

MAGNA: Modeling And Generating Network Attacks

William H. Allen, Gerald A. Marin
Department of Computer Sciences
Florida Institute of Technology
Melbourne FL, 32901

July 30, 2004

Abstract

We present a system that generates synthetic attack traffic from state-based models of the behavior of real network attacks. The execution of these attacks can be carefully controlled to produce realistic training data that could prove useful in the evaluation and development of intrusion detection systems. This tool can also be used to test host systems and networks for known vulnerabilities by launching controlled attacks and observing their impact on the target.

Keywords: Network-level security, network monitoring, intrusion detection.

Corresponding author: William Allen, wallen@cs.fit.edu

MAGNA: Modeling And Generating Network Attacks

1 Introduction

The number of security incidents reported in 2003 by the CERT Coordination Center at Carnegie Mellon University was greater than the totals reported for 2001 and 2002 combined [1]. The economic impact of computer attacks and viruses has been estimated to exceed \$12.5 Billion a year [2]. Those facts, as well as recent events such as the rapid spread of the Slammer (January 2003) and Blaster (August 2003) worms, show that there is a vital need for software and systems that can detect and identify network attacks as they happen.

To meet this need, both academic researchers and commercial software developers have been designing, implementing and refining intrusion detection (ID) algorithms and systems since the mid-1980's. However, recent evaluations of intrusion detection systems have shown that even the most current systems produce large numbers of false alarms, lack effective user-interfaces and fail to detect many previously-unseen attacks. A further problem is the need for realistic training data (both with and without attacks) used to develop and test new detection algorithms.

As with all scientific experiments, the designers of intrusion detection systems must test their detection algorithms under controlled conditions to determine their ability to detect specific attacks. However, traffic collected from real networks may contain any number of unknown attacks, and exposing a detector to that traffic provides anything but 'controlled conditions'. Clearly, researchers and developers must be able to control the introduction of attacks into an isolated test-bed network so that they may evaluate the detection capabilities of their algorithms.

In this paper, we present a system that generates synthetic attack traffic based on models of real attacks. The execution of these attacks can be carefully controlled to produce realistic training data that could prove useful in the evaluation and development of intrusion detection systems. This tool can also be used to test host systems and networks for known vulnerabilities by launching controlled attacks and observing their impact on the target.

2 A System for Modeling And Generating Network Attacks (MAGNA)

It is generally accepted that the most realistic way to create synthetic network traffic is to measure the traffic at one or more sites and produce a model of that traffic's behavior which can be used to generate new traffic [3]. However, when network attacks are added to that synthetic traffic, they are either the result of executing a copy of attack code on a host in the network [4] or of transmitting random packets with a distribution similar to that of a known attack [5]. The former method requires access to the attack's code and the latter does not take into account any attack characteristics other than the attack's overall distribution.

Attacks are often first detected in live networks due to their negative impact on the network or because of some abnormal aspect of their behavior [6]. The source code for some attacks is never revealed or is only available after considerable delay. Thus, researchers are unable to test their ID system's ability to detect those particular attacks. A further problem is that attacks are often written to execute on a specific platform or in a particular language and some researchers may not have those resources available, even if they do have access to the attack's source code. They are then forced to write a similar program in a language and for a platform that is available to them. Some key features of the attack's behavior may be lost or altered in this process.

In this paper, we present a new method for generating synthetic attacks that accurately match the behavior of actual attacks that we call MAGNA (Modeling And Generating Network Attacks). The MAGNA system does not execute an attack's source code; in fact, a copy of that code is not even necessary to create

a synthetic version of the attack. Instead, attack generation is based on a detailed model of the original attack’s behavior, produced from packets captured during the attack’s execution in a live network. This attack model, along with selected characteristics of the attack, is used to recreate the attack’s behavior in a controlled environment. As the model is refined, the synthetic attack can be ‘tuned’ to provide the same test of an ID system’s detection ability as the original attack. This new method can be combined with a traffic generation scheme, such as that used for the Lincoln Lab evaluations [4], which produces non-attack traffic from models of normal network traffic.

The term *characteristics* refers to the contents of each unique packet included in the attack. This includes all of the packet header or payload information that is needed to accurately recreate the attack packets. This information is stored as records in a database, one record per packet. To allow the construction of variations on a modeled attack, the information used in recreating that attack’s packets could be taken directly from the captured packets used to model the attack, or may come from alternate sources, such as user input or randomly-generated data.

The term *behavior* is intended to describe an attack’s interactions with the target, the type and number of packets it transmits, the timing of those transmissions and any other distinguishable actions that are required for the attack to be accurately recreated. A state-based model of attack behavior is used where each state is associated with a single step in the attack’s execution.

The goal of this technique is to generate attacks that are realistic in their behavior, not just exact duplicates of sampled attacks. When multiple instances of a specific attack are available, they are all used to enhance the model of that attack. The captured attacks are analyzed to determine the range of variations in each packet header field and the number of times each variation occurred. From that information, the probability of occurrence of each variation is calculated and that probability is used to generate packets according to the distribution of variations that was observed in the captured attacks. For example, assume that four instances of a particular attack were captured and it was seen that in two cases a certain header field contained one value in all packets and in the other two instances a second value was used in all packets. The model would be constructed so that there would be a 50% chance that the execution of that attack would use the first value and a 50% chance that it would use the second value.

2.1 Evaluating Intrusion Detection Systems

Several well-known research projects [7, 8, 4, 9] have produced test-bed environments for the evaluation of ID systems. The attacks generated in those systems came from attack code that was acquired from outside sources or was written by the research team, based on descriptions of known attacks. Recent research in IDS evaluation shows that attacks are still commonly generated by executing code [3] or by the use of a combination of tools that each produce specific types of traffic, such as fragmented packets, or that generate packets at a high rate to ‘stress test’ ID systems [10].

The developers of Intrusion Detection systems use a variety of techniques to model user activities or traffic behavior to detect intrusions. Researchers have adapted modeling and learning techniques from a range of disciplines to create novel detection methods. Those projects include ideas from neural networks, state machines, colored petri nets, statistical analysis and immunology. However, this research does not focus on modeling attack behavior for the purpose of attack generation.

3 The MAGNA System Design

The MAGNA system consists of four modular components (shown in Figure 1). We begin with a brief overview of these components and then discuss each in more detail.

- *attack modeling component* - creates a model of the attack by using a combination of captured network traffic that contains an instance of the attack and configuration options specified by the user
- *model storage component* - stores the modeled attack in a set of databases so that the attack can be recreated as needed
- *attack editing component* - allows researchers to create variations on the original model for each attack to further test their detection algorithms

- *attack generation component* - executes the selected attack by building attack packets from the stored model and injecting the attack into a live network

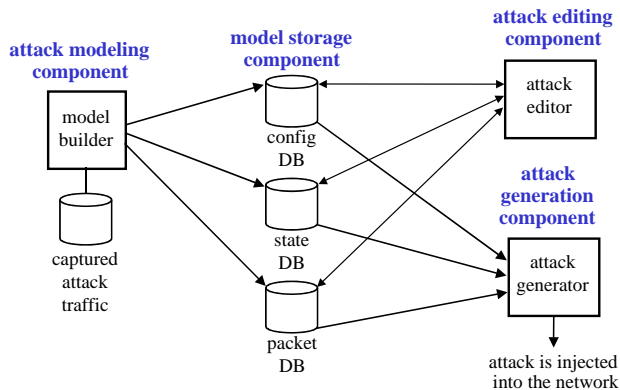


Figure 1: A diagram of the MAGNA System

Attack modeling in MAGNA is currently a manual process in which the human analyst creates a set of files that are used as input to the attack databases. The attack storage component is a command-line program which reads these attack model files and builds the appropriate database tables. We are currently researching the topic of automated attack modeling with the goal of developing a modeling component that would generate a model of an attack, support manual refinement of the model and then create the appropriate database entries to store the completed model for later use.

The attack editing component employs a Java GUI that interfaces with the attack model databases to modify attack models or produce variations on previously modeled attacks. The attack generation component was deliberately separated from the other system components and was designed as a command-line program so that it can be easily controlled by scripts or remotely executed via a telnet or ssh connection to the machine on which it executes.

In addition to producing data for the evaluation of Intrusion Detection systems, MAGNA could be used to directly test network devices and host machines for a wide range of vulnerabilities. The system to be tested could be installed in the testbed network and a number of modeled attacks could be launched against it while both the network and the system under test were monitored closely. By use of a carefully designed suite of attacks and configuration tests, analysts could evaluate the system’s resistance to attack and possibly detect previously unknown vulnerabilities or instabilities.

3.1 The Attack Modeling Component

An attack’s behavior is represented by a state diagram which describes the steps needed to accurately reproduce the attack. That diagram is converted into a state transition table that is stored in the attack state database. The state transitions are each associated with some action, which is represented by a *function* that recreates that action at execution time. Each function is designed to be *atomic* in nature and executes a simple step within the (possibly) complex chain of events that constitutes an attack. The system is designed so that the existing set of functions can easily be extended to handle attacks with behaviors that can’t currently be modeled. Each function can accept a set of input parameters and will either produce an external event, such as the transmission of a packet, or will change the system’s state by storing or modifying the value in a specific state variable. These state-based models are read from the attack databases and executed on a virtual machine which uses a global system state to store information about the attack’s progress as it proceeds through the state diagram.

3.1.1 Example: the *neptune* attack

As an example, we will discuss the construction of the model for the *neptune* or *SYN-flood* attack. This attack sends a series of TCP SYN packets to a specific port on the target host, each requesting that a TCP connection be made to that port, but does not complete the normal TCP handshaking sequence to fully establish the connection. Incomplete TCP connections are held in a fixed-length queue, but can be removed after a time-out period if the connection is not completed. If this queue fills, no further connection requests will be accepted at that port and even legitimate requests will be blocked. Once the *neptune* attack sends enough requests to fill the queue, it can maintain the denial-of-service by transmitting additional requests often enough to replace expired requests.

Figure 2 shows the state diagram for the *neptune* attack. In the example shown here, states 3, 4 and 5 form a loop that transmits a series of N packets until the attack ends. Once completed, the *neptune* attack's state diagram would be used to create a state transition table which was written into the attack state database.

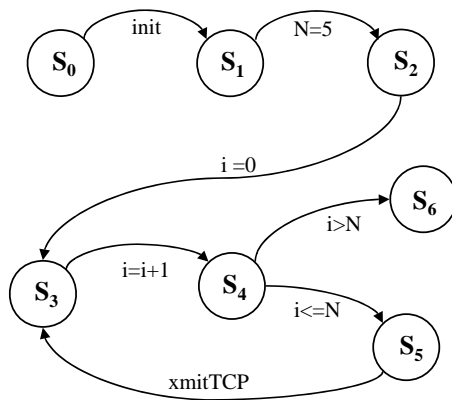


Figure 2: The state diagram for the *neptune* attack with its associated state transitions. When the last state (S_6) is reached, attack execution ends.

The *neptune* attack could be modeled in at least three different ways. Although it sends many packets, all of them are similar and a simple model could just repeat the same packet many times. However, this would mean that all of the attack's packets would use the same source port number, same TCP sequence number and same IP identification number. An attack generated that way would not be difficult to detect because many ID systems can easily recognize a series of identical packets as an attack. We chose to model this attack by using a series of packet headers that were taken from a captured *neptune* attack and which contained realistically-changing values in those fields. A more complex model could generate random values for many of the packet header fields, taking care to use only valid values for those fields. This would obscure the fact that all of the packets came from the same source and make it appear that the target was simply receiving a high volume of normal traffic.

3.2 The Model Storage Component

There are three databases used for storage of the attack models. The *attack configuration* database stores a list of all modeled attacks and also contains specific information about each attack. The *attack state* database contains the state transition tables that describe the behavior of the attacks. The *attack packet* database holds the packet header and data fields that are used to create packets for the modeled attacks.

3.3 The Attack Editing Component

One of the goals of the MAGNA system is to allow users to create modified versions of known attacks to test whether their detection algorithms will detect variations from the expected attack behaviors. Rather than

allow the user to modify an existing attack, the system creates a copy of that attack and provides support for altering the copy so that its behavior or characteristics can be changed in measurable, but carefully controlled ways. The original attack remains intact so that it may be used to build new variants or to compare the detection system’s responses to both the original attack and its derivatives. Once built, models are marked as read-only so that they may not be altered by accident.

The attack editing component is implemented as a Java program (see Fig. 3) that interfaces with the MySQL server where the attack databases are stored. The contents of the attack configuration, state and packet databases are displayed in separate panels. Variations on modeled attacks can also be created and stored in the databases through this interface.

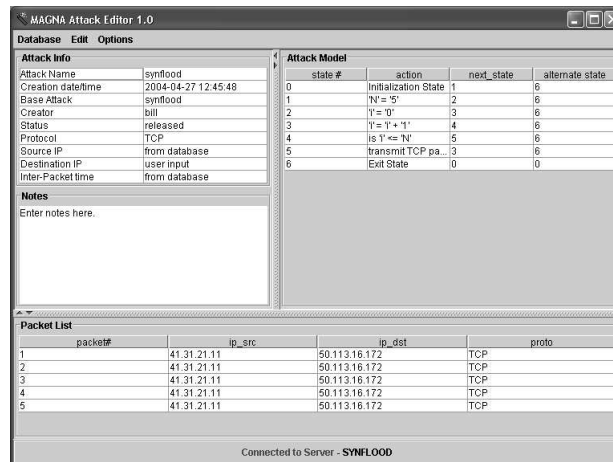


Figure 3: The MAGNA attack editor.

The attack editor could also be used to manually create new attacks to test newly-discovered vulnerabilities for which attacks have not yet been written or to evaluate the security of new network protocols or devices. Test suites could be assembled that combine attacks with non-attack tests and they could be executed via scripts to exhaustively test new or existing systems.

3.4 The Attack Generation Component

The attack generator is written in the Python language to take advantage of its network and database support and its portability. The latter feature allows users to install the attack generator on a wide range of platforms so that test-bed networks can be constructed easily. Note that the database server where the attack databases are stored does not have to be a part of the test-bed network, as long as the attack generator can connect with it (perhaps via a second network interface) to access the attack models. The attack modeling and editing components can be used by anyone with access to the attack databases. However, attack generation may require ‘root’ or ‘administrator’ permissions if it injects packets into the network via the link-level interface,

Although the model building and attack editing components both require user interaction to select and enter configuration and attack options, the attack generation component was designed so that it could be executed from a command-line interface without interaction. This allows the attack generation program to be called from scripts or executed at a preset time by a scheduling program such as *cron*. Any user input that might be needed at runtime to customize the execution of an attack, such as the target’s IP address, can be entered via command-line arguments (e.g., `-d destination-IP-address`).

4 Testing the Attack Generator

We tested the MAGNA system by modeling a set of known attacks (taken from the 1999 Lincoln Lab data) and comparing the packets generated by each attack with the packets extracted from the Lincoln Lab attack

data. In this section, we describe the experiments used to test the attack generator and show the results of those tests.

4.1 Experimental Design

The ultimate goal of this project is not to recreate attacks exactly as they appeared in the Lincoln Lab data; it is to produce realistic attacks from a model based on captured attacks. However, in our initial experiments with the MAGNA system, our goal was to demonstrate that MAGNA can accurately reproduce captured attacks from a model of the attack’s original behavior. As described in Section 2, the modeling process can be enhanced by capturing variations in the execution of attacks and determining the probability of each variation, so that those probabilities can be used to create more realistic attacks. However, that technique was not used for the experiments described here.

To test the system’s ability to accurately recreate known attacks, a range of attacks from the 1999 MIT Lincoln Lab ID evaluation were modeled and those models were used to generate attacks in a controlled setting. The synthetic attacks were captured so that they could be compared with the original attack data to determine if the models were accurate. A description of these attacks is provided in [11]. This set of modeled attacks employs a range of IP-level protocols (ICMP, IP, TCP, UDP) that are currently supported by MAGNA.

The attacks that were modeled for these experiments included:

- the *land* attack, which uses a single malformed packet to cause a denial of service
- the *teardrop* attack, which employs invalid packet fragmentation to attack a single host
- the *neptune* (or *SYN-flood*) attack, a denial-of-service attack which is directed at a single host
- the *smurf* attack, which uses ICMP echo request/reply packets to either target a single host or flood an entire LAN
- the *udpstorm* attack, which uses the *chargen* and *echo* services to produce a flood of UDP packets in a network
- the *ipsweep* surveillance scan

The experiments were conducted in an isolated network which contained several ‘target’ hosts running the Microsoft Windows, Linux and Solaris operating systems. However, the operating system version installed on each host was more recent than the Lincoln Lab evaluation and most of the vulnerabilities that were exploited in the Lincoln Lab attacks had been corrected. Thus, the target hosts were often not affected by many of the modeled attacks. Therefore, success in generating an attack was determined by comparing the synthetic attacks generated in the test-bed network with the original attacks captured from the Lincoln Lab data and by comparing the generated attack’s behavior to the detailed descriptions of the original attack’s behavior in [11], [6] and [4]. While the experiments were running, traffic was captured by ‘analyzer’ machines which were running the *tcpdump* packet capture software. The packets captured from each analyzer were compared to determine if the generated packets were a) correctly injected into the network and b) behaved like the original attacks in the Lincoln Lab data. Response traffic from the target hosts was also captured and analyzed to aid in further determining the accuracy of the generated attacks.

4.2 Experimental Results

In this section, we will discuss the results from our experiments and show that the attack generation process successfully recreated the behavior of 83% of the modeled attacks taken from the 1999 Lincoln Lab experiments.

The results from our preliminary experiments are shown in Table 1. In 5 of the 6 cases, the attacks were recreated successfully and the attack packets matched the original packets in the Lincoln Lab data. The *teardrop* attack could not be generated by MAGNA because the operating system it was running on (Solaris 5.8) did not allow MAGNA to transmit packets with invalid fragmentation offsets (the exploit *teardrop* is based upon). In practice, this attack is seldom effective because the vulnerable operating systems have been

Table 1: Attack generation results.

Attack Name	Successful?
<i>land</i>	yes
<i>teardrop</i>	no
<i>neptune</i>	yes
<i>smurf</i>	yes
<i>udpstorm</i>	yes
<i>ipsweep</i>	yes

replaced or patched to prevent its use. However, if it were necessary to recreate this attack, the attack generator could be installed on an older, unpatched operating system that allows the creation of packets with invalid fragmentation offsets. The *udpstorm* attack packets were generated successfully, but the attack did not succeed because the (updated) target machines did not respond as intended by the designers of *udpstorm* and the expected flood of UDP packets did not occur. This attack should still work against an older, unpatched version of Windows or Linux.

Other than the destination IP address and timestamp, the synthetically-generated attack packets are identical to the packets from the Lincoln Lab data. However, using the attack editor we could alter certain configuration options to create variations on the original Lincoln Lab attacks that might make them less detectable by an ID system. For example, the *neptune* attack’s characteristics could be changed so that the ‘spoofed’ source IP address (which was always set to 11.21.31.41 in the Lincoln Lab data) could be randomly generated to avoid detection by an IDS that counts how many ‘half-open’ TCP connections come from one particular source address.

Variations like this can be used to determine if an IDS that is based on signature detection is using rules that are too restrictive and is only detecting *neptune* attacks that look exactly like the one found in the Lincoln Lab data. While that might seem unlikely, consider the fact that the ‘spoofed’ source IP address used for each instance of the *neptune* attack in the Lincoln Lab data was the one shown above. If the designers of an IDS developed their detection signatures from the same attack source code that the Lincoln Lab team used, the signature for the *neptune* attack might include that particular source IP address and their detector could fail to detect any *neptune* attacks that used a different spoofed source address.

5 Future Work

We are currently developing or researching the following enhancements or extensions to the MAGNA system which would serve to increase its flexibility, utility, and ease-of-use:

- We are conducting research on automated attack modeling with the goal of developing a program that would build attack models directly from the captured attack traffic (as discussed in Section 3.1). Note that attack packets must still be isolated by a human analyst, no current IDS can perform that task accurately enough to create a valid model.
- We are currently refining the probability-based modeling technique that was discussed in Section 2 to produce more realistic models of attack behavior
- We are extending the MAGNA system to generate a wider range of attacks and to provide more flexibility in modifying modeled attacks.

6 Conclusions

A new method for attack generation was presented which uses models of the behavior of captured network attacks to produce realistic, synthetic copies of those attacks under controlled conditions. The accuracy

and effectiveness of this system was demonstrated in experiments where previously captured attacks were modeled and the traffic generated from those models was shown to accurately recreate the original attacks.

This technique provides researchers with more control over the execution of attacks in a laboratory setting, improving their ability to test new algorithms and to compare those designs with existing systems. It can also be used to test existing systems for known vulnerabilities by subjecting them to controlled attacks.

References

- [1] Computer Emergency Response Team, "CERT Coordination Center at Carnegie Mellon University," www.cert.org, 2004.
- [2] Computer Economics, "Computer Economics Inc. web site," www.computereconomics.com, 2004.
- [3] L. M. Rossey, R. K. Cunningham, *et al.*, "LARIAT: Lincoln adaptable real-time information assurance testbed," in *Proceedings, IEEE Aerospace Conference*, 2002.
- [4] J. Haines, R. Lippmann, D. Fried, *et al.*, "1999 DARPA intrusion detection evaluation: Design and procedures," Massachusetts Institute of Technology, Lincoln Laboratory Technical Report 1062, 2001.
- [5] R. Maxion and K. Tan, "Benchmarking anomaly-based detection systems," in *Proceedings, International Conference on Dependable Systems and Networks*, 2000.
- [6] S. Northcutt and J. Novak, *Network Intrusion Detection: an Analyst's Handbook*, 2nd ed. New Riders Publishing, 2001.
- [7] T. Champion and M. Denz, "A benchmark evaluation of network intrusion detection systems," in *Proceedings, IEEE Conference on Aerospace Systems*, 2001.
- [8] H. Debar, M. Dacier, *et al.*, "An experimentation workbench for intrusion detection systems," IBM Research, Zurich, IBM Research Report RZ 2998, 1998.
- [9] N. Puketza, M. Chung, R. Olsson, and B. Mukherjee, "A software platform for testing intrusion detection systems," *IEEE Software*, Sep/Oct 1997.
- [10] N. Athanasiades, R. Abler, *et al.*, "Intrusion detection testing and benchmarking methodologies," in *Proceedings, IEEE International Workshop on Information Assurance*, 2003.
- [11] K. Kendall, "A database of computer attacks for the evaluation of intrusion detection systems," Master's thesis, Massachusetts Institute of Technology, 1999.