Evaluating Intrusion Detection Systems without Attacking your Friends: The 1998 DARPA Intrusion Detection Evaluation

R. K. Cunningham, R. P. Lippmann, D. J. Fried, S. L. Garfinkel, I. Graf, K. R. Kendall, S. E. Webster, D. Wyschogrod, M. A. Zissman

Abstract

Intrusion detection systems monitor the use of computers and the network over which they communicate, searching for unauthorized use, anomalous behavior, and attempts to deny users, machines or portions of the network access to services. Potential users of such systems need information that is rarely found in marketing literature, including how well a given system finds intruders and how much work is required to use and maintain that system in a fully functioning network with significant daily traffic. Researchers and developers can specify which prototypical attacks can be found by their systems, but without access to the normal traffic generated by day-to-day work, they can not describe how well their systems detect real attacks while passing background traffic and avoiding false alarms. This information is critical: every declared intrusion requires time to review, regardless of whether it is a correct detection for which a real intrusion occurred, or whether it is merely a false alarm.

To meet the needs of researchers, developers and ultimately system administrators we have developed the first objective, repeatable, and realistic measurement of intrusion detection system performance. Network traffic on an Air Force base was measured, characterized and subsequently simulated on an isolated network on which a few computers were used to simulate thousands of different Unix systems and hundreds of different users during periods of normal network traffic. Simulated attackers mapped the network, issued denial of service attacks, illegally gained access to systems, and obtained super-user privileges. Attack types ranged from old, well-known attacks, to new, stealthy attacks. Seven weeks of training data and two weeks of testing data were generated, filling more than 30 CD-ROMs. Methods and results from the 1998 DARPA intrusion detection evaluation will be highlighted, and preliminary plans for the 1999 evaluation will be presented

Introduction

To date, intrusion detection research has lacked a corpus of data that could serve as the basis for system development, improvement and evaluation. To meet that need, we have developed a corpus of data for the DARPA 1998 off-line intrusion detection evaluation, using a network and scripted actors to loosely model the network traffic measured between a US Air Force base and the internet. This year's data includes normal network traffic for many different Unix services and protocols, while future years will address networks populated with a mix of Unix and Windows/NT machines. Although the evaluation is primarily intended to improve the state of intrusion detection research, the unique combination of normal data, attack data, and labeled sessions has also prompted evaluation of fielded commercial and U.S. government systems.

Methodology of Formal Evaluations

To evaluate formally the utility of an intrusion detection system, one must know both the probability of detecting an attack (P_d) and the probability of issuing a false alarm (P_{fa}). Most current intrusion detection systems report a variety of warning levels and allow system administrators to adjust an alert threshold that effectively trades off P_d for P_{fa}. By lowering the threshold, an administrator can discover more attacks, but will likely have to see more false alarms. Similarly, an administrator can raise the threshold to reduce false alarms, but this will also likely cause the system to miss some additional attacks. By knowing these two values (P_d and P_{fa}) and how they change with changes in the threshold, one can plot a receiver operating characteristic (ROC) curve [1]. Knowledge of the entire ROC curve allows the user to set thresholds to match the amount of available effort and desired level of security.

Most systems also provide some degree of local configuration to allow experts to customize the system to a given environment. To avoid learning how to run and customize each intrusion detection system, and to reduce the time required to perform the evaluation, we elected to perform an off-line evaluation of all systems. In this type of evaluation, two sets of data are provided to the participants. The first set is training data, which contains sessions and an indication of whether the session is a normal or attack session. Expert users or system developers configure their systems to achieve the highest detection rates and the lowest false alarm rates. In the second data set, only the network sessions are provided, and it is the job of the participant to declare whether the session is a normal or attack session. These declarations are returned and scored by the MIT Lincoln Laboratory team.

A second on-line evaluation is being performed in tandem with our effort. In that work, a subset of systems we are evaluating will again be tuned using our training data. The systems will be tested in real-time using tools and techniques the two research groups have cooperatively developed.

Creating Data that can be Widely Distributed

Three different approaches were considered to create the data used in this evaluation: collect operational Air Force network traffic, occasionally attacking a few acceptable machines; sanitize operational traffic, inserting attack sessions after the fact; and synthesize both normal and attack sessions on a private network.

The first option was quickly rejected. Although data collected from a functioning network would be (by definition) real data, no director of an operational base or business wants to have his infrastructure attacked, even to establish how well various systems can defend against that attack. Furthermore, there is substantial personal, private and sensitive information such as e-mail, web access patterns, passwords, and information about the network configuration that could be gleaned from the distributed data. This information could expose vulnerabilities in the real network. Finally, it is difficult to be certain that the collected traffic, which we would label as normal, does not include attacks from outside crackers.

To avoid attacking the operational network, and to safeguard sensitive information, we next considered sanitizing real data and inserting attacks into the traffic after the fact. Although it seems reasonable to change passwords, substitute e-mail, change web addresses, and modify host IP addresses in real collected data, in practice it is extremely difficult to do this reliably. Furthermore, the amount of traffic recorded implied that this sanitization process would be automated with only occasional spot checking, so the likelihood of accidentally releasing sensitive information seemed significant. Finally, inserting attacks into operational data might introduce artifacts. This, combined with the realization that some unknown attacks might be present in the collected data led us to investigate a third option.

To safeguard sensitive information, to prevent the introduction of artifacts, and to ensure reliable labeling of sessions, the network traffic of an entire Air Force base was simulated. To establish which services are in use and the relative traffic levels, we began by installing a network traffic sniffer on a local Air Force base, and recorded the amounts and types of services used. That effort revealed that traffic included SNMP, SMTP, telnet, FTP, HTTP, POP, domain, time and many other services and protocols. This traffic is consistent with a business where personnel send mail messages, transfer files, and access databases and information servers.

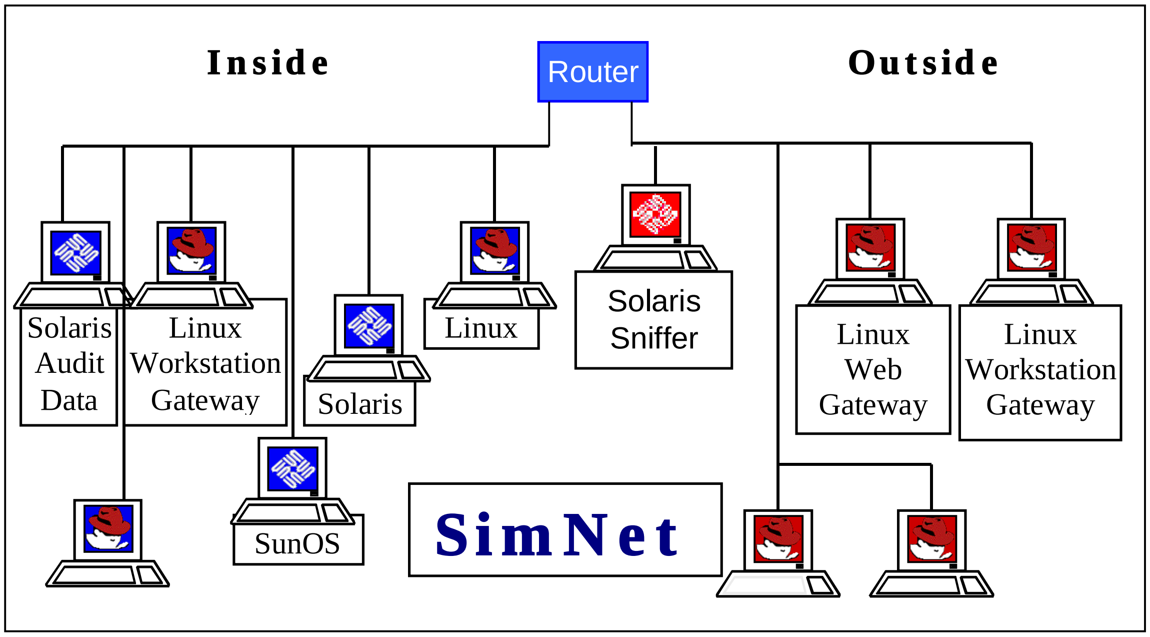

Figure 1. The Simulation Network. Inside computers simulate an Air Force base, while outside computers simulate the internet. Gateway computers simulate many hundreds of workstations. Named computers have set roles and fixed ip addresses, while unnamed computers are set to different ip addresses as the simulation proceeds.

A local, isolated network was established and split into two with a router in between. (See Figure 1.) The inside simulated a base consisting of tens of machines and many hundreds of users, and the outside simulated the thousands of machines and the many users who make up the internet. To simulate many machines, a modification was made to the LINUX kernel [2]. To populate the base, classes of normal users and their work were identified: some people were programmers, others were secretaries and system administrators, and others were managers. Scripted actors were developed to simulate the work of each of these: the manager read and responded to mail, the secretary wrote and formatted memos, the programmer wrote, compiled and tested programs, and the system administrator installed software, responded to questions, and repaired the systems. In addition, real people sent requests to a live system administrator who responded and repaired damage caused by scripted and real attackers.

Attack scenarios were developed for different attackers. One was a spy who collected information and left a back door; another was a novice hacker who broke in and then left, and a third was a malicious employee. Attacks included surveillance/probing attacks, denial-of-service attacks, remote-to-root attacks, and user-to-root attacks. Most were old attacks, but some were newly developed for this exercise. The attacks targeted SunOS systems, Solaris systems, Linux systems and the Cisco router, and varying effort was used to hide the attacks, consistent with the level of expertise of the attackers. Some of these attacks were scripted, although many required human intervention.

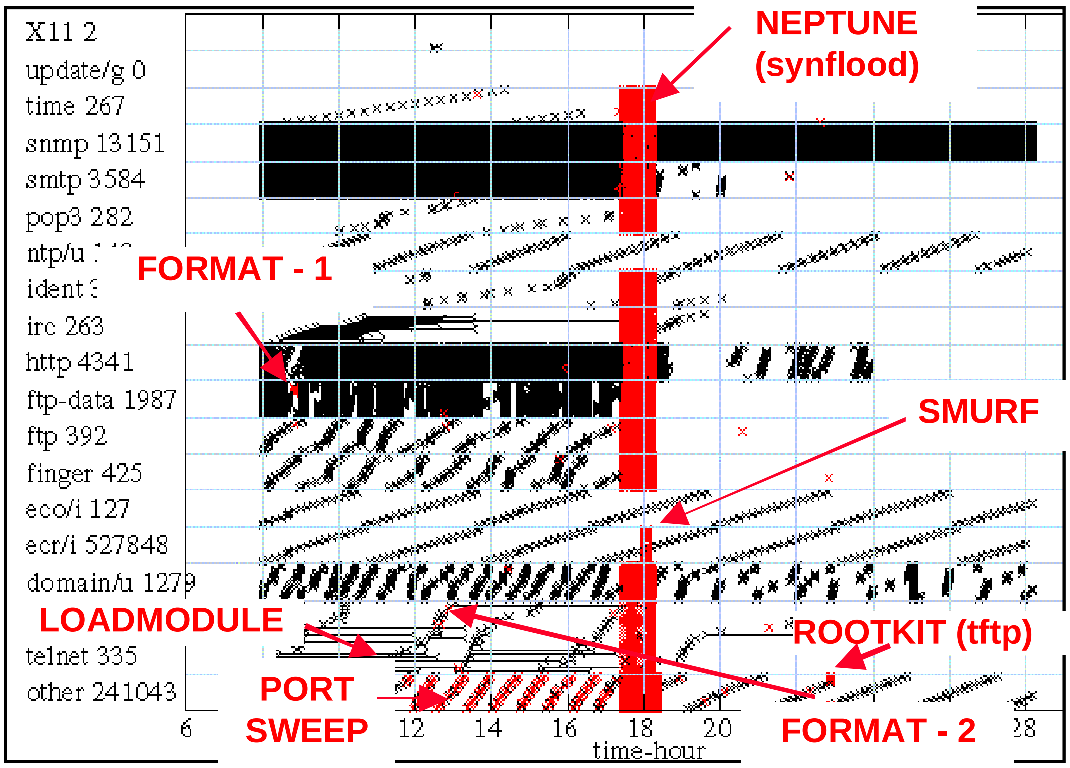

Figure 2. Traces of connections occurring during one simulation day. In this figure each service is assigned a vertical block in which sessions are indicated with the greater-than sign (>) connected via a line to a less-than sign (<). Each session is plotted in the block with a slight vertical offset from the previous one. More heavily used services appear as dark, vertical bands, while less heavily used services appear as diagonal bands. Several different attacks occurred on the depicted day, and are indicated in the figure's overlay.

Sample Data

Six weeks of *training data* and two weeks of *test data* was provided to each participant. Distributed data contained system dumps, periodic output from the Unix ps command, audit data from the Solaris Basic Security Module, and sniffed network traffic. For the training data, participants received list files to indicate whether a session was a normal or attack session, and if an attack, the type of attack performed. After participants trained their systems, two additional weeks of data was distributed. For the test data, the participants' systems labeled each session as a normal or an attack session.

An example of the generated traffic is displayed in Figure 2. During each simulated 22-hour day, traffic for a variety of services and protocols is generated by scripted and human actors. Traffic is heaviest during the day, when bases are the busiest. During the day the actors scan the world-wide web, (http), send mail (smtp/pop), transfer data (ftp), enquire about other users (finger), log into information servers (telnet) and have their computer systems stay synchronized to a master clock (time). The day shown in Figure 2 has several examples of attacks: a network mapping attack (Port Sweep), two denial-of-service attacks (Neptune, Smurf), and two user-to-root attacks (format, rootkit) [3]. In the case of the format attack, the attack began in one session and ended in a later session using a different service.

Conclusions

We have succeeded in simulating a large operational network and automated the production of both normal and attack traffic. Evaluations of intrusion detection systems must be done in the presence of both to establish the performance of an intrusion detection system. Without normal traffic the total number of false alarms generated by the system cannot be known, and without this knowledge the cost of maintaining the system is not known.

Initial evaluation of a simple keystring-spotting system showed poor results. Although this system can detect all attacks executed in the clear, it has an extremely high false alarm rate. Those who use this technology, currently deployed in many commercial intrusion detection systems, will spend substantial time evaluating sessions for which an attack did not occur.

Ten systems are currently being evaluated, and preliminary results indicate that these next-generation systems are already significantly better than the simple keystring-spotting system. We anticipate that this improvement will continue as we produce new and more complex data sets in the years ahead.

References

1. Swets, J.A., *The Relative Operating Characteristic in Psychology*. Science, 1973. **182**: p. 990-999.

2. Durst, R.S. and Champion, T. G. Packet Address Swapping for Network Simulation, US Patent Application, 1998.

3. CERT, *Computer Emergency Response Team Advisories*, 1998, www.cert.org.