CS 638 Lab 7: Introduction to Network Security

Joe Chabarek and Paul Barford
University of Wisconsin –Madison
jpchaba,pb@cs.wisc.edu

Securing information technology (broadly defined as networks, the host systems which they connect and the data that resides on those hosts) from unwanted access or attack is an extremely challenging proposition. Consider the following four issues:

1. There is nothing inherent in computing or communication systems that provide for or otherwise facilitate security. This means that nearly every aspect of IT security is an after-the-fact, bolt-on solution that may be able to be avoided by attackers.

2. In general, computing and communications systems provide for an amazing level of anonymity *i.e.,* the basic intention and objective of the Internet is to enable anyone to communicate with anyone else in the Internet in a free and open fashion without necessarily having to disclose who is communicating in a truthful fashion.

3. There is an inherent imbalance in the objectives and requirements for attackers and defenders. Attackers need find only *one* vulnerability in a target system, while defenders must protect against *all* possible types of attacks.

4. The size and complexity of today's networks, operating systems and hardware is extreme and continues to increase. This significantly complicates the task of verifying that a system is free of vulnerabilities.

While there are certainly many other challenges, the combined effect gives attackers significant advantages. Fueled by the very real economic opportunities (*e.g.,* identity thieft, extortion, etc.), there is every reason to fear that malicious activity in the Internet will continue to escalate in the coming years. In a very real sense there is an arms race between malicious parties and the community of IT security analysts and companies and researchers who develop IT security solutions.

1 Overview and Objectives

The IT security domain is vast – spanning the space from very serious mathematics that are the basis for cyptography, to the very practical aspects of day-to-day security management in IT infrastructures. The former is concerned with developing algorithms that can provide a provable level of security – even when adversaries know the algorithms that are being used! The latter is concerned with deploying, configuring and managing systems and tools that together form the defensive framework around either small (*e.g.,* your home) or large IT infrasturctures. Our focus will be on the latter.

This lab is designed to introduce you to some of the basic concepts of *network security*, which is a subset of the more general problem of IT security. Network security can be consider the tasks related to securing the communications infrastruction (all systems in the network but not including end hosts) from unwanted access or malicious attacks. Network security also extends to the tasks associated with analyzing and evaluating traffic carried on the network – the ultimate goal of which is to block all traffic that is identifed as unwanted or malicious.

However, even the domain of network security is well beyond the scope of a single lab. Thus, we will further restrict our focus to a subset of activities, tools and systems that are used by network security analysts in their daily activities. Specifically, we will use a standard Network Intrusion Detection System (NIDS) in our experiments. NIDS are tools used to gather information and report on malicious traffic in networks. NIDS are widely used in networks throughout the Internet. We will also use a standard open-source scanning tool to generate traffic for the NIDS. Hopefully it is clear that if we want to exercise a NIDS, something about the traffic used in the experiments needs to be able to be identified as unwanted or malicious. Instead of using actual attack tools (which can certainly be found on the Internet), we will use a scanning tool which is also used by network administrators for testing their security infrastructures. Finally, in order to provide further context for these tools, we will also cover a few of the standard threats that networks must defend against.

Upon completion of the lab exercises below, students will be able to:

- *Understand basic aspects of network security including both threats and defense methods.*
- *Configure and operate a basic installation of the Snort Network Intrusion Detection System*
- *Configure and operate a basic installation of the Nessus security scanner.*

2 Methods and Tools for Attacking Networks

In this section, we will provide a brief overview of three of the most common threats in the Internet today. While the threats described below are quite serious, they are by no means a complete list. If you are interested in learning more about threats, there are many on-line resources, and you are encouraged to investigate these since understanding the adversary is critical when it comes to building effective defenses. However, be careful! There are also many instances of malicious software (malware) available in the Internet **-do no harm!** since there could be very real legal consequences for you.

Denial of Service (DoS) Attacks are attacks that are meant to prevent a host or network path from operating in a normal fashion by sending a flood of packets that overload capacity. If the target is an end host, this can simply be done by sending a flood of TCP SYN packets (without corresponding ACKs to SYN-ACKs sent by the receiver) that can cause the host to expend its kernel capacity for managing simulataneous connections. Overloading network paths is a bit more complex since it takes a more focused effort that is aware of network topology and capacity. However, this can also be done without too much effort by using multiple systems for the attack. An attack of this kind that employs multiple systems is called a Distributed Denial of Service (DDoS) Attack. DoS attacks are common in the Internet and are often used as a threat for extortion (*e.g.,*"pay me or I will DoS your network or server").

Worms are one of the most well known types of malware in the Internet. Worms are self propogating software that contain at least one exploit (a vulnerability like a buffer overflow, which enables the worm to install itself on the target host) as well as a propogation mechanism (a means for identifying other target hosts). One of the first and best known instances of a worm was the Morris Worm, which was written by Robert Morris who at the time ('89) was a graduate student at Cornell Univeristy. The Morris Worm included three specific exploits for the Unix operating system. The only objective of the Morris Worm was self-propagation, but it still caused huge problems in the Internet. More modern worms that have received a lot of attention in the popular press including Code Red I, Code Red II, Nimda, Sasser, and Blaster have caused significant damage by flooding links and requiring huge "disinfecting" efforts. A more dangerous threat are so-called flash worms that use a pre-computed hit list of target hosts in order to rapidly propogate. With small exploits (*e.g.,* the Slammer worm was a single UDP packet) a flash worm could cause large infection rates in seconds.

Botnets are one of the most significant threats to networks today. Botnets consist of groups of computers that have been compromised by software distributed by a "bot herder" who then uses command and control systems to operate the systems remotely, typically without the knowledge of the system's owner. Typical uses of botnets include information theft, sending spam, and conducting distributed denial of service attacks. Many computers in botnets are systems whose owner's neglected to update virus protection and therefore has open vulnerabilities for which there are well known exploits, or whose users downloaded some kind of software from the Internet which contains malicious code. The distributed nature of botnets complicates many security policies especially blacklisting (identifying certain hosts as malicious and filtering out their traffic). Millions of systems are part of botnets today and there is a very active underground economy based on "renting" botnets for a fee. The widely distributed and dynamic nature of botnets make them very difficult for network security analysts to deal with.

3 Tools for Securing Networks

There are a surprisingly small number of tools that are used by network security analysts to defend against unwanted or malicious attacks. We will describe several of the common tools below. The best tools for protecting against unwanted or malicious attacks is common sense, and keeping all of the security software and infrastructure updated and tested.

3.1 Firewalls

A firewall is a network device that is typically deployed at the edge of a network, and protects hosts within the network from specific types of external traffic. It does not, however, protect hosts from other hosts on the same network so if a virus infiltrates the network it can spread freely in the absence of other countermeasures. Firewalls inspect all traffic traversing a link and employ user-defined rulesets to allow or deny each packet on the link. A firewall can be either *inclusive* where all traffic that is not explicitly allowed is blocked and *exclusive* where any traffic is allowed onto the network except that which is specifically blocked. The key feature of firewall rulesets is that they are very coarse-grained, only considering information contained in TCP/IP headers (*e.g.,* block all traffic on a specific port or from a specific network source address range). More recently developed firewalls incorporate features that allow various forms of proxying and deep packet inspection, but these operations require a large amount of resources and don't usually scale to wirespeed performance. Internal firewalls and filtering have also become common in enterprise networks, but the construction and maintenance of rulesets in a provable manner is still a research topic.

3.2 Intrusion Detection Systems

Network Intrusion Detection Systems (NIDS) are important tools in network security infrastructures. Like firewalls, NIDS are network devices typically deployed on the edge of a network *e.g.,* on an ingress/egress link. The goal of NIDS is to identify and report all malicious traffic observed on a link in a network. NIDS inspect all packets on a link: if a packet or sequence of packets matches one of the *signatures* in the NIDS database, it is identified as malicious and an alert is generated. Key differences between NIDS and firewalls is that NIDS do not actively block/drop packets and that that the signatures can include both header and payload information. The composition of signatures is critical to NIDS effectiveness since exploits often have well defined patterns in packet payloads. However, attackers can also use obfuscation techniques to avoid being detected! There are various combinations of firewalls and IDS systems that are usually referred to as active response systems or intrusion prevention systems. Once a signature has been triggered dynamic rules are sent to the firewall which quite possibly is in the same physical device or a TCP RST packet is injected into the network to stop the connection.

Snort is an open-source intrusion detection system that is widely used in the Internet. Snort has an active community of contributors who in addition to making improvements to the basic NIDS engine, also make periodic updates to the signature set so that the latest malware can be identified. Snort comes pre loaded with a base ruleset. However there are online resources such as Bleeding Snort that provide tuned rulesets. Snort will be used in the experiments described below. The use of Snort in this lab is quite basic, but you are encouraged to investigate it more fully on your own.

3.3 Penetration Testing via Nessus

Savvy network administrators understand that it is important to periodically test and evaluate their infrastructure for vulnerabilities. A commonly used tool to scan a network for vulnerabilities is Nessus. Nessus is essentially a packet generator with the specific intent of creating traffic streams that can help to identify weakness in a network.

Nessus software runs as a daemon on a host. When instructed, the daemon takes a list of machines to scan and tries to connect to different ports on these machines in search of a service or open port which is unpatched or open and can be exploited by malicious software. A client must be run which contacts the daemon. The command line client is run with the

```
/opt/nessus/bin/nessus -q -T txt localhost 1241 <user> <password> targetsfile resultfile
```

The targets file contains the ip addresses of the machines that you want to scan, one per line. The resultfile is the name of the output file. The user and password are set during installation, outlined below. You are encouraged to investigate the various capabilities of Nessus - it is a cool and very useful tool. **But, be sure to restrict your use of Nessus to Schooner systems since its traffic will be seen a malicious if generated on live networks.**

4 Setup

Setup a simple topology as shown in the prelab. You need at least two machines running FC6-STD which will be used for snort and nessus, but to make things a bit more interesting, add a few machines. Use different OSID types on these machines, FC4-STD,FBSD54-STD,and FBSD62-STD are reccomended. You can just use these machines as scanning targets.

4.1 Snort Setup

Installing the snort RPM from the cs638/rpms directory, adds the snort binaries, which should be kept relatively up to date, but does not install the rules, which should be the most currently available. Once your snort node has booted, we will need to manually install a ruleset and change a few items in the snort configuration.

First untar the ruleset into place

```
Last login: Thu Dec  6 10:29:39 2007 from ops.schooner.wail.wisc.edu
[blodge@node0 ~]$ sudo /bin/tar -C /etc/snort -xvzf /share/snortrules-pr-2.4.tar.gz
rules/
rules/attack-responses.rules
rules/backdoor.rules
.....
```

Now using ifconfig find the experimental interface you will want snort to listen on. You will have to manually change the interface definition in /etc/rc.d/init.d/snortd from eth0 (the default) to the proper interface. An lastly copy the modified snort configuration file into place and start snort.

```
[blodge@node0 ~]$ sudo cp /proj/cs638/scripts/snort.conf /etc/snort/snort.conf
[blodge@node0 ~]$ sudo /etc/rc.d/init.d/snortd start
Starting snort:                                        [  OK  ]
[blodge@node0 ~]$
```

4.2 Nessus Setup

The Nessus RPM installation can take a significant (a few minutes) amount of time to install, which might cause problems if you try to have the testbed auto install it, so we will do it manually. When your host is up login and run the following.

```
Last login: Thu Dec  6 14:58:59 2007 from ops.schooner.wail.wisc.edu
[blodge@node0 ~]$ sudo rpm -Uvh /proj/cs638/rpms/Nessus-3.0.6-fc6.i386.rpm
```

This will install Nessus into /opt/nessus. Instructions on adding a nessus user and starting the daemon are shown after the RPM installation process.

5 Tasks

1. Begin the Nessus daemon (described above) on one of the nodes.
2. Run the nessus client (as described above)
3. Look at the results file carefully as it contains some very informative descriptions of different security issues.
4. Next, set up the Snort NIDS on a node that sits between the host running Nessus and the nessus clients.
5. Run the same Nessus scans and see what is generated in the Snort log.
6. Download and install the Bleeding Snort rule set from the Internet, and rerun the Nessus tests. Again, examine the alert log.