff member of the Machine Intelligence Group at MIT Lincoln Laboratory, where his research focused on digital image processing and image understanding, including parallel implementations of algorithms for enhanced visualization and image region classification. As part of that work, he contributed to early drafts of the real-time message passing interface (MPI/RT) specification. In early 1998 he moved to the Information Systems Technology Group, where his research has focused on developing intrusion detection systems that do not require advance knowledge of the method of attack. He was appointed Assistant Leader of the group in August 2000. He is a member of Sigma Xi and a senior member of the IEEE*