Dynamic Honeypot Construction

Amanjot Kaur

Assistant Professor S.D.S.P.M. College for Women, (Rayya), Amritsar, India
amanjotkbhullar@yahoo.co.in

ABSTRACT

System security personnel fight a seemingly unending battle to secure their digital assets against an ever-increasing onslaught of attacks. Honeypots provide a valuable tool to collect information about the behaviors of attackers in order to design and implement better defenses, but most current configurations are static setups consisting of either low interaction or high-interaction environments. Although static honeypots help address this issue, the ability to construct dynamic honeypots easily would enable security personnel to identify potential security vulnerabilities in the attempt to build better defenses. This research effort describes a method to automatically and dynamically configure honeypots based on the results of network scans. These dynamically constructed honeypots then emulate a system or network of systems in order to collect information to better protect that network. This dynamic honeypots configuration methods has been implemented and tested, and can now enhance the ability of system administrators to identify system vulnerabilities.

KEYWORDS: Honeypots, static honeypots, dynamic honeypots

1. INTRODUCTION

In 2004, there was an estimated $17.5 billion in computer damage ranging from cash and identity theft to the overt destruction of computers and secure computer networks. This rising threat requires that corporations protect their assets and those of their customers from an ever increasing danger that cannot be stopped through traditional methods. It is important for organizations to secure their digital assets by finding means to detect and prevent vulnerabilities before they exploited. One step towards this goal is through the development and application of a honeypot, which is information system resource whose value lies in unauthorized or illicit use of that resource", allowing such undesirable behavior to be monitored. This implies that a honeypot has no purpose except to capture attacks or packets sent to the machine. At the simplest level, there are two general categories for honeypots, low-interaction and high interaction. The interaction defines the level and extent of the associated emulation, i.e., the level of activity between the honeypot and an attacker.

2. STATIC HONEYPOTS

Whether low-interaction or high-interaction, most honeypots are traditionally static. Although they may evolve over time to better emulate a current system, (e.g., apply a patch or service pack to a component), they are generally configured manually. They remain static throughout their implementation life cycle regardless of the evolution of the networks that they model.

2.1 Low-interaction Honeypots

As the name implies, low-interaction honeypots are limited in the extent of their interaction with the attacker. They are generally emulators of services and operating systems, and the attacker's activity is limited by the level of emulation provided by the honeypot. The emulation process could be as simple as listening on port 21 and recording any attempts to access that port. The complexity, or level of interaction, could be increased by emulating a basic FTP server which responds appropriately to login attempts, and maybe even file transfer requests. By increasing the level of interaction the honeypot appears more realistic to an attacker, allowing for more complex attack behavior to be monitored, and decreasing the likelihood of the attacker detecting that the system is a honeypot. The deployment and maintenance of such systems is fairly simple and does not involve much risk. However, there would be maintenance and evolution required in order to keep up with the constantly changing nature of the systems connected to the network. In addition, the emulated services mitigate risk by containing the attacker's activity. Unfortunately low-interaction systems generally log only limited information. From an attacker's perspective, a low-interaction honeypot can be easily detected simply by executing a command that the emulator does not support.

2.2 High-interaction Honeypots

High-interaction honeypots are a far more complex solution and typically involve the deployment of real operating systems and applications. The major advantage associated with a high-interaction honeypot is the capture of extensive amounts of information. By allowing the attackers to interact with real systems, the full extent of their behavior can bestudied and recorded. A high interaction honeypot may be a physical system, but can also be implemented as a virtual machine and in order to gather data which is most relevant they typically involve the use of the same operating systems and services that are found in the organization deploying the honeypot. As a result, an organization that utilizes
Windows workstations and Linux servers would likely deploy Windows and Linux-based honeypots. In order to actually capture the behavior of the attacker in such environments, some additional monitoring system must also be present, such as a packet sniffer at the network level or, more commonly, as an application on the honeypot host itself, which is the approach used by the Sebek data capture tool.

2.3 Benefits

Honeypots, whether they are high or low interaction, allow the administrators to detect anomalies that might otherwise go unnoticed. Since most production machines have a large amount of network traffic, consisting of both legitimate and unwanted connections, it is difficult for a system administrator to exhaustively examine the logs for anomalous behaviors. Since honeypots have no production value, any interaction within the machine is likely to be either pre-attack scanning, or a potential attack. This allows the system administrator to target and analyze the logs created by the honeypot system without having to determine which traffic is suspicious and which is legitimate, as they would when analyzing logs from production systems. When large honeynets (multiple networked honeypots) are used in a production network environment, the system administrator can compare the logs from the various honeypots and see if a scan was a result of an accidental encounter or a more deliberate probe or attack, in which case the logs might show that several ports were scanned on multiple honeypots during a brief period of time. This information can then be researched to determine if the scan came from outside the network. This small piece of information can help to ascertain whether machines inside the network have been compromised or if there is a vulnerability in the network defenses that has allowed a potential intruder into a secured area. The use of highinteraction honeypots allows the system administrator to capture the tools used for the attack or the potential code from a new virus before it strikes the production environment. Even low-interaction honeypots provide the system administrator with an improved identification of the types of network traffic on the system and can help isolate potential anomalies for further investigation.

3. DYNAMIC HONEYPOTS

While there are several methods to implement honeypots , "one of the biggest challenges we face with most security technologies, including honeypots, is configuring them". This research effort addresses that issue by providing a methodology to automatically configure a honeynet that emulates a selected network. The Honeyd Configuration Manager tool produced as part of this project scans an identified network and allows the honeypot administrator to choose associated parameters to control the creation of a honeynet that emulates the network to the level desired by the system administrator.

3.1 Background

In July 2004, Iyad Kuwatly, Malek Sraj, Zaid Al Masri, and Hassan Artail wrote a paper called "A Dynamic Honeypot Design for Intrusion Detection.". In this paper they proposed a design for a dynamic honeypot system. This design consists of a combination of both high and low interaction honeypots. In December 2004, Jeffery Hieb and James H. Graham, in their paper entitled "Anomaly-Based Intrusion Detection for Network Monitoring Using a Dynamic Honeypot" used a passive fingerprinting tool p0f [KMM04] to monitor the network, thereby determining active systems and open ports. This information is recorded in a database then periodically a virtual honeynet (multiple networked honeypots) is created using Honeyd to represent the production machines. While these approaches are interesting, they both rely on passively scanning network traffic, and as a result can only generate honeypots that mimic services that communicate during the scanning period. In contrast, active scanning, which is the method described in this paper, results in honeypots which simulate all open ports on a system, even if there is no traffic to those ports during the scanning process.

3.2 Purpose

Most computer networks are a diverse collection of computers and computer systems especially in academic and professional settings. There are frequently a wide variety of operating systems for workstations and servers and each of these have different network ports open to accomplish their individual duties. Many networks must also accommodate the need for individuals to take their work home or to connect remotely from an external environment. They then reconnect to the network when they get to work the next day. These actions change the structure of the network as the attached devices evolve and change through time, some predictably, and others less so. Effective and authentic emulation of these changes require that the honeynet system dynamically (by use of real-time or periodic execution of an update process) set up a virtual network to mimic the real network to the maximum possible extent.

3.3 Benefits

The use of dynamically configured honeypots or honeynets provides several advantages over static configurations:

1. A system administrator or researcher can quickly build a honeypot without the need to have indepth knowledge of the honeypot configuration mechanism, which in the case of Honeyd takes the form of a text configuration file. The dynamically constructed configuration can then either be used to deploy a honeypot, or as a basis for refinement by the honeynet administrator.
2. The honeynet administrator is not required to know the details of the network topology or installed systems in the network to be mimicked, as this information can be automatically ascertained.
3. The honeynet configuration can be easily reconfigured to reflect the current network topology. This reconfiguration can either be performed on a predefined schedule, asynchronously in response to some event such as the assignment of a new IP address by a DHCP server, or even in near-real time if desired. This resulting honeynet can then be deployed on a different subnet for testing purposes, or interwoven into the production systems in the same network to provide information on attacks against the production network. In addition to simply identifying operating systems and open ports, the honeynet configuration can be automatically augmented with scripts to emulate services at a higher level of interaction, allowing the Honeyd virtual machines to respond to an attacker in a manner similar to a production machine.

3.4 Specifications

The Honeyd Configuration Manager system was implemented in Perl using the Linux OS, as shown in figure 1. The system actively scans the network using Nmap for OS detection and to determine which TCP/UDP ports which are open. While it does consume some bandwidth, active scanning quickly provides sufficient data from which a honeypot or honeynet configuration file can be constructed. The alternative approach of using passive scanning, such as that performed by pf, only provides information about ports that are communicating, and as a result gathers information about the network and active hosts more slowly. Passive scanning also has limitations in the desired.

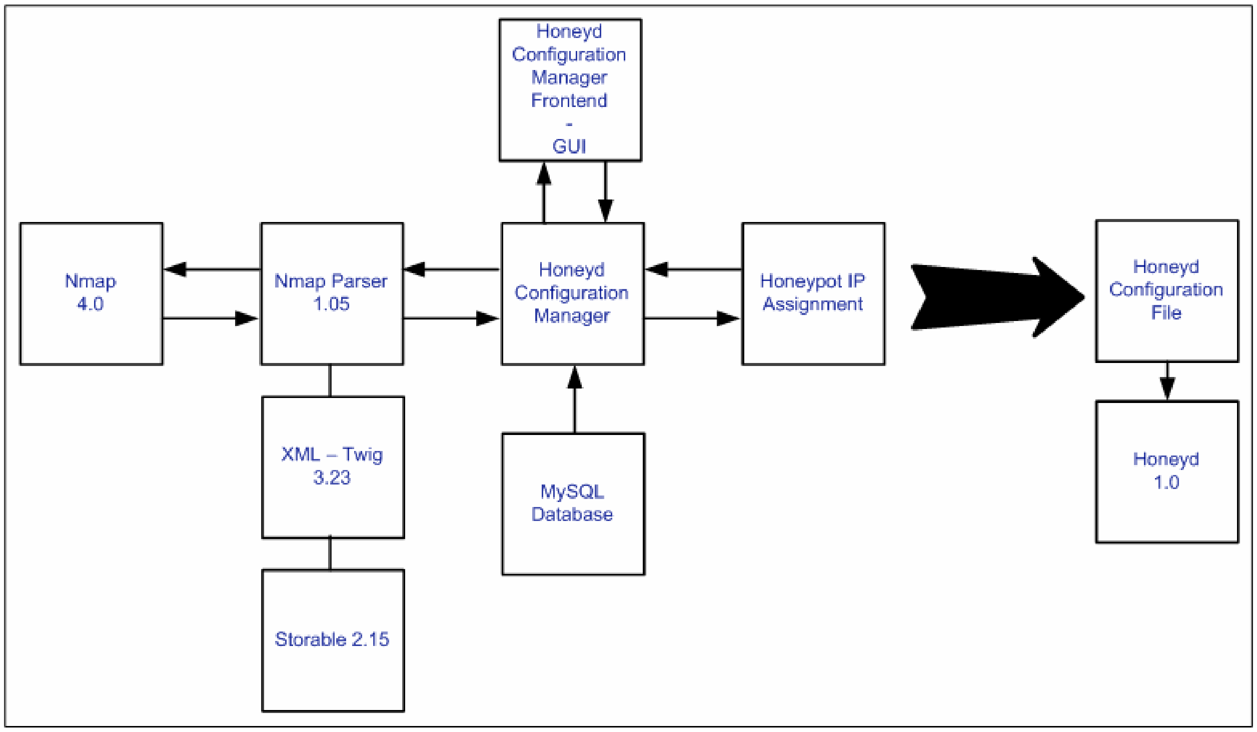


Figure1:Honeyd Configuration Manager Components network topology which it can monitor, including issues related to Layer 2 switches and routers. Passive scanning requires the detection of packets on the network to determine the network configuration by looking at the packet headers. It is necessary that the scanner intercept all packets for a more complete picture. This is made more challenging with the use of Layer 2 switches and routers which, for security and efficiency, isolate the different devices. Routers can also confuse the passive scanner by modifying the packet headers. The default Nmap scan used by the Honeyd Configuration Manager is a SYN stealth scan, RPC scan, UDP scan, and OS detection, as shown in the following Nmap command:

nmap -sS -sU -sR –O <port range>

Other scan options are available in Nmap, and can be selected for the Honeyd Configuration Manager using either the GUI or the CLI.

The Nmap results are analyzed real-time by the Honeyd Configuration Manager to determine the resulting honeynet configuration, which is stored as a Honeyd configuration file. As the Honeyd file is created the program uses the IP assignment method selected by the administrator to determine the network configuration of the honeypots. These IP assignments allow the administrator to create replicas of the scanned machines and networks in a manner most appropriate to the task, such as research into attack behavior or the deployment of honeypots in a production environment. The four options for IP address assignment are as follows:

1. Configure the honeypots to use the same IP addresses as the real systems. For example, if a real system is found with the IP address 192.168.0.7, a system will be added to the Honeyd configuration file with the same address.

This option should not be used to create a honeynet which will exist on the same network as the production systems as it will result in IP address conflicts. However, it can be a useful starting point for creating a Honeyd configuration file prior to manual modification, or if the honeynet will be used when the production systems are isolated from the network or powered down (overnight, for example in some circumstances).

2. Modify the network component of the IP address, while preserving the host component. For example, if a host with IP address of 192.168.0.12 is found while scanning a class C network, the network component can be mapped to another subnet, such as 172.16.0.0/24, resulting in a system in the Honeyd configuration for use outside of the

production network, such as for testing in a laboratory or other controlled environment

3. Use a designated IP address range in which to place the resulting honeypots. For example, if the user designates 172.16.0.20-40 as the honeypot IP address range, then the first production host found, such as 192.168.0.25, would result in a system in the Honeyd configuration file with the IP address 172.16.0.20. Honeypot representations of additional production hosts identified in the scan would continue to receive IP addresses in the assigned range until it was completely populated.

4. Interweave the honeypot systems into the production network where possible, although this requires that the assigned IP addresses on a given production network can be identified. If the scanning system is on the same subnet as the systems being scanned then unassigned IP address can be identified by whether or not there is a response to an ARP packet. In the event that the scanning system is not on the same subnet as the systems being scanned other approaches can be used to identify unassigned IP address, including pings or reverse DNS lookups, although these are likely to provide inaccurate results so the resulting Honeyd configuration file should be used with caution. Regardless of the method used to identify the unassigned IP addresses, if new systems are added to the production network it is possible that an IP address conflict may result, depending on the method used to assign address to new systems. This approach is most useful when building a honeynet configuration for use in conjunction with existing production systems, such as in a commercial environment. Once the production systems, with their operating systems and open ports, have been identified, the Honeyd configuration file is created and used to start, or restart, the honeynet. The reconfiguration of the honeynet using this method can be invoked manually by the honeynet administrator, or configured to execute automatically on a periodic schedule or in response to an identified about the available scripts, including which services they emulate, and for what operating systems they are appropriate. Using this approach, when an open port is found by the Honeyd Configuration Manager is possible to quickly query the database for an appropriate script. For each script, the database includes information about the operating system(s) and port(s) for which it is appropriate, to ensure, for example, that a script which emulates the Microsoft IIS 5.0 Web Server does not appear on a Linux based honeypot. If a suitable script is found, it can then be automatically added to the Honeyd configuration to provide a more in-depth emulation (i.e., higher interaction level) on the particular port. The MySQL database used for testing contained six emulation scripts, but this database-based framework allows for addition scripts to be utilized by simply adding new records to the database.

Figure 2: A basic Honeyd configuration file created dynamically by the Honeyd Configuration Manager.

```
##### Honeyd Configuration File ######
##### Sun Apr 16 20:04:09 2006 #####
########################################
create Windows1
set Windows1 personality "Microsoft
Windows 2003 Server or XP SP2"
set Windows1 default tcp action reset
set Windows1 default udp action reset
set Windows1 default icmp action open
add Windows1 tcp port 21 open
add Windows1 tcp port 23 "perl
/Honeypot/Honeyd_scripts/telnet/
faketelnet.pl"
add Windows1 tcp port 25 open
add Windows1 tcp port 80 open
add Windows1 tcp port 110 open
add Windows1 tcp port 143 open
add Windows1 tcp port 143 open
add Windows1 udp port 123 open
add Windows1 udp port 135 open
add Windows1 udp port 137 open
add Windows1 udp port 138 open
add Windows1 udp port 139 open
add Windows1 udp port 445 open
add Windows1 udp port 500 open
add Windows1 udp port 1900 open
add Windows1 udp port 4500 open
add Windows1 udp port 31337 open
set Windows1 ethernet "Intel Corporate"
bind 192.168.192.103 Windows1
########################################
```

Figure 2: A basic example of a Honeyd configuration file created by the Honeyd Configuration Manager is shown in figure 2. This honeynet consists of a single honeypot, which is a

Microsoft Windows based system, with open several ports, and a telnet emulation script listening on port 23. Typical Honeyd configuration files for production networks are substantially larger, and commonly feature honeypots using several different operating systems, open ports, and emulated services.

asynchronous event, such as the assignment of a new DHCP address, which may suggest that a new system has been added to the environment. In addition to providing a configuration which emulates the production hosts and open ports, it is possible to add scripts to Honeyd which can provide a more in-depth emulation of a given service. For this research effort, a small MySQL database was used to store the configuration file with IP address 172.16.0.12. This would be useful when creating a honeynet details.

3.5 *GRAPHICAL USER INTERFACE*

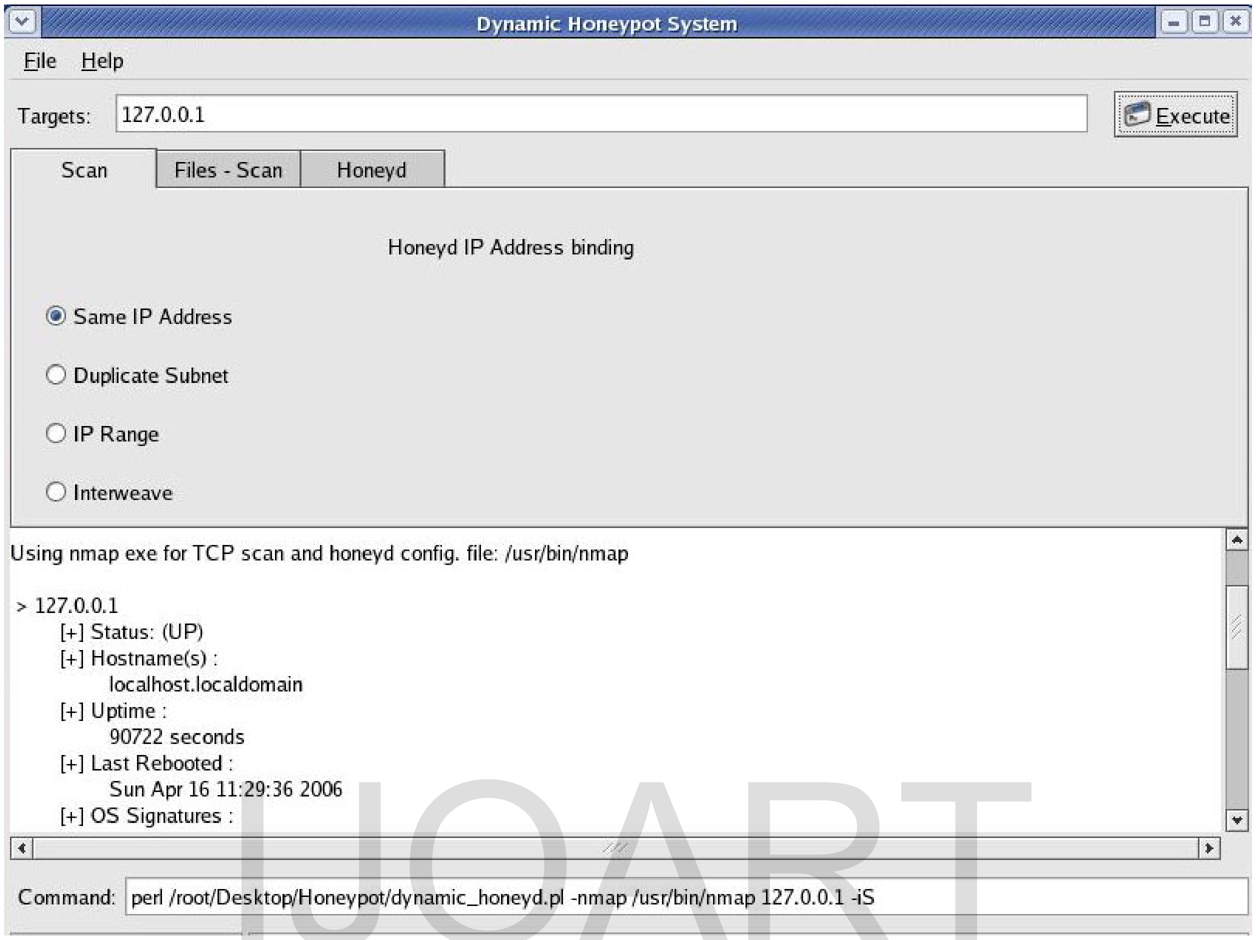
A Graphical User Interface (GUI) was also created as a front-end for the command line based Honeyd Configuration Manager. This allows the user to easilycontrol the behavior of the Honeyd Configuration Manager, including the IP assignment mechanism, Nmap scan parameters, and whether service emulation scripts should automatically be used. In addition, the GUI allows the user to monitor the results of the scanning process, and to select whether the tool should merely create a Honeyd configuration file, or should also start a Honeyd instance at the completion of the configuration process. The GUI was implemented using Glade User Interface Builder for GTK+ and GNOME. Figure 3 shows an example of the GUI in use

4. CONCLUSION

While honeypots can be a valuable tool in both research and production environments, they are typically not easy to configure, particularly if the goal is to mimic a real network environment. This paper describes a methodology for dynamically building honeypot and honeynet configurations based on active network scanning to determine network topology, attached hosts,operating systems, open ports, and accessible services for a given target network. This research effort resulted in the implementation and testing of a tool, the Honeyd Configuration Manager, which applies that methodology to allow system administrators and researchers to construct honeynets based on real networks quickly and easily.

5. FUTURE CONSIDERATIONS

There are several avenues in which this research will be expanded: incorporate both passive and active scanning, the use of low and high interaction honeypots on the same system, and increased honeypot emulation abilities. A greater understanding of the network can be gained by the use of active scanning to determine an initial configuration, while applying passive scanning to either dynamically modify the configuration, or as an asynchronous trigger for additional active scanning. Integrating both low and high interaction honeypots by initiating first contact with the low interaction honeypots then directing the more interesting traffic to a high interaction honeypot if additional functionality is required. Additional research into improved emulation of services to further expand the capabilities of the dynamic honeypots is underway and new test scenarios are continually being constructed. This increased research will give system administrators a much needed edge in the continual tug-of-war game they play with attackers.



Selected: [✓ Same IP Address]

```
Using nmap exe for TCP scan and honeyd config. file: /usr/bin/nmap

> 127.0.0.1
      [+] Status: (UP)
      [+] Hostname(s) :
            localhost.localdomain
      [+] Uptime :
            90722 seconds
      [+] Last Rebooted :
            Sun Apr 16 11:29:36 2006
      [+] OS Signatures :
```

Command: perl /root/Desktop/Honeypot/dynamic_honeyd.pl -nmap /usr/bin/nmap 127.0.0.1 -iS

Figure 3: The Honed Configuration Manager GUI

6. REFERENCES

[1] Grow, Brian. *Hacker Hunters*, Retrieved from http://www.businessweek.com/magazine/content/05_22/b3935001_mz001.htm.
[2] *Know Your Enemy: Learning with VMware. Building Virtual Honeynets using VMware*. Retrieved April 18, 2006 from http://www.honeynet.org/papers/vmware/
[3] *Honeyd Homepage*. Retrieved February 15, 2005 from http://honeyd.org/contrib.php.
[4] *Honeynet Project. Know Your Enemy: Learning about Security Threats*. Pearson Education, Inc., Boston, MA, second edition, 2004.
[5] *Insecure.org Homepage*. Retrieved February 15, 2006 from http://www.insecure.org/.
[6] Iyad Kuwatly, Malek Sraj, Zaid Al Masri, and Hassan Artail. *A Dynamic Honeypot Design for Intrusion Detection*, American U. of Beirut. 2004.
[7] Hieb, Jeffery and Graham, James H. *Anomaly-Based Intrusion Detection for Network Monitoring Using a Dynamic Honeypot*. Intelligent System Research Laboratory. Dec. 2004
[8] Provos, Niels. *Honeyd - Network Rhapsody for You*. Retrieved February 15 from http://www.honeyd.org
[9] *MySQL Homepage*. Retrieved February 15, 2006 from http://www.mysql.com/.
[10] Quynh, Nguyen Anh. *Xebek: A next generation honeypot monitoring system*. EUSecWest/core06, London, U.K., Feb 2006.
[11] Sebek Homepage. *The Honeynet Project*. Retrieved February 15, 2006 from http://honeynet.org/tools/sebek/.
[12] Spitzer, Lance. *Honeypots Tracking Hackers*, Addison-Wesley, Boston, 2003.