Air Force Intrusion Detection System Evaluation Environment

Terrence G. Champion
tgc@sappho.rl.af.mil

Robert S. Durst
sdurst@sappho.rl.af.mil

Air Force Research Laboratory
INFOSEC Technology Office
80 Scott Drive, Bldg 1124
Hanscom AFB MA 01731-2909

Introduction

Ongoing DARPA-sponsored research in computer intrusion detection (ID) has produced emerging systems that exploit a variety of techniques such as network sniffing, filesystem checks and local host audit record inspection, or more complex correlations of distributed sensors' various reports. This research has become increasingly important as computer attacks increase in number, publicity and damage done. The Air Force Research Laboratory (AFRL) INFOSEC Technology Office has volunteered to regularly test and evaluate these emerging ID systems for possible insertion into critical national networks.

The evaluations included running the ID systems in a live network and then flooding that network with a variety of traffic, both "normal" (legitimate traffic that any office network would reasonably expect to occur, such as email, HTTP, telnet, etc.) and attacks, including fairly recent and more well-known network attacks. For the sake of the testbed's real security and the integrity of the evaluations, AFRL built a self-contained test environment to model an actual base network.

Supporting the AFRL test effort, the Massachusetts Institute of Technology's Lincoln Laboratory developed non-real-time evaluation tools for assessing the performance of individual ID systems. Their testbed, designed for repeatability, dramatically accelerated playing back huge volumes of data to quickly produce high-confidence performance measurements. However, ID systems in the field will likely have to defend larger and more complex networks and will interact with other components. This interaction and the resulting performance required AFRL's building a more complex, real-time environment.

AFRL Virtual Network and Testbed

AFRL's immersive test environment simulates the complexity of a typical metropolitan area network (MAN) found at military bases. In theory a top-level firewall protects a single point of entry into the base MAN. From the firewall, arrays of routers branch out to their respective subnetworks; these subnetworks can carry a variety of traffic services and protocols. AFRL's network model simulates the size and diversity of a base MAN, using software to dynamically assign arbitrary source protocol addresses to individual network sessions running on special traffic generation computers. For the 1998 evaluation AFRL used two traffic generators, an "outside" machine which ran network sessions between the model base and the simulated Internet, and an "inside" machine which ran network sessions within the model base's address space and which simulated the presence of a much larger network than actually represented by the physical machines. The entire testbed was completely isolated in AFRL's laboratory and was not connected to any live network. The virtual network architecture is shown in Figure 1 and the actual physical network architecture is shown in Figure 2.

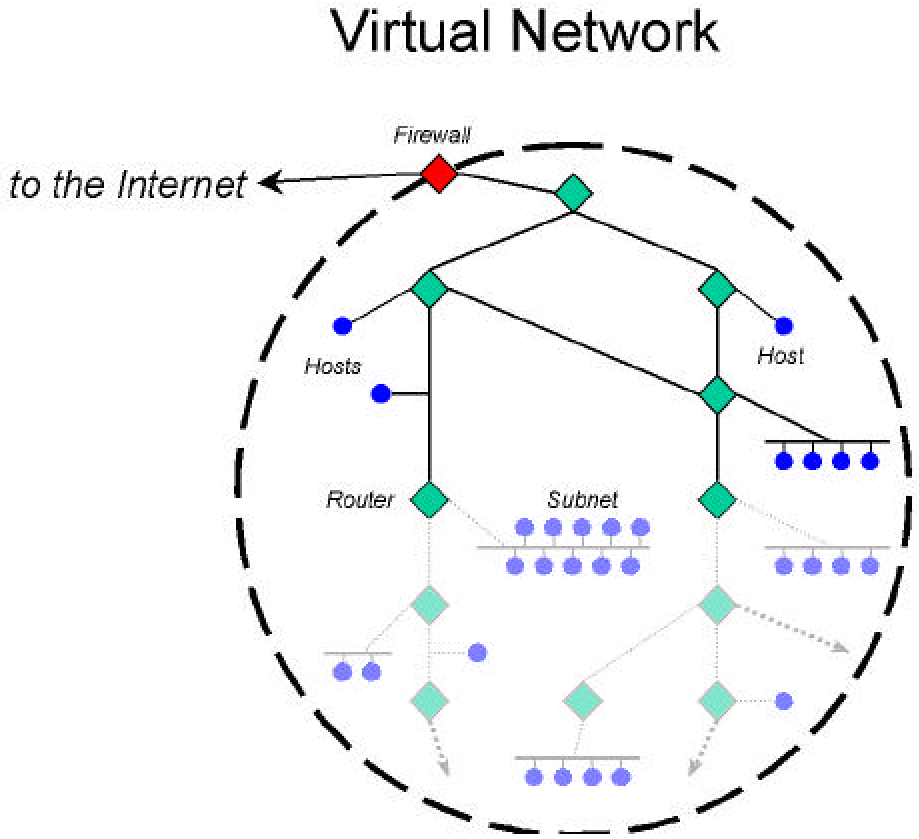

Figure 1. *Virtual test network architecture.*

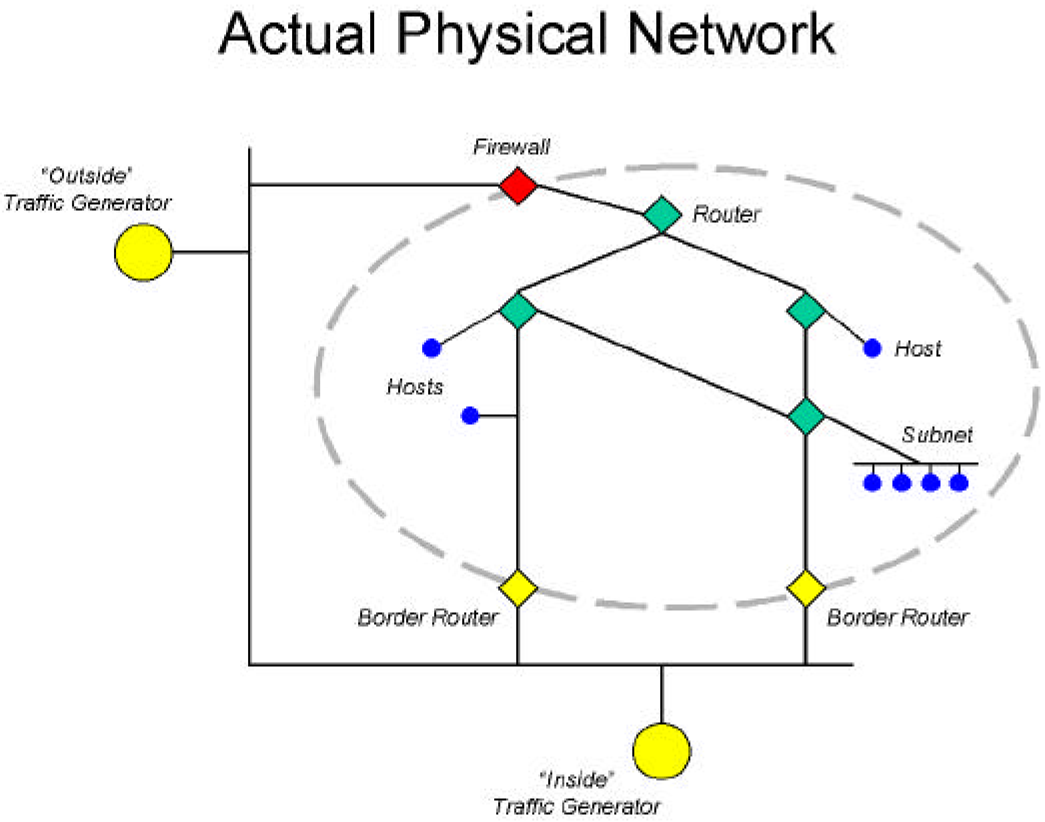

Figure 2. *Actual physical test network architecture.*

For realism’s sake the traffic generators implemented a technique to assign arbitrary source network addresses on a per-process basis. For example, the outside traffic generator could run several simultaneous network sessions (telnet, FTP, etc.), with each session appearing to originate from a unique IP address. AFRL implemented this source-swapping technique by modifying the Linux 2.0 kernel, and the modifications allow assigning arbitrary source addresses to TCP, UDP, ICMP and raw-protocol IP packets. By connecting the traffic generators to the test network through “border” routers, AFRL eliminated the hardware’s hints that the packets were coming from the same two machines instead of from multiple machines inside and outside the network under test. In order to preserve the integrity of the simulated network, AFRL permitted ID systems to be installed anywhere on the physical network except the border routers.

Scripted Traffic Generation and Testing

AFRL developed this “virtual networking” to simulate the presence of many hosts. In addition, Lincoln Laboratory had developed a large library of pre-scripted test sessions and procedures for its non-real-time evaluation, so AFRL made its network a superset of the Lincoln Laboratory network and used Lincoln Laboratory's session-generation tools and data files. The AFRL testbed, running in real time, played a four-hour subset of the Lincoln Laboratory scripted network sessions; this set of data included approximately 2,000 "legitimate" sessions (*i.e.*, nonattack sessions) of various services, or normal data, plus about 30 attack sessions, or anomalous data.

The legitimate sessions, which ran between the two traffic generators to and through the physical test network, included sessions of the following types: SMTP, HTTP, telnet, FTP, ping, IRC, POP, XNTP (time synchronization), DNS, finger and auth. The attack types represented were remote-to-local access, user-to-root access, denial of service and surveillance. The attacks are described in Table 1.

Attack Name or Type	**Description**
Dictionary	Repeatedly tries to guess user's password
Worm	Hops between machines, mailing back the password file to the attacker
FTP put hosts	Puts an **.rhosts** file containing "+ +" in user's directory
HP/UX PPL overrun	Overruns HP/UX PPP daemon to obtain root access
LoadModule	Exploits weakness in SunOS **loadmodule** program to get root access
FTP coredump	Crashes Solaris FTP server to get shadow password file
NMAP	Probes a network looking for hosts and services
WebGAIS	Executes arbitrary commands on the server
IP sweep	Probes a network looking for hosts
Excite/CGI	Exploits CGI vulnerability to execute commands on the web server
PHF	Exploits PHF module vulnerability to execute commands on server
Format	Exploits weakness in SunOS **format** command to get root access
Port scan	Probes a network looking for a few well-known hosts
Eject	Exploits SunOS **eject** vulnerability to get root access
HP/UX PPL coredump	Crashes HP/UX PPP daemon to get root access
Apache	Denial of service attack against Apache server that eventually crashes it (and the

	computer it runs on)
Neptune	Denial of service attack
Land	Denial of service attack
Process table	Denial of service attack using half-open connections

Table 1. *Types and descriptions of attacks run against ID systems.*

1998 Evaluation Results

AFRL's first round, 1998 evaluations of DARPA ID systems used the Lincoln Laboratory-style scripted traffic generation technique. Each ID system under test faced the same four-hour suite of Lincoln Laboratory's network sessions and Lincoln's and AFRL's attacks. All of these scripted sessions were run from a master script that controlled the timing and sequence of each session. Legitimate session scripts served as normal background traffic, and attacks presented a variety of severities of exploits against several operating systems.

Actual evaluation of an ID system's performance was partly objective and partly subjective, depending on both the ID system's raw detection capabilities as well as the manner in which it reported suspicious activity. Some systems rated all activity on a scale (1 to 10 points, or "low-medium-high" risk assessments); some systems only reported attacks in a binary context ("attack seen" *vs.* no reporting). Those systems that reported binary results were required to identify what attack they were catching. Subjectivity was introduced when ID systems false-alarmed: AFRL personnel analyzed why those systems produced false alarms and judged whether they were reasonable decisions given conditions in the test network.

Some attacks manifested themselves as single discrete network sessions and ID systems either caught them or missed them—or often caught "something" but incorrectly identified them. Other attacks such as probes or ping sweeps consisted of hundreds of individual sessions, so if an ID system caught a fraction of these sessions and identified them as a single attack, it counted as a detection instead of hundreds of individual misses. Sessions that were incorrectly identified but still tagged as evidence of an attack were generally scored as positive attack detections.

AFRL evaluated a Government off-the-shelf (GOTS) system as a baseline, three DARPA sponsored ID systems and one commercially available ID system. The DARPA ID systems were chosen for their maturity for a real-time evaluation. One of those systems consisted of several network monitors placed at strategic points around the network, providing 100% coverage of the infrastructure. The other two DARPA systems were host-based systems that monitored isolated computers. The GOTS and the commercial system were placed at the top of the network and monitored all traffic passing between the network and the simulated Internet, as is typically done at military bases.

Receiver operating characteristic (ROC) curves were generated for each system based on its detection and false alarm performance. Each ROC curve shows the false alarm rate incurred by choosing a particular detection rate on a given system. Detection rates for each system were calculated using all of the attacks that were performed against the entire network, not just the attacks for which the detector was designed to trap, in order to measure each system's contribution to the mission of protecting a critical installation.

The baseline GOTS system logged and analyzed all network sessions that passed its sensor and assigned to each session a warning value from 1 to 10 based on keyword matches found in the network packet data. Putatively attacks would get warning values higher than legitimate sessions, but the system often false alarmed or underscored attacks. Human operators would then analyze the entire network log in order to identify bona-fide attacks from false alarms. Generally these GOTS sessions were protecting installations that passed large volumes of network data, so even a 1% false alarm rate would produce too much data for a human operator to examine. As a result, the operators generally inspected only those sessions that were tagged with a high warning value. In the AFRL evaluation, four of the approximately 30 attacks were tagged with warning values of 3.162 or lower. To confirm that these four were indeed attacks, an analyst would have to review transcripts of over 12,000 additional network connections that scored higher than 3.162.

Figure 3 shows the ROCs for the DARPA ID systems compared to the ROC for the GOTS system. False alarm rates are shown only up to 1% because anything greater than that at a large installation would be unmanageable. While one of the DARPA systems' detection performance wasn't significantly better than the GOTS system, all of the DARPA systems showed great improvement over the GOTS system in their false alarm rates.

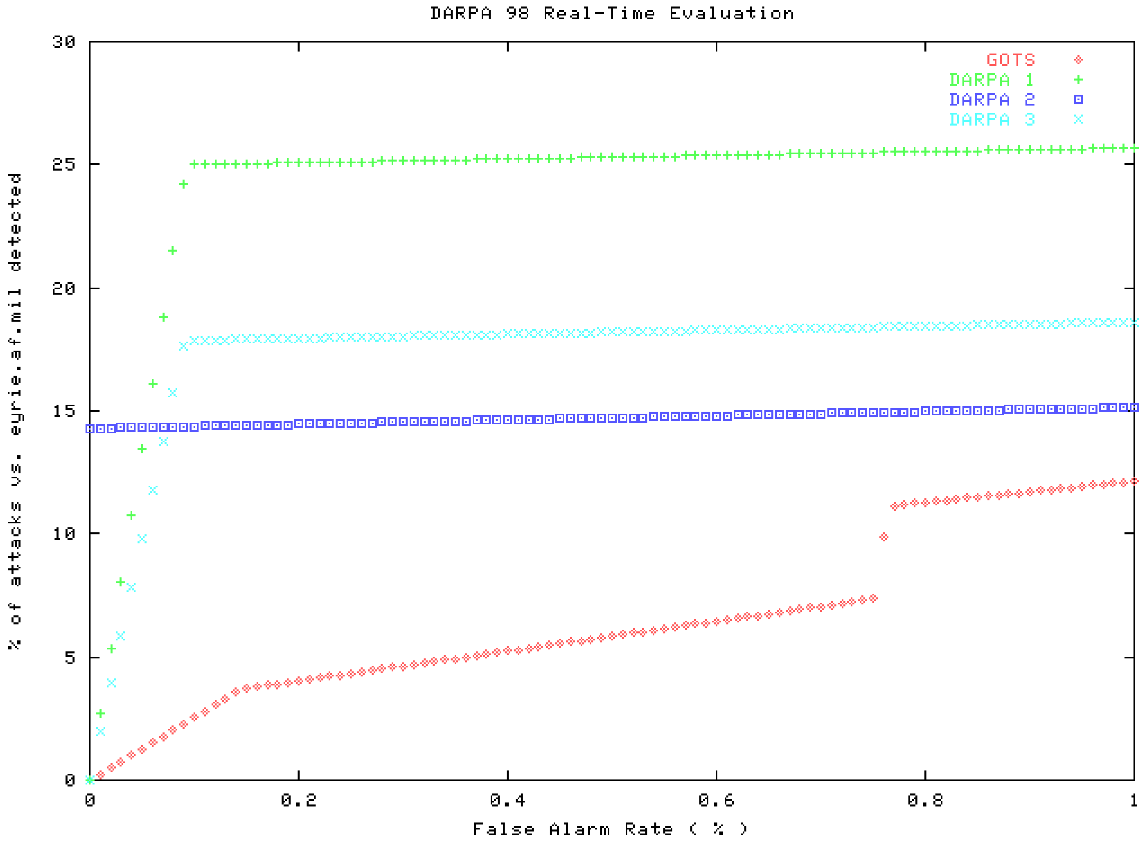

Figure 3. *Receiver operating characteristic (ROC) curves for the GOTS and three DARPA ID systems.*

Immersive ("Live") Traffic Generation and Testing

As mentioned earlier, the normal data in the 1998 real-time evaluation was taken from the Lincoln Laboratory evaluation. Using the Lincoln data offered three benefits: 1) statistical detectors evaluated in our environment would already be trained on the Lincoln data; 2) using the Lincoln scripts assured repeatability; and 3) AFRL could concentrate on the virtual networking without worrying about traffic generation. In practice these advantages became largely illusory—the Lincoln data was not adequate for training systems in the AFRL environment as it was highly punctuated, representing eight hours a day five days a week, and was generated week to week, with the model for generating the data changing each week. Repeatability the strict sense turned out to be an impossible task. AFRL could launch the scheduled session scripts fairly reliably, but the interactions between network components each time were slightly different, and sometimes these interactions would cause the ID systems to alarm. Even porting a tiny four-hour subset of the session scripts from the Lincoln network to the AFRL network took a great deal of effort, although often the more complex topology of the AFRL network introduced errors that did not appear in the Lincoln network.

To evaluate systems that use the statistical properties of network traffic to look for anomalies that might indicate an intrusion or misuse, AFRL needed an environment that could produce enough data to allow such systems to train. AFRL again considered using more training data generated by Lincoln Laboratory, but rejected the idea for the following reasons: 1) a 24-hour environment allowed a richer variety of usage patterns, reflecting peak hours and predictable gross behavior for many users and more unpredictable usage of (say) "power users" or system administrators; 2) too many inconsistencies were present from one week to another in the original training data and these could very likely cause a higher false alarm rate than would occur in the field; and 3) the effort required to port four hours of data from the Lincoln testbed to the AFRL testbed was prohibitive, and AFRL would not be able to generate enough data from the Lincoln experiment to be able to test any system which trained in its own environment.

To address these issues AFRL needed a traffic generation scheme with the following characteristics:

?? Consistent, dynamically reconfigurable statistical properties

?? Unattended round-the-clock operation

?? Rich variability

?? Easily maintainable

?? Easily extensible to new services or attacks

?? Scalable to more than a small number of traffic generators

AFRL decided that the only way to accomplish these goals was to develop a distributed traffic generation technique that could automatically generate sessions. In order to ensure that the system had enough variability made for a more difficult initial design, but the effort required to develop such a traffic generator appeared to be a one-time cost. Once the new distributed traffic generation system was in place it should produce richer data sets with more statistical consistency than could be achieved with sessions generated by hand or on an ad-hoc basis. It also should be able to generate more data for playing into a network under test.

Traffic Generator Architecture

The distributed traffic generation system architecture is shown in Figure 4. There are five layers to the design: the scheduler, the master controller, the slave layer, the automata layer, and the virtual networking layer.

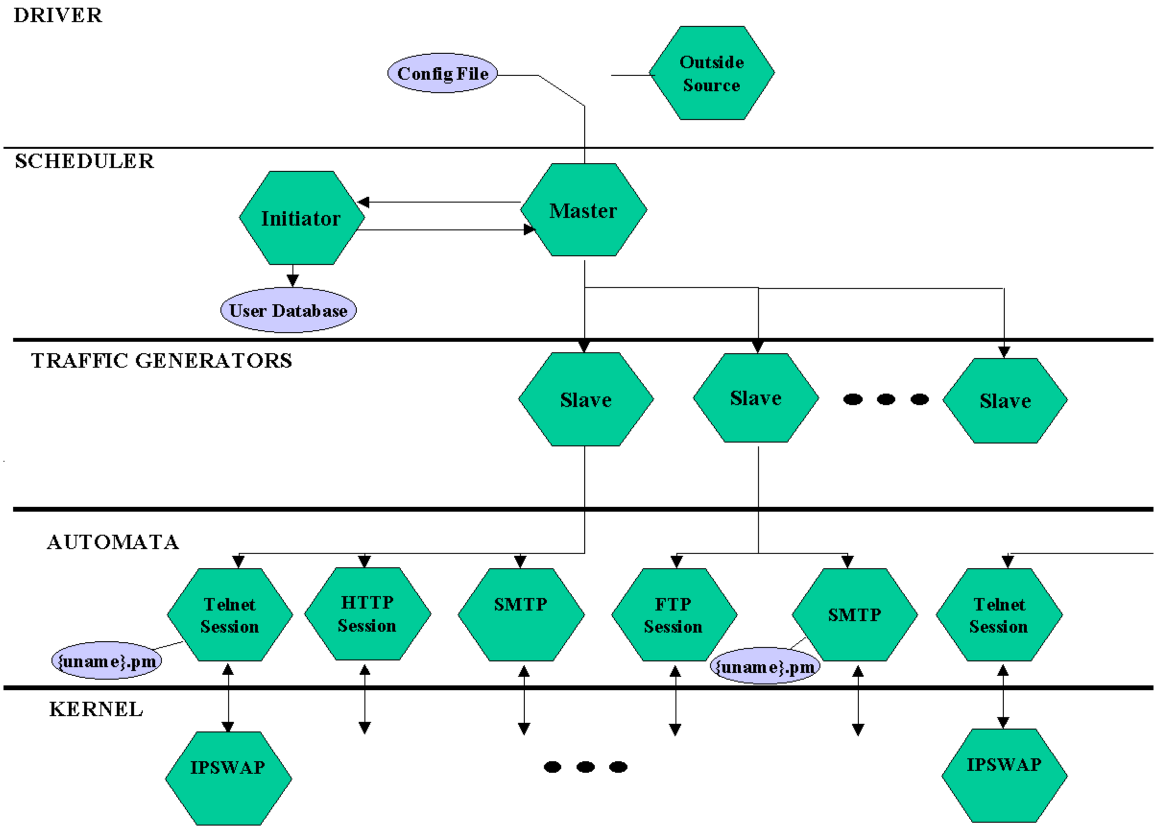

Figure 4. *Architecture for AFRL's full-time traffic generation system.*

The foundation of the traffic generation system is the virtual networking layer. This layer is based on AFRL's previous work developing the virtual networking technology that was used in the 1998 DARPA ID system evaluations. Virtual networking allows individual network-oriented processes to be tagged with arbitrary source IP addresses, simulating the presence of an arbitrary number of hosts.

The automata are actual traffic-generating modules that initiate and conduct network sessions of various services. Automata may interact with each other or with real hosts in the physical network under test. Each automaton implements a single network-oriented process, and runs as if it were a program under the control of a specific user.

The slave layer launches and manages automata as directed by the master controller. Each slave traffic generator is assigned a number of virtual source IP addresses from which automata will run sessions.

The master controller directs slaves to start automata with particular network properties (apparent source IP address, apparent user, duration of activity) and to periodically reactivate or terminate dormant but still-open network connections. The master determines how many sessions to start each time interval by generating a Poisson random variable, and a duration of activity based on the arrival rate and mean number of sessions requested for each service supported. Adjusting these two variables can vary the long-term correlation of the load. Once the master has determined the number of new sessions to start, it accesses a database of virtual users to determine which user would be active given the time of day and the service requested. The database query also returns a source IP address and a password for the virtual user. From this information the master determines which slave direct to start a new session.

The scheduler drives the coarse statistical properties of the simulation. It communicates with the master controller on a regular basis and transmits the properties to be modeled for each service supported. The two parameters it typically transmits for any service are the mean number of sessions that should be active in the current interval, and the arrival rate of new sessions in that interval. The scheduler can get those statistics from a file, from a user interface or can estimate them from a live source such as a feed from a real network.

> *Example.* Traffic Generation Control. *(Scheduler)* From a graphical interface, the test operator selects a small load and arrival rate for SMTP, HTTP and FTP sessions. These parameters are passed to the master controller. *(Master)* The master generates a real Poisson number of sessions to make active based on the parameters passed to it by the Scheduler. The master checks if any ongoing sessions satisfy the Scheduler's request; if so, it conditionally activates them. If not, it generates source information (user, password, source IP address) for each session. After determining what actions to transmit to the slaves for session management, the master contacts each slave in turn and updates that slave with only the information pertinent to it. *(Slaves)* Each slave accepts directives from the master controller and either initiates new sessions of the appropriate service/user/source, or reactivates or terminates open but dormant sessions. Each session runs as a slave's automaton process, and the slave starts or contacts the appropriate automata to effect the master's instructions. *(Automata)* New automata start new virtual networking sessions and start generating traffic; reactivated automata continue generating

traffic; and terminated automata simply close their connections and die themselves.

Advantages of Immersive Environment

Besides being statistically consistent and capable of full-time unattended operation, using a dynamic immersive environment instead of prescripted sessions has other advantages. It will be much easier to bring new services into the testbed as automata as they are developed, instead of having to reconstruct an entire suite of scripts that incorporate all known services. Other network activity may either be coded as automata, run as scripts, or be conducted by a live human. Because it will be distributed, much more traffic can be generated without reaching physical limits of the traffic generators' hardware (such as the number of slots available to hold network interfaces).

It will also be easier to port and maintain this distributed system. For one, the traffic will be generated on-the-fly by automata, so very little mirroring of data files will have to be done between the traffic generators. In addition, traffic-generation slaves may be added or removed from ongoing tests without interrupting the flow of traffic.

1999 Test Method

The procedure for this year's evaluation has been designed with two goals in mind: obtaining reliable estimates for detection/false alarm rates; and handling anomalous, non-repeatable events that occur while the test is under way. To achieve these two goals we subdivide the test into equal blocks of four hours. We randomize the exploits among the blocks such that no attack appears in a block more than once and every attack occurs multiple times.

As each attack will occur with a different set of background data we will be able to account for environmental factors that could affect the performance of individual ID systems. In addition, by averaging over blocks which have similar traffic loads we will be able to come up with reasonable estimates of false alarm rates.

By dividing the test into blocks we also hope to be able to isolate anomalies that occur while we are executing the test. Such things as disk crashes and disabled hosts can be isolated in time and the test blocks associated with the anomaly can be thrown out.

1999 Test and Evaluation Schedule

The immersive test environment will be used both to evaluate new DARPA ID systems, to select candidate components ready for integration into the Information Assurance Automated Intrusion Detection Environment (IA:AIDE, an advanced concept technology demonstration being developed by the Defense Department), and to help AFRL recommend which computer security products are suitable for fielding nationwide in military installations.

AFRL will begin immersive testing systems for the IA:AIDE in the late summer to early fall of 1999, and will conduct a real-time evaluation of DARPA ID systems in the late fall of 1999.

Acknowledgements

We thank the University of California at Santa Barbara for their help analyzing and debugging the connectivity of the AFRL test network. Without their help the 1998 test would not have succeeded.

We also thank all parties who provided ID systems for evaluation in the 1998 test, specifically:

Participant and System	URL
UCSB's "NetSTAT" and "USTAT"	www.cs.ucsb.edu/~kemm/NetSTAT
Stanford Research Institute's "EMERALD"	www.sdl.sri.com/emerald/index.html
MCNC and North Carolina State University's "Scalable Intrusion Detection for the Emerging Network"	www.anr.mcnc.org/ JiNao.html
Oregon Research Institute's "StackGuard"	www.cse.ogi.edu/DISC/projects/immunix/StackGuard

Terrence G. Champion

Terrence Champion works as a research scientist at the Air Force Research Laboratory's INFOSEC Technology Office (Hanscom AFB MA). For the past three years Mr. Champion has been involved in the evaluation of intrusion detection systems for emerging military requirements. Mr. Champion also designed and developed the AFRL ID evaluation testbed and associated software. Mr. Champion has recently co-founded Skaion Corporation (Arlington MA) to provide network security test and evaluation products and services.

Robert S. Durst

Robert Durst, formerly a computer security program manager in the Air Force, currently works as a research scientist for SenCom Corporation (Bedford MA) supporting the Air Force Research Laboratory INFOSEC Technology Office (Hanscom AFB MA). Mr. Durst helped develop the AFRL ID evaluation testbed and associated software. Mr. Durst has recently co-founded Skaion Corporation (Arlington MA) to provide network security test and evaluation products and services.