ISSN: 1545-679X

Information Systems Education Journal

Volume 5, Number 5 **http://isedj.org/5/5/** **May 3, 2007**

In this issue:

Virtual Laboratory Intrusion Detection Experience for Information Systems Professionals

Valerie J. Harvey
Robert Morris University
Moon Township, PA 15108 USA

Randall Johnson
Robert Morris University
Moon Township, PA 15108 USA

John C. Turchek
Robert Morris University
Moon Township, PA 15108 USA

Abstract: This paper describes how to design and implement an intrusion detection module that may be implemented in various courses taught in an information system curriculum and covers the industry-standard Snort Open Source intrusion detection system (IDS). This paper proposes that virtualization offers three significant instructional advantages in delivering a rich IDS experience: (1) server independence giving each student control of an IDS configuration, (2) a unique IP address on the "virtual" network for each server so that students are able to work in teams, including in distance learning situations, and (3) demonstration of centralized logging as typically deployed in production networks by configuring each virtual machine to send log messages to the instructor's virtual machine. Students then can generate, observe, log, and analyze various types of network traffic between their virtual servers in a safe, ethical manner. Documentation of commands and results is included.

Keywords: intrusion detection, virtualization, information security

---

Recommended Citation: Harvey, Johnson, and Turchek (2007). Virtual Laboratory Intrusion Detection Experience for Information Systems Professionals. *Information Systems Education Journal, 5* (5). http://isedj.org/5/5/. ISSN: 1545-679X. (Also appears in *The Proceedings of ISECON 2006:* §3722. ISSN: 1542-7382.)

This issue is on the Internet at **http://isedj.org/5/5/**

The **Information Systems Education Journal** (ISEDJ) is a peer-reviewed academic journal published by the Education Special Interest Group (EDSIG) of the Association of Information Technology Professionals (AITP, Chicago, Illinois). • ISSN: 1545-679X. • First issue: 8 Sep 2003. • Title: Information Systems Education Journal. Variants: IS Education Journal; ISEDJ. • Physical format: online. • Publishing frequency: irregular; as each article is approved, it is published immediately and constitutes a complete separate issue of the current volume. • Single issue price: free. • Subscription address: subscribe@isedj.org. • Subscription price: free. • Electronic access: http://isedj.org/ • Contact person: Don Colton (editor@isedj.org)

EDSIG activities include the publication of ISEDJ, the organization and execution of the annual ISECON conference held each fall, the publication of the Journal of Information Systems Education (JISE), and the designation and honoring of an IS Educator of the Year. • The Foundation for Information Technology Education has been the key sponsor of ISECON over the years. • The Association for Information Technology Professionals (AITP) provides the corporate umbrella under which EDSIG operates.

Virtual Laboratory Intrusion Detection Experience for Information Systems Professionals

Valerie J. Harvey
harvey@rmu.edu
Computer and Information Systems, Robert Morris University
Moon Township, PA 15108 USA

Randall Johnson
johnsonr@rmu.edu
Technical Services, Information Systems, Robert Morris University
Moon Township, PA 15108 USA

John C. Turchek
turchek@rmu.edu
Computer and Information Systems, Robert Morris University
Moon Township, PA 15108 USA

ABSTRACT

This paper describes how to design and implement an intrusion detection module that may be implemented in various courses taught in an information system curriculum and covers the industry-standard Snort Open Source intrusion detection system (IDS). This paper proposes that virtualization offers three significant instructional advantages in delivering a rich IDS experience: (1) server independence giving each student control of an IDS configuration, (2) a unique IP address on the "virtual" network for each server so that students are able to work in teams, including in distance learning situations, and (3) demonstration of centralized logging as typically deployed in production networks by configuring each virtual machine to send log messages to the instructor's virtual machine. Students then can generate, observe, log, and analyze various types of network traffic between their virtual servers in a safe, ethical manner. Documentation of commands and results is included.

Keywords: intrusion detection, virtualization, information security

1. INTRODUCTION AND RATIONALE

This paper describes how to design and implement a network intrusion detection module that may be implemented in various courses taught in an information system curriculum or in a specific information security curriculum. Such a module is suitable for use in courses at both the undergraduate and graduate levels. It serves to introduce general intrusion detection system (IDS) concepts as encountered in Network Intrusion Detection Systems (NIDSs) and Host Intrusion Detection Systems (HIDSs). The appendices contain examples of commands and the results to be verified in inspecting logs.

With respect to model I/S curricula (Gorgone et al., 2002), the module may be used with

IS2002.6 Networks and Telecommunication and MSIS 2000.3 Data Communications and Networking. model. ISACA was one of the first professional associations to prepare a model curriculum for information security to assist in the development of programs for I/S assurance professionals. Their 1998 Model Curriculum was recently updated in 2004 and has five Domains including the Technical Infrastructure and Operating Practices Domain that includes specific requirements for intrusion detection systems. Thus, this module would be ideal for any course mapped to the ISACA model (ISACA, 2006). Other drivers of curricular content include the Homeland Security Presidential Directive/Hspd-7 of December 17, 2003 (Bush, 2003) on Critical Infrastructure Identification, Prioritization, and Protection, NSA and the enhancement of Open Standards such as COBIT, ITIL, and ISO 17799. COBIT 4.0 was released in late 2005. One of its domains is the "Deliver and Support" Domain which has specific objectives focused upon access controls. Similar components exist for ITIL as well as for ISO 17799. For ITIL, they are contained in Security Management, Security Management Measures, 4.2.4 Access Controls. Intrusion detection is addressed in several areas in ISO 17799 such as sections 3.1, 4.1, 4.2, 6.1, 7.1, 8.1, 8.5, 8.6, 9.1, 9.2. 9.4, 9.5, and 9.6. ISO 17799 is also known as ISO/IEC 17799:2000 or more recently as ISO/IEC 17799:2005. It was established by a joint technical committee (the ISO/IEC JTC 1) in 2000 and later updated in 2005. Thus, any course mapped to these standards may also incorporate this module.

Since it is necessary to assure that data being audited has not been improperly modified, Sarbanes-Oxley "Section 404 also requires the company's auditor to attest to, and report on management's assessment of the effectiveness of the company's internal controls and procedures for financial reporting in accordance with standards established by the Public Company Accounting Oversight Board" and further, "IT and the process owner must be responsible for: Access control over sensitive and critical applications and data files supporting the process (including security for preventing viruses and hacker intrusions)" (Anand, 2004). The importance of intrusion detection is also specified by the National Security Agency (NSA, 2006).

According to the Information Security Manager Key works/education contents matrix (Table 5) of Kim and Surendran (2001), intrusion detection and interception includes the following key works: selection of safeguards, test of selected safeguard, safeguard implementation, operating and maintenance, and monitoring.

2. INSTRUCTIONAL REQUIREMENTS

The course seeks to help students appreciate the need for intrusion detection through configuring and using an intrusion detection tool and using tools to generate and send messages to hosts and logging the impacts detected and recorded through the actions of the intrusion detection tools. An intrusion is an attempt to penetrate or interfere with a host through a message or set of messages sent to it, in order to modify its operation and/or data under its control. The attempts could be designated as attempted "break-ins." Mukkamala et al. (2006) characterize intrusions as "attempts to compromise information assurance." It is important for students to have hands-on exercises to gain direct experience in recognizing and documenting intrusions, received over the Internet in the form of packets, in a controlled environment.

To cover intrusion detection in the information security curriculum, it is advisable to cover the industry-standard Snort open source intrusion detection system (IDS) (Chen at al, (2004)). Unfortunately, offering a lab based on Snort poses a challenge for institutions without UNIX or Linux-based labs. Although Snort does run on Windows, its native and typical production deployment platform is a UNIX-like system such as Linux. On a UNIX-like system, access to packet capture capability typically requires root privileges. Snort could certainly be run in shell accounts on a single server, with a shared root password or the sudo command, but virtualization offers three significant instructional advantages. First, each virtual machine is an independent server, so each student has full, private control of their IDS configuration, which cannot be changed or even seen by other students. Second, each virtual server has a unique IP address on the "virtual" network,

so students are able to work in teams to generate, observe, and log various types of network traffic between their virtual servers in a safe, ethical manner. Finally, configuring each virtual machine to send log messages to the instructor's machine is not only very convenient during the lab session, but also demonstrates centralized logging as typically deployed in production networks. The results are a richer learning experience. Although the systems in RMU labs run Windows XP, as do most of the students' systems at home, the use of Linux virtual machines and secure telnet enabled them to experience Linux-based Snort intrusion detection. The example described in this paper used Xen 2.0. In the meantime, Xen 3.0 has become available.

3. TECHNICAL REQUIREMENTS AND ARCHITECTURE

This module requires a single Linux server with sufficient disk space, RAM, and CPU power to support individual virtual machines for all the members of a course on information security or on networking, and a suitable range of IP adresses. The hardware server ("host") is partitioned into virtual servers using the free Xen hypervisor. The Xen host ("dom0") runs Linux with the Xen hypervisor and tools on hardware and the virtual machines ("domU" instances) run a Linux kernel modified to run on Xen. Each virtual machine sees a single Ethernet interface, eth0, which is attached to a virtual interface for each domU in the host and bridged to the host's physical Ethernet interface as shown in Figure 3.1. Thus, each virtual server is visible on the same network segment as the host. Access to the virtual machines can be restricted, if desired, by implementing iptables rules in the host. The virtual lab architecture for the IDS exercise described in this paper offers:

- Full, independent root access for each student
- Full isolation of student configuration changes during the lab exercise
- Full bridged networking equivalent to a physical switched network for the virtual machines
- Opportunity for configuration to implement automatic logging of lab exercise results to a central logging server operated by the instructor (see Lockhart, 2004), pp. 135-136.

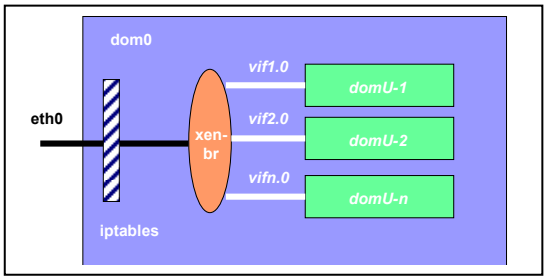

Figure 3.1. - Xen Bridged Networking Architecture

To implement centralized log file collection and analysis, it is necessary to configure **syslog-ng** for a centralized audit host with all student server logs forwarded to the instructor's virtual machine. To implement intrusion detection it is necessary to run the Snort IDS. In order to detect port scans, carry out simulated IIS exploits, a custom Snort rule needs to be written and tested. The virtual machines need to be preloaded and preconfigured with the following software: **pcap, snort, nmap, ftester, telnet, netstat, ps** (to check active processes), and **vmstat** (to check available memory).

4. DESIGN, PREPARATION, AND IMPLEMENTATION

A group of IP addresses is obtained, one for each student in the group, one for the instructor, and one for a "mystery" operating system. The virtual machines are set up:

1. The host server is prepared
2. Range of IP addresses determined
3. A template virtual machine is created
4. Scripts create individual student virtual machines from template
   4.1. On the first startup, a script in the template creates the user, sets the root password, and emails the password to student and instructor
5. Configured the instructor virtual machine

   *5.1.*Configured instructor *syslog-ng*

5.2. Configured instructor */var/log/HOSTS* directory (central logging destination)

6. The "mystery" operating system virtual machine is custom installed.

Fig. 4.1 Virtual Lab Host Server

Figure 4.2 Students Working on Laptops

The addresses do not need to be publicly routable (and may be of the restricted addresses specified in RFC 1918) unless it is desirable for students to be able to access their virtual machines away from the campus network. Providing VPN for students to access the campus network would eliminate the need for public addresses. The RMU lab utilized routable IP addresses in the range of k.m.n.101 through k.m.n.122 for the student virtual machines, where *k.m.n.* represent the first three octets of the IPv4 address. The host server hardware is also obtained and set up. The host server used for the lab at RMU was an HP ML370G3 with a single 2.8 GHz CPU, two 36 GB SCSI disks in a hardware-based RAID-1 mirror, and 1 GB RAM. This server supported a lab of 22 virtual machines each with 48 MB RAM, 1 GB disk. RAM is the limiting factor, because Xen allocates the full amount of RAM for each virtual machine out of the host's *physical* RAM at domU startup. The virtual machines performed admirably with their low 48 MB RAM. Neither host CPU speed nor disk space was an issue at any time during the RMU lab. Regarding the location of the set of virtual machines in the campus network, each student only needs connections from the outside world to the SSH port (22) and from the other students in the group.

Once the host is prepared, the virtual machines are set up. First, a template virtual machine is created. The template is built from a Debian Sarge netinst image running Linux kernel 2.6.11.12-xenU. Snort, nmap, and syslog-ng and all their dependencies are preinstalled from Debian packages because the focus of the lab is intrusion detection, not Linux system administration. Next, a set of scripts was prepared to automate the creation of a virtual machine for each student. For each student username listed in a simple ASCII roster file, the script creates a virtual machine consisting of a disk image file and a Xen configuration file. The hostname is the student's username, which simplifies interpretation of syslog entries in the Instructor's central log server. When each virtual machine starts for the first time, a script contained in the template disk image generates and sets a strong root password and then e-mails it to the student and the instructor. See Appendix A for example 4.1 of e-mail to a student.

Along with this e-mail, students are separately notified in advance to have laptops available for classroom use. This is not a general requirement; each student requires only access to any computer (Windows, Linux/UNIX, or Mac) that has a SSHv2 client available.

For another example of the use of virtual machines in information assurance education see Shepens et al. (2003) and Dodge (2005).

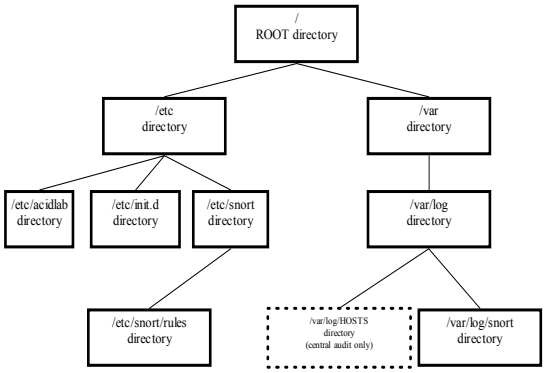

Fig. 5.1 Directory Model (selected directories) for RMU Information Security Virtual Machines

5. DOCUMENTION

Documentation needs to specify the editors that will be available to use and training resources with regard to these editors. In this case the editors **vi**, **nano**, and **emacs**, all widely used in the Unix and Linux environments, were all preloaded onto the students' virtual machines. Students need to be instructed on downloading, installation, and configuration of a suitable Secure Shell telnet system, such as PuTTy. In this case PuTTy was recommended to the students and they were showed how to configure it to (1) provide *keepalive* capability for sessions, (2) use the SSH protocol, and (3) increase buffer space to at least 500 lines for the display. To assure appropriate practice and log development, a group work procedure should be described, such as round robin port scanning, which would assure interesting and realistic results to review. Students will also need to check occasionally whether more than one instance of Snort has been started, and they should check free memory each week when beginning log work.

Students are provided with a map of the directory structure for selected relevant directories (see Figure 5.1) to help them visualize their virtual machine working environment. This model includes the directory used in configuring syslog, the directory where logs reside, and the directory where Snort's rules reside. They can also see that the instructor has a special directory to receive central audit copies of their logs.

Students are provided with a map of the directory structure for selected relevant directories (see Figure 5.1) to help them visualize their virtual machine working environment. This model includes the directory used in configuring syslog, the directory where logs reside, and the directory where Snort's rules reside. They can also see that the instructor has a special directory to receive central audit copies of their logs.

6. LEARNING ABOUT SYSLOG AND CONFIGURING SYSLOG FOR CENTRAL AUDITING

Students learn to send messages that will be recorded in syslog (**syslog** messages) using logger. See Appendix B, Example 6.1).

They learn to configure syslog for forwarding of messages to a designated central audit host (the virtual machine in the group belonging to the course instructor) in the virtual machine group by editing the syslog-ng configuration file syslog-ng.conf to add the text shown in Appendix B, example 6.2).

Subsequent to the configuration file change, each student should restart syslog and verify that syslog has been stopped and restarted (see Appendix B). Students then should use **netstat** to confirm TCP connections to the central audit host (see Appendix B, examples 6.6-6.7).

The instructor can easily monitor course activity by reviewing the files in

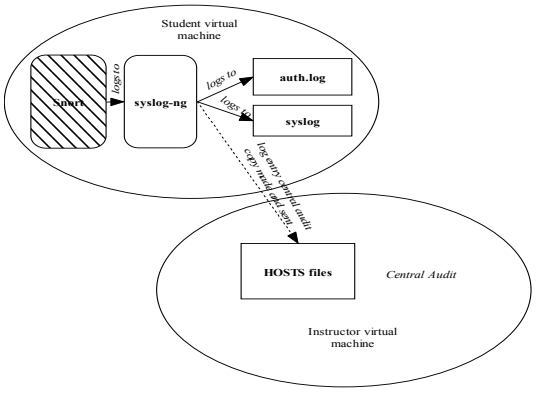

Figure 6.1 – Model of Student VM Logging and Central Audit Logging

/var/log/HOSTS to check for presence, size, and most recent date the file was changed. If any student fails to edit the configuration file successfully, the instructor will detect this quickly by the absence of any log files for that student in /var/log/HOSTS (see Figure 6-1.) It was found that in some cases student damaged the file and in some cases deleted it. An extra copy of the syslog-ng configuration file was provided in the virtual machines so that students could recover easily from such errors.

7. RUNNING AND CONFIGURING SNORT, USING TOOLS, AND EXPLOITING LOGS

Starting Snort and using nmap tool to scan a target

Students first learn to start Snort, initially not configured to enable **syslog**, and to use **nmap** to scan an assigned target, so they check the authorization log for results. They see that no results have been logged and thus learn that they must explicitly enable **syslog** in starting Snort in order to produce log results.

They can also test Snort in sniffer mode by the appropriate command to become more familiar with packet structure.

Using Snort with preconfigured Snort rules

Students then learn to start Snort with its beginning rule configuration and configure Snort operation on the command line.

Once Snort has been started, students conduct a port scan of one of the virtual machines of their colleagues in the course by using **nmap**. A round robin scan assignment assures that every student's virtual machine is scanned by someone else in the group of students. They then verify that that the scan(s) are not logged by checking the **auth.log** file.

They then restart Snort, adding the proper configuration –s to invoke logging. Again they use nmap to invoke port scanning (Example 7.3, Appendix C) and check the auth.log file (Example 7.4, Appendix C). They can then verify the result by noting various logged entries like the one in Example 7.5 (Appendix C).

Using the ftester tool.

Students also learn to use **ftester** to simulate a cmd.exe attack on IIS and verified "exploit" capture based on preconfigured Snort rule (Appendix C, examples 7.7-7.8). Students can then observe the result **syslog** entries.

Writing and using custom Snort rules.

Students learn more control over simulation by writing a custom snort rules. The first example invokes alerts to a TCP connection to an arbitrary port address (12345). They can edit **local.rules** as shown in Appendix C, example 7.9). See also Alder et al. (2004), Chapter 5, Koziol (2003), Chapter 12, Cox and Gerg (2004), Chapter 7), Orebaugh et al. (2005), Shema et al. (2006) and Rehman (2003).

It is recommended that each local rule be given a Snort ID (sid) in the appropriate range for local rule sids (>1000000). After completing editing of **local.rules** they should restart Snort and attempt to initiate a TCP connection to port 12345 (a port number assumed unused) by using telnet (see Appendix C, examples 7.6-7.8)

Exploiting logs.

Logs produced by exercises are reviewed. Attacks from outside the group should be identified, classified, and documented. Student can use a utility, like WinSCP, to transfer log files to their systems for analysis. See Appendix C, examples 7.11-7.22).

Tool limitations.

Configuration is accompanied by discussion of what configuration achieves and examples of how certain configurations can result in packet loss (Alder et al. (2004), p. 488), or detection that is narrow to be effective. Hardware requirements to avoid packet loss are covered in Alder et al. (2004), pp. 472-479.

8. OS FINGERPRINTING

Operating system fingerprinting can be carried out. A "mystery" operating system can be used on one member of the virtual machine group to provide practice (see Appendix D, examples 8.1-8.3).

Students can then verify the results. In some cases there may not be enough ports open to provide a result. A result for a "mystery" system on which NetBSD is running is shown in Appendix D. They can scan another student system to compare results with those from the mystery system. They then get a sense of the degree to which their operating system information is exposed and vulnerable.

9. ETHICAL CONSIDERATIONS AND INSTRUCTION

Ethical issues relating to scanning of ports must be explained and reviewed (Thibodeaux et al. (2001), Jamieson (2001)). Students must be alerted to the impropriety or prohibition of scanning ports in organizations and of systems over which the students does not have administrative control or permission. Port scanning is permitted (and required as a learning activity) for the group of virtual machines serving the individual members of the course and is therefore permitted in this case for the systems identified by the instructor. Students must receive an explicit policy statement for the course relation to the ethics of port scanning and the limitations on port scanning which may be done within the scope of the course and what may not be done with the tools described and made available in the course. They are informed as to legal penalties and risks for abuse of the computing environment.

Each student wishing to participate in the activities signs a confirmation of having received, read, and understood the policy and of granting permission to fellow students in the course to scan their system and to "attack" it in the manners prescribed for the course. The students are then notified of which systems they have permission to scan or "attack" in prescribed manners.

10. CONCLUSION AND RECOMMENDATIONS

This module was included in a graduate course in I/T Assurance and Security at Robert Morris University. Students learned the principles of intrusion detection by configuring using and an industry-standard tool, by learning about vulnerabilities, and by learning about performance limitations of tools. Students worked well together to assure that everyone in the course received entries in their logs. It was a great advantage to have the virtual machines so that each student had an independent virtual machine and IP address. Since this course was offered in the evening for students employed during the day, the students very much appreciated being able to do their intrusion detection exercises at home. They exchanged phone numbers and e-mail addresses so they could collaborate effectively in their exercises. Using publicly routable IP addresses was thus quite valuable in this course. Such an approach could also support distance learning.

Problems and their correction

Student documentation was in most respects adequate and effective. The first version of student documentation identified a particular address in certain examples with the result that that student's virtual machine generated a disproportionate share of log files and quickly ran out of memory. The documentation was revised and reissued and students were encouraged to check and report on available memory more regularly. It was also found that students would often start multiple instances of Snort without realizing they had done this, with adverse effects, so they were then encouraged to regularly check active processes so they could detect and correct such a situation.

It is important that students have prerequisite Linux editing skills, so this should receive focus early in the course to avoid problems and to offer individual assistance.

Plans for the future

For the next iteration of the course, it was decided to add ACID (Analysis Console for Intrusion Detection), to include more memory for each student's virtual machine, to include automatic notification by e-mail, and to design a greater variety of packet generation for students to detect and analyze and add instruction for this to the documentation.

11. REFERENCES

Alder et al. (2004) Alder, Raven, Babbin, Jacob, Baker, Andrew R., Caswell, Brian, and Poor, Mike, Doxtater, Adam, Forster,

James C., Kohlenberg, Toby, and Rash, Michael, Snort 2.1 Instrusion Detection, 2nd ed. (Syngress, 2004).

Anand (2004). Anand, S.*The Sarbanes-Oxley Guide for Finance and IT Professionals* (Sarbanes-Oxley Group, 2004), pp. 50-51.

Bush (2003). Bush, George W., "December 17, 2003, Homeland Security Presidential Directive/Hspd-7" at http://www.whitehouse.gov/news/releases/2003/12/20031217-5.html

Chen et al. (2004). Chen, Jim, Tsao, Victor, Williams, Barry, and Olojo, Tokunbo, "Lessons Learned from Teaching Intrusion Detection and Intrusion Prevention with Snort."

Cox and Gerg (2004). Cox, Kerry & Gerg, Christopher, *Managing Security with Snort and IDS Tools* (O'Reilly, 2004).

Dodge (2005). Dodge, Ronald, "Virtual Labs for Information Assurance," Third Annual Information Technology Security Conference, San Diego, CA.

Gorgone et al. (2000). Gorgone, J. T. et al., "MSIS 2000: Model Curriculum and Guidelines for Graduate Degree Programs in Information Systems" http://www.aisnet.org/Curriculum/msis2000.pdf , ACM, AIS.

ISACA (2006) at www.isaca.org/cobit/

Jamieson (2001). Jamieson, Shaun, "The Ethics and Legality of Port Scanning," (2001), available online at http://www.sans.org/rr/whitepapers/legal/71.php

Kim and Surandran (2001). Kim, Ki-Yoon, and Surendran, Ken, "A Curriculum Development for Information Security Manager Using DACUM," In *The Proceedings of ISECON 2001*, v. 18 (Cincinnati): §39a., available online at http://isedj.org/isecon/2001/39a/ISECON.2001.Kim.pdf

Koziol (2003) Koziol, Jack, *Intrusion Detection with Snort* (SAMS, 2003).

Lockhart (2004). Lockhart, Andrew, *Network Security Hacks: 100 Industrial-Strength Tips & Tools* (O'Reilly, 2004).

Mukkamala (2006). Mukkamala, Srinivas, Sung, Andrew, and Abraham, Ajith, "Cyber-Security Challenges: Designing Efficient Intrusion Detection Systems and Anti-Virus Tools." In Vemuri, V. Rao, *Enhancing Computer Security with Smart Technology*. (Auerbach, 2006), pp. 125-163.

NSA (2006). NSA, "Information Assurance" at http://www.nsa.gov/ia/.

Orebaugh et al. (2005) Orebaugh, Angela D., Biles, Simon , and Babbin, Jacob, *Snort Cookbook* (O'Reilly, 2005).

Rehman (2003). Rehman, Rafeeq Ur, *Intrusion Detection Systems with Snort: Advanced IDS Techniques with Snort, Apache, MySQL, PHP, and ACID* (Prentice Hall, 2003).

Shema et al. (2006). Shema, Mike, Davis, Chris, Phillip, Aaron, and Cowen, David, "*Anti-Hacker Toolkit,* 3rd ed. (McGaw-Hill/Osborne, 2006).

Shepens et al. (2003). Shepens, Wayne J., Ragsdale, Daniel J., Surdu, John R., and Schafer, Joseh, "The Cyber Defense Exercise: An Evaluation of the Effectiveness of Information Assurance Education," at www.blackhat.com/presentations/bh-federal-03/bh-fed-03-dodge.pdf

Thibodeaux et al. (2001). Thibodeaux, Maxwell S., Ragsdale, Daniel J.,and Marin, John A., "Ethical Aspects of Information Assurance Education," Proceedings of the 2001 IEEE Workshop on Information Assurance and Security (United States Military Academy, 2001

APPENDIX A (Section 4, Design, Preparation, Implementation)

Example 4.1. Example e-mail to student (*k.m.n.* represent the first three octets of the IPv4 address of the student's virtual machine).

From: root <root@snortlab-00>
To: <xyzst1@rmu.edu>, <instructor1@rmu.edu>, <techserv1@rmu.edu>
Date: 3/16/2006 5:03 pm
Subject: INFS6760A: Lab Setup Information for xyzst1

Virtual machine IP address: *k.m.n.*101
Username: xyzst1
Password [and root password]: 5514af38

APPENDIX B (Section 6, Syslog and Central Audit)

Logger command example for syslog checking:

Example 6.1. command: `logger –i –p local3.info "System information: "`uname –a`` -i adds the process ID to the message and –p local3 specifies syslog and level. uname –a sends the system identification string.

Syslog configuration file edit:

Example 6.2 code to add (*k.m.n.* represent the first three octets of the IPv4 address of the instructor's virtual machine):

```
destination d_audit {
    tcp("k.m.n.99" port(5140));
};

log {
    source(s_all);
    destination(d_audit);
};
```

Note: IP address k.m.n.*99* is the arbitrarily designated central audit system.

Syslog restart and verification:

Example 6.3 command: `/etc/init.d/syslog-ng restart`

Example 6.4 command: `tail /var/log/syslog`

Example 6.5 result to verify:

```
Mar 16 09:29:31 vm-00 syslog-ng[966]: syslog-ng version 1.6.5 going down
Mar 16 09:30:01 vm-00 syslog-ng[8003]: syslog-ng version 1.6.5 starting
```

Using netstat to confirm TCP connection to central audit host

Example 6.6 command: `netstat -an | grep 5140`

Example 6.7 result to verify (*k.m.n.* represent the first three octets of the IPv4 address of the local virtual machine):

```
vm-00:/etc/syslog-ng# netstat -an | grep 5140
tcp        0      0 k.m.n.124:3285     k.m.n.99:5140      ESTABLISHED
vm-00:/etc/syslog-ng#
```

APPENDIX C (Section 7, Snort and Using Logs)

Snort command configuration:

Example 7.1 command:

```
snort –evi eth0
```

Configuring switches:
-e display Layer 2 packet data
-i <network interface> interface for sniffing (in this case: eth0 (see Fig. 3.1 above))
-v verbose mode

Starting Snort:

Example 7.2 command:

```
snort -A full -c /etc/snort/snort.conf -b -d -i eth0 -l /var/log/snort –z
```

Configuring switches:
-A <alert-mode> sets alert mode to full (full, fast, none)
-c <configuration file> uses configuration file identified
-b enable logging packets in tcpdump format for speed
-d dump application-layer data
-i <network interface> interface for sniffing (in this case: eth0 (see Fig. 3.1 above))
-l <directory> location for logging packets
-s enable syslog
-z <assurance mode> reduces noise

Checking to see if scans are logged:

Example 7.3 command: `nmap –sS <target ip>`

Example 7.4 command: `tail /var/log/auth.log`

Restarting Snort configured for logging.

Example 7.5 command:

```
snort -A full -c /etc/snort/snort.conf  -b -d -i eth0 -l /var/log/snort -s –z
```

Configuring switch:
-s enable syslog

Verifying scan logging:

Example 7.6 result to verify:

```
Mar 16 10:22:20 vm-00 snort: [122:1:0] (portscan) TCP Portscan {PROTO255}
k.m.n.100 -> k.m.n.124
```

Using and verifying ftester attack:

Example 7.7 command:

```
cd /root
ftester-1.0/ftest -f ftest-cmd.exe.conf -v -d 0.1 -s 1 -F -g 2
```

Example 7.8 result to verify:

```
Mar 16 10:34:15 vm-00 snort: [1:1002:7] WEB-IIS cmd.exe access [Classification:
Web Application Attack] [Priority: 1]: {TCP} <IP address 1> -> <IP address 2>
```

Snort local rule edit:

Example 7.9 line to add:

```
alert tcp any any -> any 12345 (msg: "TCP traffic to port 12345"; flow:stateless;
sid:1000001);
```

Example 7.10 command: telnet k.m.n.99 12345
They can then observe the result, in the targeted machine, by inspecting **auth.log**.

Example 7.11 result to verify:

> Mar 16 11:28:58 vm-00 snort: [1:0:0] TCP traffic to port 12345 {TCP} <IP address 1>:4644 -> <IP address 2>:12345

Examples of exploiting logs (7.12-7.22)

Example 7.12 Result to verify:
Mar 18 11:09:37 <host IP address> sshd[1650]: Illegal user sgi from 211.162.78.106
Mar 18 11:09:37 <host IP address> sshd[1650]: reverse mapping checking getaddrinfo for servers.szgwbn.net failed - POSSIBLE BREAKIN ATTEMPT!

Example 7.13 Result to verify:
May 7 15:00:12 snort-xen-01 sshd[18194]: Illegal user test from 211.125.74.47
May 7 15:00:14 snort-xen-01 sshd[18196]: Illegal user oracle from 211.125.74.47

Example 7.14 Result to verify:
May 7 07:43:06 snort-xen-01 snort: [122:1:0] (portscan) TCP Portscan {PROTO255} 205.146.48.114 -> 205.146.48.99
Dictionary attack from possibly spoofed IP address:

Example 7.15 Result to verify:
Apr 11 11:08:07 vm-mxkst15 snort: [1:1420:11] SNMP trap tcp [Classification: Attempted Information Leak] [Priority: 2]: {TCP} k.m.n.117:47755 -> k.m.n.113:162
Apr 11 11:08:07 vm-mxkst15 snort: [1:1421:11] SNMP AgentX/tcp request [Classification: Attempted Information Leak] [Priority: 2]: {TCP} k.m.n.117:47755 -> k.m.n.113:705

Example 7.16 Result to verify:
Apr 11 16:55:28 vm-mxkst15 sshd[14988]: Did not receive identification string from 82.224.139.101

Example 7.17 Result to verify:
Apr 6 19:21:43 205.146.48.111 sshd[5585]: Failed keyboard-interactive/pam for illegal user waldo from 172.24.3.247 port 2639 ssh2

Students should learn to recognize "normal" log entries that are not exploits.

Example 7.18 Result to verify:
Apr 12 10:17:01 205.146.48.101 CRON[4098]: (pam_unix) session opened for user root by (uid=0)

Example 7.19 Result to verify:
Apr 12 10:17:01 205.146.48.101 CRON[4098]: (pam_unix) session closed for user root
Apr 12 10:20:16 205.146.48.101 syslog-ng[799]: STATS: dropped 0

Example 7.20 Result to verify:
Apr 12 10:30:55 205.146.48.102 thttpd[23086]: up 14400 seconds, stats for 3600 seconds:
Apr 12 10:30:55 205.146.48.102 thttpd[23086]: thttpd - 0 connections (0/sec), 0 max simultaneous, 0 bytes (0/sec), 0 httpd_conns allocated
Apr 12 10:30:55 205.146.48.102 thttpd[23086]: map cache - 0 allocated, 0 active (0 bytes), 0 free; hash size: 0; expire age: 1800
Apr 12 10:30:55 205.146.48.102 thttpd[23086]: fdwatch - 304 selects (0.0844444/sec)
Apr 12 10:30:55 205.146.48.102 thttpd[23086]: timers - 4 allocated, 3 active, 1 free

Example 7.21 Result to verify:
Apr 6 19:43:12 205.146.48.117 kernel: Normal free:0kB min:0kB low:0kB high:0kB active:0kB inactive:0kB present:0kB pages_scanned:0 all_unreclaimable? no

Example 7.22 Result to verify (these entries can reveal Snort's starting and stopping):
Apr 6 09:51:43 205.146.48.112 kernel: device eth0 entered promiscuous mode
Apr 6 09:54:21 205.146.48.112 kernel: device eth0 left promiscuous mode

APPENDIX D (Section 8, OS Fingerprinting)

Nmap command for OS fingerprinting:

Example 8.1 Command: `nmap -O -sS <target IP address>`

Verifying results of nmap command:

Example 8.2 Result to verify:

```
Starting nmap 3.81 ( http://www.insecure.org/nmap/ ) at 2006-03-15 14:08 EST
Interesting ports on <target IP address>.rmu.edu (k.m.n.125):
(The 1662 ports scanned but not shown below are in state: closed)
PORT   STATE SERVICE
22/tcp open  ssh
MAC Address: AA:00:00:72:13:73 (Digital Equipment)
Device type: general purpose
Running: NetBSD
OS details: netbsd 1.6ZH - 2.0RC4
```

Students check each other's reports.

Example 8.3 Result to verify:

```
Starting nmap 3.81 ( http://www.insecure.org/nmap/ ) at 2006-03-25 21:18 EST
Interesting ports on <target IP address>.rmu.edu (k.m.n.104):
(The 1659 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
22/tcp  open  ssh
80/tcp  open  http
111/tcp open  rpcbind
113/tcp open  auth
MAC Address: AA:00:00:01:4A:A1 (Digital Equipment)
Device type: general purpose
Running: Linux 2.4.X|2.5.X
OS details: Linux 2.4.0 - 2.5.20
```