DESIGN OF NETWORK SECURITY PROJECTS USING HONEYPOTS

ABSTRACT

Honeypots are closely monitored decoys that are employed in a network to study the trail of hackers and to alert network administrators of a possible intrusion. Using honeypots provides a cost-effective solution to increase the security posture of an organization. Even though it is not a panacea for security breaches, it is useful as a tool for network forensics and intrusion detection. Nowadays, they are also being extensively used by the research community to study issues in network security, such as Internet worms, spam control, DoS attacks, etc. In this paper, we advocate the use of honeypots as an effective educational tool to study issues in network security. We support this claim by demonstrating a set of projects that we have carried out in a network, which we have deployed specifically for running distributed computer security projects. The design of our projects tackles the challenges in installing a honeypot in academic institution, by not intruding on the campus network while providing secure access to the Internet. In addition to a classification of honeypots, we present a framework for designing assignments/projects for network security courses. The three sample honeypot projects discussed in this paper are presented as examples of the framework.

1. INTRODUCTION

According to Bruce Schneier [1], "Security is a process, not a product." This famous quote is well echoed by the phenomenon that, although there exist umpteen numbers of security tools that are available today (either as commercial or open-source solutions), none of these tools can single-handedly address all of the security goals of an organization. The security professionals are thus looking for more advanced tools which are effective in detecting and recovering from security breaches. In order to monitor the activities of a hacker, the methodology adopted is to deceive them, by giving them some emulated set of services on a system which appears to be legitimate. Their activities are then logged and monitored to gain insight into the tactics of the blackhat community. This idea is adopted in *honeypots* - a system whose value lies in being probed, attacked or compromised.

Honeypots have received a lot of attention lately from the research community, owing to its use in capturing and logging suspicious networking activities, which can be used to gain valuable information about the behavior of hackers. Apart from its use as a research tool, it has also been deployed in educational institutions as a study tool. For example, the Honeynet Project at Georgia Institute of Technology has been used in network security classes in order to teach students how to use tools such as *ethereal* and *tcpdump* in order to analyze attack traffic [3]. However, the problem with deploying such a honeypot inside a campus network is twofold. First, the installation of honeypot is quite risky and must be carried out with utmost precision and care, so that the campus network is not intruded. Secondly, legal and ethical issues (such as the *Wiretap Act 18 U.S.C. 2511* [2]) must be taken care of before such a network is deployed.

The focus of this paper is not on the legal or ethical issues of deploying honeypots, but rather on using honeypot projects as a tool in teaching and studying network security issues. Our approach is to design some lab projects that demonstrate how honeypots projects may be used as effective tools in teaching network security classes. The design of our projects tackles the challenges in installing a honeypot in academic institution, by not intruding on the campus network while providing secure access to the Internet. The rest of this paper is organized as follows. In Section 2 we briefly discuss the requirements and challenges in designing network security projects. Section 3 gives a classification of honeypots and a few example honeypots we used in our projects. In section 4 we explain a framework for designing assignments/projects for network security courses, and in section 5, we explain a few of the projects we carried out. In Section 6 we provide a summary and work yet to be done in the future.

2. THE PROTOTYPE NETWORK FOR EXPERIMENTAL NETWORK SECURITY PROJECTS

As discussed earlier, the design of network security projects using tools such as honeypots, for use in an academic environment, is a challenging task. Experimental network security projects are typically considered as "dangerous" and not permitted in a University campus network. In order to perform such experiments, a dedicated security lab is typically necessary. We have deployed a prototype computer security network, as illustrated in Figure 1, on which experimental projects in network security could be implemented. Details of the design and our experiences of deploying the prototype network can be found in a separate paper [reference omitted for blind review].

The network is composed of the following devices and configurations:

(a) A DSL router, which connects the lab to the Internet via a DSL connection;

(b) A dual-homed software router/firewall (Pascal), acting as the gatekeeper between the network and the outside world;

(c) A regular switch connecting the DMZ (*Demilitarized Zone*) to the server cluster;

(d) A DMZ that contains publicly accessible servers (such as a Web server) and testing workstations (e.g., for monitoring network traffic, etc.);

(e) A 2nd router/firewall (Einstein) separating the DMZ from the server cluster;

(f) A hub or switch connecting the 2nd router/firewall with the back-end servers;

(g) A set of network security servers, including firewalls, VPN server, IAS (Microsoft's Internet Authentication Server), and Radius server;

(h) A test bed of computers, which are equipped with swappable disk units and are connected via a VLAN switch and routers.

Currently the prototype network consists of two routed-Virtual LANs, which are useful for simulating several network configurations. The honeypots experiments were performed using the testbed machines in VLAN and Newton, which is the test machine in the network.

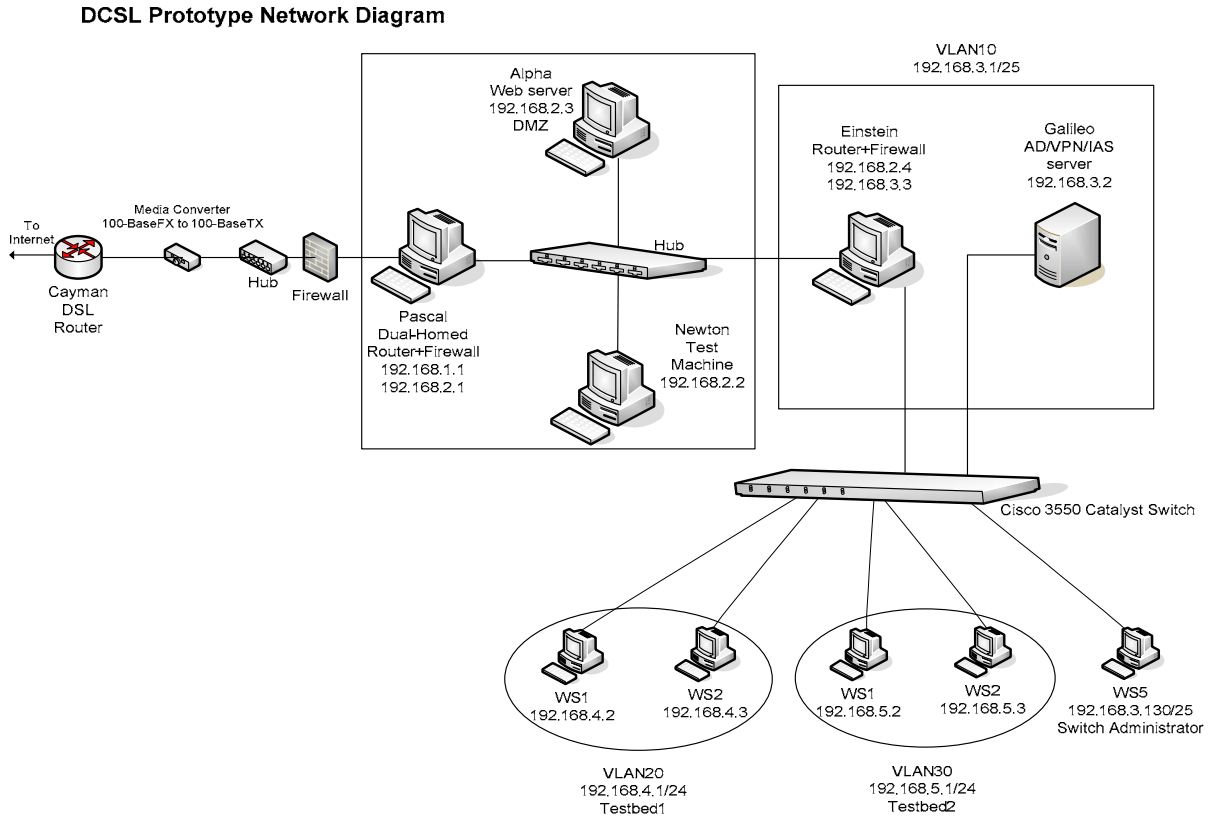

Figure 1. The prototype network

3. CLASSIFICATION AND EXAMPLES OF HONEYPOTS

Honeypots are divided into two general categories: production honeypots and research honeypots. *Production honeypots* add value to the security of a specific organization and help mitigate risk, and are typically implemented within an organization as they help in detecting attacks. Production honeypots are easier to build and deploy, because they require less functionality. This type of honeypots give less information about the attackers than research honeypots. *Research honeypots* are designed to gain information about the blackhat community. The primary goal is to research the threats that organizations may face, such as who the attackers are, how they are organized, and what kind of tools they use to attack other systems, etc. Research organizations such as universities and security research companies often use research honeypots.

Based on the level of interaction, honeypots are classified into three categories: low interaction, medium interaction and high interaction honeypots. *Low* interaction honeypots are primarily production honeypots that are used to help protect a specific organization [5]. A low interaction honeypot is easy to install and it emulates very few services. Attackers can only scan and connect to several ports. The information about the attackers and the risk is limited since the attacker's ability to interact with the honeypot is limited. An example of low interaction honeypots is 'Honeyd'. Honeyd is a small daemon that creates virtual hosts on a network and runs on both UNIX and windows platform [8]. The hosts can be configured to run arbitrary services, and their personality can be adapted so that they appear to be running certain operating systems.

Honeyd improves cyber security by providing mechanisms for threat detection and assessment. It also deters adversaries by hiding real systems in the middle of virtual systems. Honeyd works on the principle that when it receives a probe or connection for a system that does not exist, it assumes that the connection attempt is an attack. When honeyd receives such traffic, it assumes the IP address of the intended destination. It then starts the emulated service for the port on which it receives the connection. Once the emulated service is started, it interacts with the attacker and captures all of his activities. When the attacker is done, the emulated service exits. This process is repeated each time honeyd receives attacks.

In Honeyd, a virtual honeypot is configured with a template created in the Honeyd configuration file (honeyd.conf) that define the characteristics of a honeypot, including operating system type, the port they listen on, and the behavior of emulated services. Each template is given a name. New templates are created using *create* command. The *set* command assigns a personality from Nmap fingerprinting file to the template. The personality determines the network behavior of the given operating system that is simulated by Honeyd. The set command also defines the default behavior for network protocols: *block, reset* or *open. Block* indicates that all packets for the specified protocol are dropped by default. *Reset* means that ports are closed

by default. *Open* means that all ports are open by default. The *add* command is used to specify the services that are remotely accessible. The *bind* command assigns a template to an IP address. An example Honeyd configuration file is shown below:

```
create windows
set windows personality "Windows NT 4.0 Server SP5-SP6"
add windows tcp port 80 "perl scripts/iis-0.95/iisemul8.pl"
add windows tcp port 139 open
add windows tcp port 137 open
add windows udp port 137 open
set windows default tcp action reset
set windows default udp action reset
bind 192.168.5.102 windows
```

High interaction honeypots give vast amount of information about attackers, but they are extremely time-consuming to build and maintain, and they come with highest level of risk [5]. The primary goal of high interaction honeypot is to give the attacker access to a real operating system in which nothing is emulated or restricted. This provides more information about attackers. These types of honeypots are placed within a controlled environment such as behind a firewall. Because of the control mechanisms, high interaction honeypots can be difficult and time-consuming to install and configure. An example of high interaction honeypots is 'Honeynets'. What makes a Honeynet different from most honeypots is that it is an entire network of systems. Instead of a single computer, a Honeynet is a network of systems designed for attackers to interact with. The honeypots within the Honeynet can be any type of system, service, or information. They are simply a network that contains one or more honeypots.

Medium interaction honeypots offer attackers more ability to interact than low interaction honeypots but less functionality than high interaction honeypots [5]. This type of honeypots can expect certain activity and are designed to give certain responses which are more than what a low interaction honeypot would give. It requires more time to install and configure than low interaction honeypots.

4. A FRAMEWORK FOR DESIGNING NETWORKING PROJECTS

We have designed a standard framework for representing computer security projects. The use of a framework provides an infrastructure for designing, implementing and evaluating projects, and consists of the following components:

a) Learning Objectives: The goals for the assignment and the learning opportunities.

b) Tools utilized: The tools necessary to implement the particular project. For example, a project on *Installation and Configuration of Honeyd* requires tools such as *arpd* and *Honeyd*.

c) Requirements: The formal description of the problem.

d) Problem classification: The assignment can be classified based on the approach taken to solve the problem. Some of the issues to be considered:

(i) Will it be a study experiment?

(ii) Will it involve programming?

e) How it may be implemented in the security lab: This is one of the main issues, and is concerned with the infrastructure necessary for the particular project. Some of the questions that need to be addressed are:

(i) Will the assignment require a change of the network topology?

(ii) What kind of resources will be provided to the students to implement it?

f) Level of difficulty: The level of difficulty for the project (Beginner, Intermediate, or Advanced).

g) Grading criteria and methods: The basis for grading the projects and the methods employed by the teaching assistant and/or the instructor to grade them.

5. LAB PROJECTS

In this section we discuss three of the honeypot projects that we have designed. To aid in easy adoption and evaluation of the projects, all the projects are represented using the standard framework as discussed earlier. The first project involves installation and configuration of *honeyd*. The second project is related to network node discovery using *nmap*. The third project involves virtual honeynets.

5.1 Project 1: Installation and Configuration of *Honeyd*

a) Learning Objective:

Imagine that you have been asked to administer a network. As a good network administrator, it is important for you to protect the network from being attacked by hackers. This experiment gives you hands-on experience on installation and configuration of virtual honeypot network using *honeyd*, which acts as a decoy and protects your network from attackers. *Honeyd* provides mechanism for threat detection and assessment. It also deters adversaries by hiding real systems in the middle of virtual systems.

b) Tools utilized:

(i) **Honeyd** is a small daemon that creates virtual hosts on a network [8]. The hosts can be configured to run arbitrary services, and their personality can be adapted so that they appear to be running certain operating systems. Some of the features available in Honeyd are as follows: Simulation of large network topologies; Configurable network characteristics like latency, loss and bandwidth; Integrate physical machines into network topology; Supports multiple entry routers to serve multiple networks.

(ii) **Arpd** is a daemon that listens to ARP requests and answers for IP addresses that are unallocated. Arpd is used in conjunction with Honeyd, to ensure that honeyd host responds to the arp request for the IP's of the honeypots. Arpd responds with the MAC address of the honeyd host for any request to an unused IP address.

c) Requirements:

This experiment requires you to set up a virtual honeypot network with the following virtual IP addresses:

- Honeypot A (192.168.2.102) must simulate an IIS web server
- Honeypot B (192.168.2.103), D (192.168.2.105), E(192.168.2.106) must support TCP services like telnet, ftp running on Linux 2.3.12
- Honeypot C (192.168.2.104) must simulate POP3 mail server
- Cisco router (192.168.2.5) inside virtual honeypot network must simulate a telnet server.

Scripts for simulation of different services can be downloaded from http://www.honeyd.org/contrib.php. Information on how to set up the configuration file *honeyd.conf* can be found at http://www.honeyd.org/configuration.php.

d) Problem classification: This experiment can be classified as a network assignment and also as a study experiment.

e) How it may be implemented in the security lab? This experiment requires manipulating IP tables. Students can be assigned a workstation along with the tools (Arpd & Honeyd) required for this experiment in zipped archive.

f) Level of difficulty: Based on the level of difficulty, this experiment can be classified as an experiment for beginners.

g) Grading criteria and methods:

The grader may test the virtual IPs in the virtual honeypot network using *ping* and *traceroute* utilities. The grader can also grade this experiment by testing the service configured on each of the honeypots. For example, if 192.168.2.5 is configured to simulate a Cisco router, then it must support telnet service, so TA can try to telnet 192.168.2.5 to check if it simulates telnet service.

5.2 Project 2: Network Node Discovery

a) Learning Objective:

As a network administrator, it's your responsibility to find and fix the vulnerabilities of the services and OS that are running on servers and workstations. We know that often OS and services running on them have some weakness and bugs, hackers can take advantage of it to gain control of the system. Traditional tools such as *traceroute* and *tcpdump* do not detect hosts that are not communicating. For example, a web server does not create any traffic unless it receives requests from others. Even some of the services running on hosts might be silent. To detect such hosts and services, we need advanced tool like *nmap*. In this project, you will use *nmap* (Network Mapper) to discover hosts and services running on it.

This project requires student to work in groups of 2. One student sets up a virtual honeypot network that simulates different operating systems and with some services running on it. The other student uses an *nmap* tool to discover the hosts and services running on its ports.

b) Tools utilized:

(i) **Honeyd** is a small daemon that creates virtual hosts on a network [8]. The hosts can be configured to run arbitrary services, and their personality can be adapted so that they appear to be running certain operating systems.

(ii) **Arpd** is a daemon that listens to ARP requests and answers for IP addresses that are unallocated. Arpd is used in conjunction with Honeyd, to ensure that the honeyd host responds to the *arp* request for the IPs of the honeypots. Arpd responds with the MAC address of the honeyd host for any request to an unused IP address.

(iii) **Nmap** is a free open source utility for exploring networks and performing security auditing. It was designed to rapidly scan large networks. Nmap uses raw IP packets to determine what hosts are available on the network, what services (application name and version) those hosts are running, what operating systems (and OS versions) they are running, what type of packet filters/firewalls are in use, and dozens of other characteristics. Nmap runs on most types of computers, and both console and graphical versions are available. Nmap is free software, available with full source code.

c) Requirements:

Part A: (i) What is the purpose of the *nmap* utility? (ii) List what you can do with nmap and how?

Part B: As mentioned in the learning objectives, this experiment requires students to work in a group of two. One student sets up the virtual honeypot network in the workstation assigned, which simulates different OS and services running on it. The virtual honeypot network must have the following honeypots:

Honeypot A must simulate an IIS web server.
Honeypot B, D, E must support TCP services like telnet and ftp.
Honeypot C must simulate POP3 mail server.

The Cisco router inside virtual honeypot network must simulate a telnet server.

Another student must install the *nmap* tool in the workstation assigned, and use it to find out the hosts and services running on virtual honeypot network set up by your group partner. Use nmap tool to find out the following:

(i) Find out all the hosts (OS type) running in the virtual honeypot network
(ii) Find all the services running on the hosts that are discovered.

Scripts for simulation of different services can be downloaded from http://www.honeyd.org/contrib.php. More information on how to set up the configuration file *honeyd.conf* can be found at http://www.honeyd.org/configuration.php.

The students must learn how to install **nmap** and **honeyd** on Linux machines. Submit a report of your findings along with the snapshots.

d) Problem classification: This experiment can be classified as a network assignment and also as a study experiment.

e) How it may be implemented in our security lab? This experiment requires manipulating IP tables. Students can be assigned a workstation along with the tools (nmap, arpd, and honeyd) required for this experiment in zipped archive.

f) Level of difficulty: Based on the level of difficulty, this experiment can be classified as an experiment for beginners.

g) Grading criteria and methods:

The grader can grade this experiment by testing the virtual IPs in the virtual honeypot network using *ping* and *traceroute* utilities. The grader can also grade this experiment by testing the service configured on each of the honeypots. For example, if 192.168.2.5 is configured to simulate a Cisco router, then it must support telnet service, so TA can try to telnet 192.168.2.5 to check if it simulates telnet service. The TA can also use *nmap* tool from the test bed to check if nmap would return correct results.

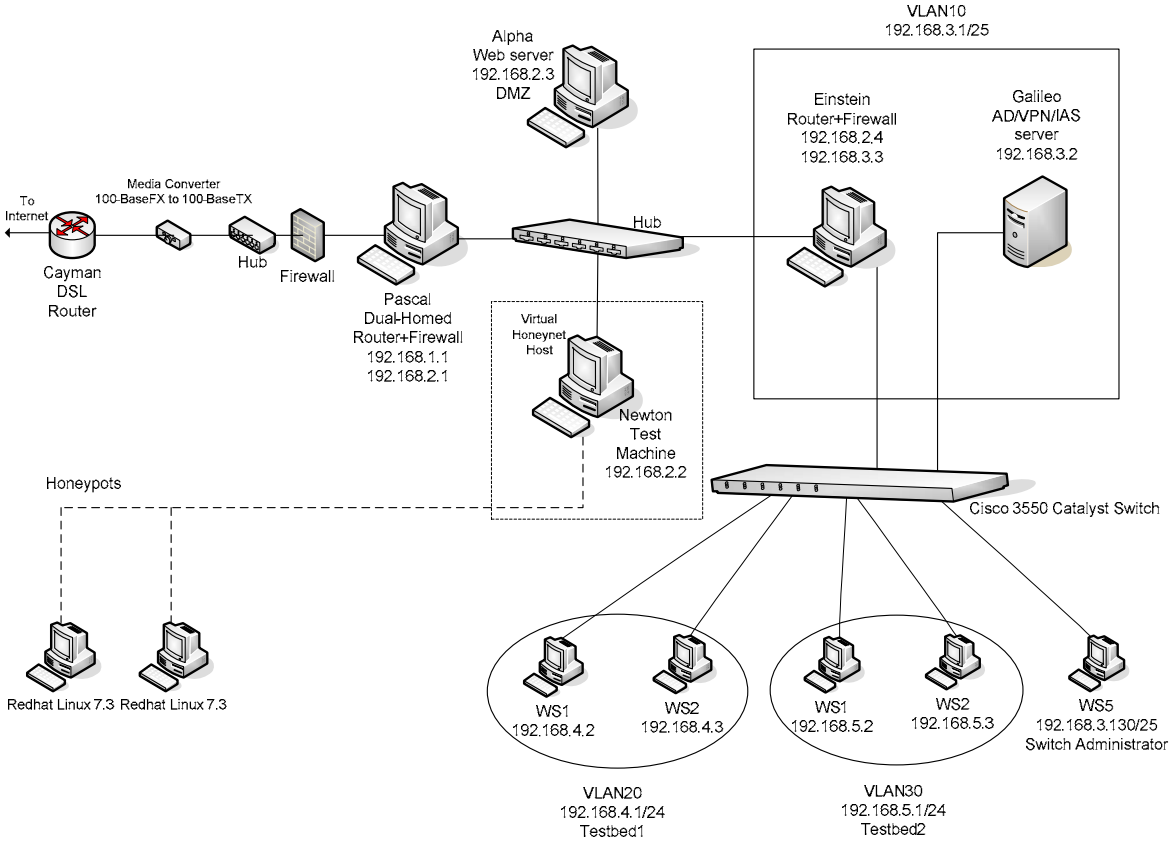

Figure 2. Implementing the virtual honeynet in the prototype network

5.3 Project 3: Virtual Honeynets

Honeynets require a variety of both physical systems and security mechanisms to effectively deploy. The Honeynet project has been researching a new possibility known as virtual Honeynets [3]. These virtual Honeynets provide the advantage of running all the systems on a single system, thereby making Honeynets cheaper to build, easier to deploy and simpler to maintain. Virtual Honeynets is not a new technology but uses the technology of Honeynet and combines them into a single system. It uses virtualization software which allows running multiple operating systems at the same time on same machine.

In this project, virtual Honeynet is implemented using a User Mode Linux, which is special kernel module that allows you to run many virtual versions of Linux at the same time on the same system. The main reasons for choosing User Mode Linux over VMware include the following: (a) Free and open source; (b) Fewer resource requirements; (c) Supports bridging and networking like VMware; (d) User Mode Linux is available with pre-configured and downloadable file systems, which makes it easy to configure Honeypots inside the Honeynet. The main limitation of User Mode Linux is that it currently supports only Linux virtual machines, but the support for windows is under development.

In Figure 2, a virtual Honeynet is implemented on Newton (192.168.2.2), and User Mode Linux patch is applied to the Linux running on Newton, thereby allowing running multiple instances of Linux on the same system. The virtual Honeynet host (Newton) provides Data Control to control what the attacker can do, by allowing all inbound traffic to Honeynet but limiting the outbound connections. For this purpose, an OpenSource firewall solution *IPTables* is used. The virtual Honeynet host also provides data capture which captures all the attackers' activities without their knowledge. For Data capture we use Open Source IDS *Snort*.

6. SUMMARY AND FUTURE WORK

This paper presents an overview of a prototype computer security lab and design of network security projects using Honeypots. The design of lab exercises for Network Security lab is a challenging issue, and this paper provides a framework for designing computer security projects. The framework describes the issues that must be considered at the time of designing projects for Computer Security labs, and may be considered as a starting point by computer science educators wishing to design Computer Security projects. The first two sample projects described in this paper were implemented and tested on the prototype network, and the third one (virtual honeynet) is currently being implemented.

We found honeypots to be a good tool for students to learn about Networking Security and the blackhat community. As part of our future work, we plan to design more network projects using honeypots. Once implemented, the virtual Honeynet will initially be closed to the Internet due to concerns of ethical and legal issues. We plan to eventually make the virtual Honeynet open to the Internet world, in order to collect research data about the blackhat community. The collected data and their subsequent analysis will help us to protect our network from attackers. We also are considering implementing a real Honeynet. These projects are challenging since using honeypots to protect a network is a new field, and re-creating a problem in the lab environment may not always be possible.

ACKNOWLEDGEMENT

Omitted for blind review

REFERENCES

[1] Schneier, B. *Secrets and Lies: Digital Security in a Networked World*, MA: John Wiley & Sons. 2000.

[2] Computer Crime and Intellectual Property Section (CCIPS), *18 U.S.C. 2511*, http://www.cybercrime.gov/usc2511.htm

[3] Honeynet Project, http://www.honeynet.org

[4] Teo, L., Sun, Y., Ahn, G. Defeating Internet Attacks Using Risk Awareness and Active Honeypots, *Proceedings of the Second IEEE International Information Assurance Workshop (IWIA'04)*, 2004.

[5] Spitzner, L. *Honeypots Tracking Hackers*, MA: Addison-Wesley, 2002.

[6] Provos, N. Honeyd: A Virtual Honeypot Daemon, *Technical Report*, Center for Information Technology Integration, University of Michigan.

[7] Provos, N. A Virtual Honeypot Framework, *USENIX Security Symposium*, August 2004.

[8] Provos, N. Developments of the Honeyd Virtual Honeypot, http://www.honeyd.org

[9] Nmap, http://www.insecure.org/nmap

[10] Weiler, N. Honeypots for Distributed Denial of Service Attacks, *Proceedings of the IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'02)*, 2002.

[11] Baumann, R. Honeyd - A Low Involvement Honeypot in Action, published as part of GCIA practical.

[12] Chandran, R., Pakala, S. Simulating Network with Honeyd, *Technical paper*, Paladiaon Networks, December 2003.