Global Information Assurance Certification Paper

Honey in the Pot or Tar in the Pit:
A Case Study of a Honey Pot and LeBrea Tar Pit

Gregory E. Kane

GSEC Practical Assignment V1.4
Option 2

ABSTRACT:

The following paper will discuss the configuration and implementation of a honeypot system, Phase I. The honeypot system was selected because of the potential usage and impact on a live LAN in a production environment. It is believed that a honeypot could greatly enhance the overall shield in a layered defensive system. The system was installed and incorporated into an existing LAN. The honeypot worked somewhat as expected, though there were some unexpected problems encountered and noted in this paper. The system was hacked on at least one occasion prior to installation of Phase II. Phase II includes using Labrea Tar Pit on the basic honeypot.

Introduction:

Generally, honeypots are used to observe and track the tactics used by the blackhat community through recording a blackhat's attack on a particular system environment[1]. Honeypots, as well as honeynets[2] versus our developing Honeypot Array, have proven to be a valuable source for the collection of information relative to the tools the blackhat community is using[3]. Honeypots can also aide system administrators as an early warning management tool against blackhat activity on their system. The topic of honeynets will not be included in this paper. Discussed in the paper is the approach taken for each type of tool used in the honeypot, problems discovered in installing and using the particular tool and the results of the implementation. An overall review of the honeypot system is also included. Additionally, a review of LeBrea Tar Pit[4] (Sticky Honeypot) will be addressed,

1 Honeypots: Definition and Value of Honeypots, Spitzner, Lance, March 2002.
2 The HoneyNet Project: http://project.honeynet.org
3 Klug, David. "Honey Pots and Intrusion Detection", September 13, 2000
4 LaBrea Tar Pit: http://www.hackbusters.net/LaBrea/

including the installation of LeBrea and results observed and obtained. As a note, an additional product honeyd[5], which is similar to Labrea Tar Pit is available, but will not be used or reviewed. The prime focus of this paper will be the efficacy of the system reviewed and the usefulness of the tools used.

The Honeypot- Phase I

The Basic System Configuration

The Honeypot used for this paper had been installed using a default installation of Mandrake 8.1[6]. The exception to the default was adding the Apache Web Server and ProFTPD. Additionally, Sendmail was installed and used in favor of the standard install of PostFix. Mandrake was chosen based upon the fact that the overall lab, and potential production system, in which the test took place has standardized on the Mandrake platform.

The honeypot was placed in the DMZ off a cable router (Figure 1). The reason for using a default installation and placing the honeypot in the DMZ was to give the computer the look of an ordinary, nothing special platform, that could be used by a potential hacker for his or her own desires. As noted in general literature many installed systems in homes and businesses are not hardened, or placed behind a well configured firewall, thus giving the intruder an easy platform picture. The specific IP address of the honeypot was not advertised in any form, relying on scans for discovery.

The honeypot was initially added to the network, and placed in the DMZ, on April 20, 2002 at 1400 hrs. The initial plan was to keep the honeypot on-line for one week. Compromise of the system was expected rather early on, which proved to be true. The focus was to be on what tools were used to compromise the system, and whether the installed honeypot tools were responsive to intrusions, rather than just how quickly the honeypot was compromised. After the completion of the honeypot testing, Labrea Tar Pit was added to the platform and kept on-line for a week. Labrea is discussed later in this paper. The basic specific computer configuration was as follows:

[5] Honeyd: http://www.citi.umich.edu/u/provos/honeyd

[6] Mandrake Linux: http://www.linux-mandrake.com/en/features.php3
NOTE: Additional packages were needed to load most of the tools selected for the honeypot. Where necessary, these additional packages are noted.

Intel based system board
333 Mhz processor
96 Megs of mainboard memory
3.2 gig Samsung Hard Drive

The Data Collection Tools Used

The selection of tools was based on the tool's availability in rpm (RedHat Package Manager) format, and the lessen need for a large amount of additional dependency packages for the tool to function. Most rpm's, unless otherwise indicated, were obtained at http://rpmfind.net. In all, over 10 tools were examined with Scanlogd, Logwatch, Tripwire, Chkrootkit, and Snort, being chosen. Where cron scripts were not available with the tool, a script was written to gather information from each tool and to forward the collected information to an email account. Each tool was scheduled at different times. Syslog was scheduled to send data every 18 minutes, with the addition of saving the log to a floppy disk. Tripwire and logwatch were scheduled every hour. Chkrootkit ran an email report every 12 minutes. It was believed that varying the times would not set a pattern for the intruder to potentially follow.

SCANLOGD RPM-2.2-1.5

OVERVIEW:

Scanlogd is a small footprint TCP port scan detection tool originally developed for a Phrack magazine article.[7] Scanlogd was chosen because of the tool's small foot print and what seemed to be an excellent tool to send port scan notifications, yet remain rather stealthy. The response email would then act as an early warning system as outside scans began.

When detecting a port scan, scanlogd writes one line through the syslog system. The manpage for scanlogd stipulates, " Logging is done with a facility of deamon and a priority level of alert. In /etc/syslog.conf you may using something like:

daemon.alert /var/log/alert "

[7] Solar Designer, 1998. "Designing and Attacking Port Scan Detection Tools" and http://phrack.org/show.php?p=53&a=13p

INSTALLATION:

Installation was rather straight forward for an rpm package. As was the normal case with Linux, additional software was needed to be loaded to solve dependency issues of the scanlogd rpm on the default Mandrake 8.1 installation. According to information on the web site for the tool[8], "This release of scanlogd can be built with support for one of several packet capture interfaces. In addition to the raw socket interface on Linux, scanlogd is now aware of libnids and libpcap." The two additional packages were downloaded and installed for this project. A script was written to send the alert log to a designated email account every 30 minutes.

OBSERVATION AND RESULTS:

Scanlogd was extremely easy to install and configure. When the alert log arrived via email, in most cases the email was blank, that is until April 25th when a major attack of the system commenced.

Scanlogd performed flawlessly in recording all probes and the final attack, to be discussed later in this paper, on the honeypot. Scanlogd actually recorded scans that were not recorded elsewhere. How ever, scanlogd did not list the ftp attempts which was probably because of a configuration problem. Based upon performance and ease of installation scanlogd was an excellent addition to the honeypot.

CHKROOTKIT RPM-0.35-3

OVERVIEW:

Chkrootkit[9] is another small footprint program that constantly monitors the system for an introduction of a rootkit signature. The selection of chkrootkit was to allow the monitoring of the potential of a rootkit being installed on the system. The program is designed, as noted on the author's website, to check for 30 of the more common rootkits such as:

[8] Scanlogd: http://openwall.com/scanlog/d/
[9] Chkrootkit: http://www.chkrootkit.org

1. lrk3, 4, 5 and 6
2. Solaris rootkit
3. FreeBSD rootkit
4. tOrn
5. Ambient's Rootkit for Linux (ARK)
6. Ramen Worm

The program checks for many more possible rootkits. More information is available at the author's website.

The chkrootkit program uses the following commands:

Awk, cut, echo, egrep, find, head, id, ls, netstat, ps, strings, sed, and uname.

The one possible problem with these commands is that they can be replaced by a rootkit, and the above is discussed on the developers website. Regardless of this possible problem, chkrootkit was installed and used.

INSTALLATION:

The installation rpm required no additional packages. A script was written that would email the results to a designated email account. A cron job was scheduled to run with the script every 12 minutes. The selection of 12 minutes was with the realization that a blackhat installing a root kit was a real possibility. As the honeypot was being remotely monitored, the reporting scheduling would give enough warning time to allow further investigation.

OBSERVATIONS AND RESULTS:

The reports that were received were detailed. All through the experiment, chkrootkit did not report any installation of a rootkit. However, many of the logs for the period of the attack were lost because the system was made inoperable by the attack. There was an instance in another log that indicated that the attacker did install something, but chkrootkit's report is missing for that period. It can only be assumed that chrootkit did respond appropriately.

TRIPWIRE RPM-2.3.1.2

OVERVIEW:

The selection of Tripwire as a file monitor was based on Tripwire's standing within the general industry as the defacto standard. Tripwire is well discussed throughout available literature and is available as a rpm package or tar ball. The version of Tripwire used for this project was from the Tripwire Open Source Project[10] rpm package. According to the Tripwire Open Source Project FAQ, "Tripwire is a tool that checks to see what has changed on your system. The program monitors key attributes of files that should not change, including binary signature, size, expected change of size, etc. The hard part is doing it the right way, balancing security, maintenance, and functionality[11]."

INSTALLATION:

The rpm for Tripwire did not require a multitude of additional packages to satisfy dependencies. The rpm was obtained from http://rpmfind.net. Installation using the rpm was simplistic. That is where simplicity ends. Configuration of Tripwire took some time. After installation of the rpm, it is required to run the installation shell script. This shell scripts creates two important files, the twpol.txt (policy file) and the twcfg.txt (configuration file). The twpol.txt is a default policy file based on RedHat 7.1. Depending on the flavor and version of Linux used, the policy file will need modification. Part of the configuration is running a comparison of the boxe's file systems against the default policy file. The resulting comparison gives an indication of the changes needed in the twpol.txt. Again, depending on the Linux flavor used, this modification can take quite sometime and is not for the faint hearted. Once the policy file is correctly edited, Tripwire has a process to encode the plain text file so that it is generally unreadable. The plain text twpol.txt can then be removed from the system. Tripwire does provide a utility to unencode the file back to plain text for editing. Finalization is straight forward based on the directions provided in the documentation. A clearly written alternate to the general Tripwire documentation for installation is also available and proved

10. The Tripwire Opens Source Project: http://www.tripwire.org/
11. The Tripwire Open Source Project: http://www.tripwire.org/qanda/faq.php

invaluable[12]. A shell script was written to send the reports to an established email account every hour.

OBERVATIONS AND RESULTS:

As expected Tripwire worked well. One aspect of tripwire is that it will report ALL file changes. This means that if the policy file is not configured tightly, the user will start receiving multiple file changes. Watch out for the log files, as these files change frequently and will report back through Tripwire. There were no false positives noted during the period of the project. During the period of this project Tripwire did work well, however the Tripwire email for the period of the attack was never received. The reason was that sendmail was rendered inoperable by the time the report was due out. The Tripwire report on the test box was not recovered for the period of the attack because of the test box becoming inoperable and the data was unrecoverable.

SNORT RPM-1.85 and Windows Version 1.7

OVERVIEW:

The following is taken directly from the Snort webpage[13].

> "Snort is a lightweight network intrusion detection system, capable of performing real-time traffic analysis and packet logging on IP networks. It can perform protocol analysis, content searching/matching and can be used to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes, OS fingerprinting attempts, and much more. Snort uses a flexible rules language to describe traffic that it should collect or pass, as well as a detection engine that utilizes a modular plugin architecture. Snort has a real-time alerting capability as well, incorporating alerting mechanisms for syslog, a user specified file, a UNIX socket, or WinPopup

[12] http://www.linuxsecurity.com/feature_stories/feature_story-81.html
[13] SNORT: http://www.snort.org/about.html

messages to Windows clients using Samba's smbclient."

Snort is a real workhorse when it comes to monitoring a system. Though Snort has IDS capabilities, the program was chosen to deliver accurate packet logging. In addition to having Snort running on the test box, a Windows version[14] of Snort was running on another box, on the inside of the network.

INSTALLATION:

Both installs were very easy and only required the addition of libpcap to the default Mandrake 8.1 install, and the Windows version. Snort was set to log to the box it was installed on.

OBSERVATIONS AND RESULTS:

Few packets were able to be removed from the inoperative honeypot. Including snort in the honeypot package appears to be an excellent choice. Snort did capture many, many packets, however these were lost due to system crash. Snort requires setting file size limits and use remote logging. Keeping the logs off of the honeypot would preclude the serious problem encountered with this project. The few logs that were able to be recovered indicated some sort of an attack that

The Attack – Lessons Learned

The honeypot began receiving minor probes and ftp attempts within hours of being placed into the DMZ on the 20th of April. These minor probes and snoops began to increase in frequency, mostly at night, and for only a few milliseconds at a time. Until the 25th, the day the attack occurred, probes and snoops were no more than 5 in a 24 hour period at first and up to 9 the night prior to the attack. A rather curious situation occurred in that the night of the 24th saw no scans or probes. Scans and probes picked back up after 0100 on the 25th. It was the quite before the storm. The attack began on April 25th at 10:25:33 as reported by scanlogd. The initial scanning from the first intruder stopped at 11:11:20. At 12:40:34 another IP appears in the scanlogd report and lasts until 12:42:05. At that point scanlogd reports another IP beginning scans at 12:45:41 and continued until

[14] Snort for Windows: http://www.silicondefense.com/software/index.htm

13:18:59. Scanlogd stopped recording after that time. The stopped response occurred because of the server being flooded and the logs filling to maximum size possible. One lesson learned was all logs should have a maximum size set. As the logs filled from the flooding, basically a denial of service attack (DOS), the system slowly stopped responding. Additionally, the small size of the hard drive did not leave enough room for the system to maintain logging operation. However, the filling of the logs did not crash the server. Sendmail logs were filled with the error messages. In retrospect, in a prudent measure, the server should have been configured to send all logs to a remote log server. However, other than the Win98 system running Snort, additional computer equipment was not available at the time. The Win98 system was protected by a firewall and virus protection. The firewall logs and the Snort logs all show the initial attack as the Win98 machine was also attacked.

The test server was being monitored remotely, via email reports with no one on-site to respond. At 13:30:01 the message log shows that eth0 was set in promiscuous mode and the log shows eth0 leaving promiscuous mode at 13:32:50. Nearly a full three minutes of running in promiscuous mode! It is believed that a rootkit was installed during the earlier attacks of that morning. There was nothing pre-installed on the test server that would open the network card to promiscuous mode and then close it. Somewhere in the logs there must be an indication of a previously installed rootkit or compromise, but currently that indication has not been found. The systems sendmail stopped at 10:49:13 for no apparent reason. However, system logs were still actively recording. The message log was pulled off the test server at 17:08:13 over 6 hours after the system stopped sending out the email reports.

After pulling the message log, syslog and alert logs from the test server, the server was disconnected from the Internet at 17:17:00. Approximately 30 seconds after disconnecting from the Internet the system froze solid. No response what so ever. A cold reboot of the system failed. The emergency boot disk had no effect in bringing the system up. An attempt was made to boot from the original media for the system installation. That did not work but produced an error message that hda1 could not be found. At this point the remaining logs were lost and the first phase of the project was terminated. Additional

forensics of the hard drive may reveal added material but is beyond the scope of this project.

An attempt was made to follow the IP's for the three main intrusion attempts. One led to a University in Sweden, which is totally untraceable and the two others led to computer security companies in the United States. The two companies in question have been contacted. Discussions have begun with management of both companies. Almost beyond belief, one of the company's management finally admitted that their servers are from time to time being hacked and the challenges the company has. This incident has helped two computer security companies take another look at their own security.

An ironic situation occurred during the 25th after 12:00:00. The email service that was receiving the email reports went down for over 6 hours. When the email service returned to operation, all email for the account was lost for those 6 hours. This adds to the lessons learned. When sending reports via email, have a stable email server provider.

What is known for sure is that an attack occurred beginning at 10:25:33 on the 25th of April. Something was downloaded to the test server by one of three intruders, which probably caused the system freeze up and the hard drive error after disconnecting from the Internet. Based on the given evidence, the last intruder probably delivered the malicious payload, which somehow terminated the server when the server was disconnected from the Internet. During the initial intrusion attempts the first intruder attempted to flood the server with packets for a Denial Of Service. The total packets received in less than two minutes were over 1.2 gigs in total size. The sniffer logs were not recovered but provided an initial log folder size count. During the attacks on the test server, which was in the DMZ, the Win98 platform was also attacked but the attacks appeared to be stopped by the firewall. The Win98 system tested clean.

The tools on the server, for the most part, lived up to their billing. They did what they said they would. The hardest tool to install was Tripwire. There are some questions about chkrootkit as there was probably a rootkit installed by an intruder and chrootkit did not report it. However, that does not mean that chkrootkit did not do its job. It may have very well been that the rootkit used was not in chkrootkit's signature configuration. Another tool that functions like chkrootkit may

have caught it. However, that data is lost and will not be recovered. This is where the project faltered. The potential for a rootkit that is not defined to be used. Then for a general system failure, loosing logs that could possibly identify the payload(s), was a blow to the overall success of this portion of the project. The tools on the server worked in most cases, but the system failed. To potentially stabilize the log failures in the future a much larger hard drive, in the range of 20 gigs, will be used. Additionally, an alternative or adjunctive tool will be located for chkrootkit

LeBrea Tar Pit RPM 2.3-1 – Phase II

OVERVIEW:

LaBrea was chosen for Phase II of the project to have intruders captured and "tar pitted" for analysis. According to the program author, "Labrea works as a low-level network application that creates "virtual machines" on your network – machines that don't really exist yet are able to answer connection attempts in a special way that slows and even stops the connecting process[15]." Labrea can react in two specific ways, which were the reasons that this software was chosen. Labrea can tar pit and send a notice to syslog and message logs. The notices indicate the originating ip of the intruder. Secondly, in becoming a "sticky honeypot" Labrea can hold that connection. Labrea was added to the same basic system as used in Phase I with the exception that chkrootkit was not installed. The decision to keep chkrootkit off the test box was made because the initial indications in Phase I that the current version of chkrootkit may not be as effective as was needed. All log reports were emailed to a designated email address. LaBrea has a multitude of switches that can be added to increase functionality. Additionally, sniffing was limited to the Win98 box.

INSTALLATION:

The Labrea rpm had only one unfulfilled dependency, which was libnet. Libnet was downloaded from the general rpm source noted earlier, and installed. In order to find an up-to-date rpm package, Labrea was downloaded from http://www.stearns.org/labrea/. Installation was not

[15] Labrea Tar Pit: http://www.hackbusters.net/LaBrea/LaBrea.txt

notable. Labrea ran out of the box with the switches as indicated at the Labrea home page.[16]

OBSERVATIONS AND RESULTS – DURING/AFTER

During the period of the test, May 2–7, Labrea did manage to tarpit one scan. There appeared to be no other scans on the virtual machines that LaBrea set up. Another point was that the LaBrea program set up these virtual machines with the same identified services. Further study and configuration of LaBrea is needed to allow the setting of different services for a range of ip's. The remainder of the tools on the test box worked well. There were multiple scans against the test box that were noted by the tools and logs. LaBrea is not designed to use active ip's for tarpitting so none were noted. What did seem somewhat out of place was that while the test box was scanned numerous times, only one virtual tarpit was scanned. This lack of scanning of the tarpitted machines is unknown and deserves further investigation.

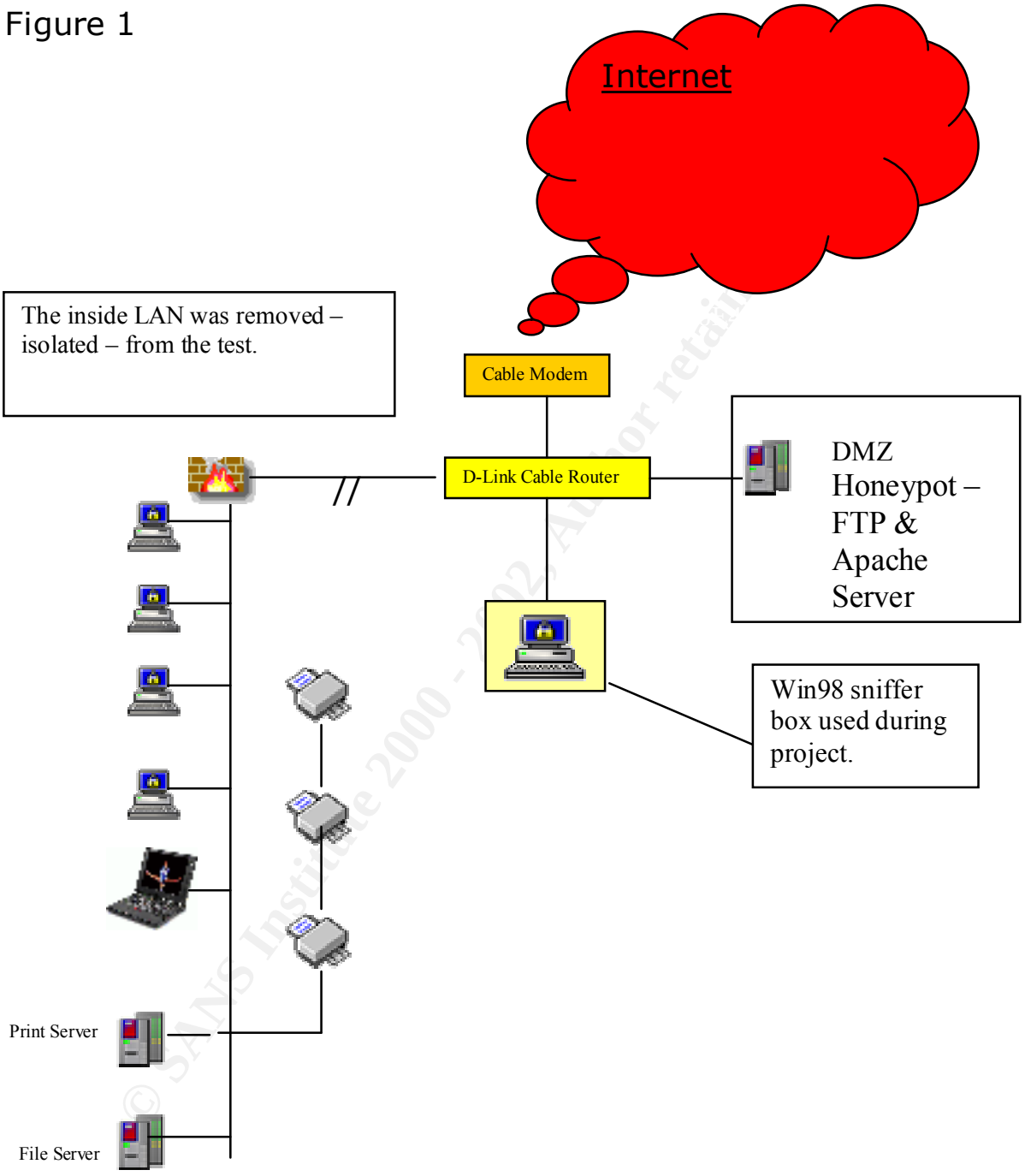
CONCLUSION – IMPACT:

Overall this project was a success. The project did capture data showing that an intrusion did occur. What it was not able to do was identify the tool or tools used to compromise the system. The failure to identify the problem and be reasonably sure how the intruder was able to compromise the system[17] was a shortcoming in this project. A separate logging computer needed to be used along with the separate computer for packet sniffing. The packet sniffing box was a Windows 98 machine using Snort for Windows. With the vulnerabilities of Windows 98 being well known the data could not necessarily be trusted. A hardened Linux box will be used in the future. The use of a small hard drive on the test box did not provide ample room for sniffer logs. There were logs from Win98 sniffer box, but these proved inconclusive. The tools selected did perform as stipulated by the individual software authors. An addition that could have been used was a keystroke logger. The logger would have given greater insight into the attack. Another observation is that Mandrake Linux 8.1 while being more secure than previous additions is still lacking a built in secure kernel. An overall more secure kernel for Linux, not an add-on package, needs to be developed for all Linux distributions. Further investigation on the use of the tools in this project, especially

16. Labrea Homepage: http://www.hackbusters.net/LaBrea
17. The Honeynet Project, Know Your Enemy, New York, Addison-Wesly, .2002. 9.

chkrootkit, and other potential tools needs to be continued. The outcome of this project will be the development and implementation of a "HoneyPot Array." The HoneyPot Array will allow continued development of tools necessary to keep pace with the blackhat community.

Figure 1



REFERENCES

1. Spitzner, Lance. "Honeypots: Definition and Value of Honeypots", March 2002.

2. The HoneyNet Project: http://project.honeynet.org

3. Klug, David. "Honey Pots and Intrusion Detection", September 13, 2000

4. LaBrea Tar Pit: http://www.hackbusters.net/LaBrea

5. Honeyd: http://www.citi.umich.edu/u/provos/honeyd

6. Mandrake Linux: http://linux-mandrake.com/en/features.php3

7. Solar Designer, "Designing and Attacking Port Scan Detection Tools". 1998. http://phrack.org/show.php?p=53&a=13

8. Scanlogd: http://openwall.com/scanlogd

9. Chkrootkit: http://www.chkrootkit.org

10. The Tripwire Open Source Project: http://www.tripwire.org

11. The Tripwire Open Source Project: http://www.tripwire.org/ganda/faq.php

12. http://www.linuxsecurity.com/feature_stories/feature_story-81.html

13. SNORT: http://www.snort.org/about.html

14. Snort for Windows: http://www.silicondefense.com/software/index.htm

15. LaBrea Tar Pit: http://www.hackbusters.net/LaBrea/LaBrea.txt

16. LaBrea Homepage: http://www.hackbusters.net/LaBrea

17.The Honeynet Project, Know Your Enemy, Boston, Addison-Wesly, 2002. 9.