Global Information Assurance Certification Paper

Building a Cookerpot

Using honeypots to improve Mandrake Linux security

Valter Santos
February 5, 2003

GIAC GSEC version 1.4b

Abstract

The Cookerpot Project is a new effort that aims to collect and to analyze data related with threats that Mandrake Linux users suffer from the Internet. To gain such knowledge are used honeypots with Mandrake Linux development version, the Cooker, as operating system. The main objective is to examine and critique Mandrake in order to establish a more secure future Mandrake Linux release. As secundary objective, the project will also provide a deeper knowledge about blackhat community habits and tactics, and honeypots deployment in general.

This paper describes the installation and configuration process of the honeypot I will use to give my contribution to the project, as well how data control and analysis will be done to produce conclusions about Mandrake Linux security. Lately, it will state how this effort can be used to help Cooker's development process to provide a more secure distribution to the end user.

Introduction

Simple put, honeypots are systems used to study blackhat methodologies. These systems exist to be proven and potentially rooted by attackers that shouldn't know that the system they are attacking is a fake one. The goal is to have a deeper knowledge about habits, tactics, tools and exploits that the blackhat community is using.

Although part of this paper is related with honeypot configuration in a general stand point, it's main purpose isn't to give definitions about Honeypot concepts, so I will point the reader to other papers at the SANS Reading Room that provide such information very well: *Hands on the Honeypot*, by Kecia Gubbels [11] and *The use of Honeypots and Packet Sniffers for Intrusion Detection*, by Michael Sink [12].

Honeypots using Mandrake Cooker as operating system are named Cookerpots by the Cookerpot Project [u1]. In this context the term "cookerpot" will be used through the text with the same meaning as the term "honeypot" itself.

The main idea is to use Mandrake's development version, the Cooker, in machines exposed to the Internet to analyze and to prevent potential dangers and insecure configuration issues that could be released on the next stable release.

This is an unusual idea, to use honeypot technology to help the ongoing software development process, patching potential danger issues. Although, the project itself is very new and, for this reason without proven results, I'm confident that the lessons that will be taken from it will be very valuable for the security of an default Mandrake installation and for the security community in general.

In the next sections I will describe the installation and configuration of my own cookerpot, some details about it integration within my personal network, used for lab testing and research, and how potential attacks coming from the cookerpot will not be allowed to damage real services on the network. Then I will focus on the Cooker installation and patching for honeypot purposes. It will be also described how data control and monitoring will be done and how the analysis of such data will lead to conclusions that can help Cookerpot Project's main objective: have a better and more secure Mandrake release.

I hope that the following text could be useful to other individuals who want to configure their own cookerpots in the particular case of the Cookerpot Project, or honeypots in a general point of view.

This is only an example of a cookerpot deployment, others should follow other methods in order to obtain different results.

Setting up the network

This cookerpot will live in an existent network that is running "true" services such as mail and http. It is therefore important to safeguard the rest of the network from the cookerpot machine to prevent that malicious actions could affect these services if the cookerpot became compromised.

The existent network (network A) is connected to the Internet through a cable connection with a DMZ containing servers running email and http services (DMZ_A). The cable provider allows for only one dynamic IP address per internet connection which is used by the services running at DMZ_A. This limits my ability to use this IP address for the cookerpot.

At my disposal is a second network (network B) in a different location, which is mainly used for standard Internet client use (browse the web, read email, etc.). This network's public IP would prove suitable for use with my cookerpot setup but its geographical location would be problematic.

To solve this, I have linked both networks with a VPN using an IPSEC encrypted tunnel through the Internet, this allows machines from network A to be able to see machines from network B using their private IPs as if they are all in the same physical network.

Since VPN configuration is out of the scope of this paper, let's only suppose that everything is already up and running with the tunnel between the networks.

Both networks are using dynamic DNS services to have full domain name resolution.

However a little complex, this kind of setup will keep multiple layers of defense. I also have tried to implement a GenII Honeynet, even so my current setup is only a single honeypot for now, I looked to follow the standards for such honeynet [6] for future setup improvements (add more cookerpots, perhaps).

The connection between machines in the network is done through hubs. I choose it instead of using switches for the single fact that it is easier to do packet sniffing in an hub-based connection than in an switched one.

Figure 1 represents a general view of the network currently in use:

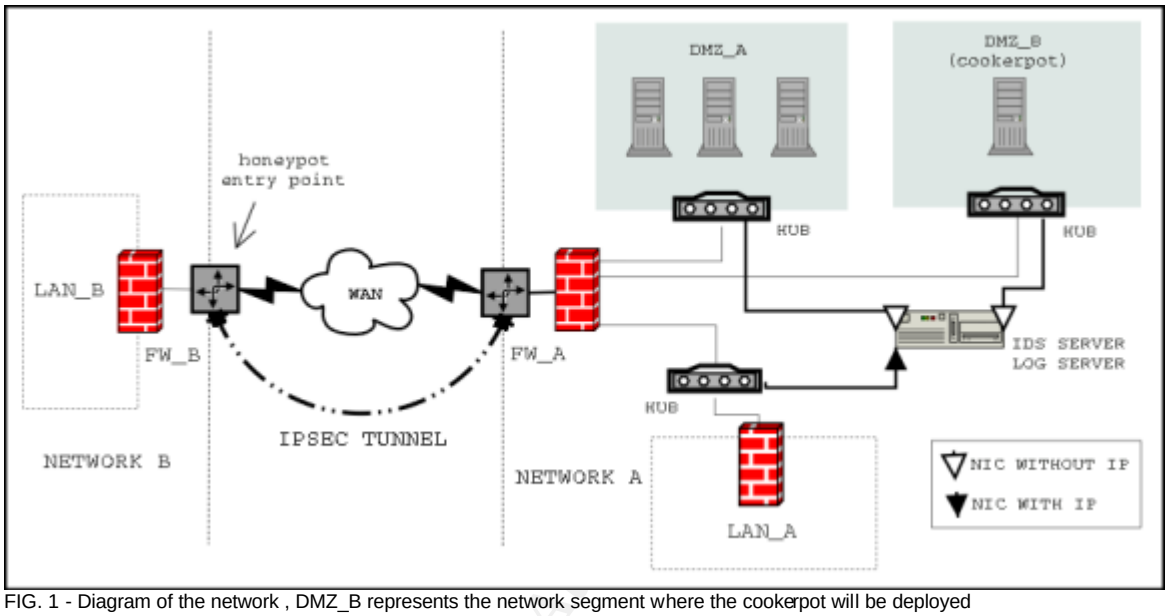

FIG. 1 - Diagram of the network , DMZ_B represents the network segment where the cookerpot will be deployed

What will happen is that when an incoming request hits the router at network B, FW_B forwards it via the IPSEC tunnel to FW_A at network A, that in its turn, forwards the request to the cookerpot at DMZ_B. The inverse should also be true: all traffic coming from the cookerpot should leave the network with network's B public IP, this will keep script kids out of legitimate services at DMZ_A.

Selected incoming traffic hitting FW_B is always routed to the cookerpot machine. The main firewall at network A is configured to be very restrictive to outgoing traffic coming from DMZ_B. Connections between DMZ_B and DMZ_A are not allowed, neither are connections between DMZ_B and both LANs at network A and network B. Connections from DMZ_B to FW_A are denied, and only proxyed http access is allowed with bandwidth limitation. This kind of policy should prevent the cookerpot to be used as jump point to attack other machines on the Internet.

However, this limitation may change in the future to let attackers to have more flexibility and creativity and to allow me to analyze how they operate more closed. But for now, what it's important is to find existing vulnerabilities in the Cooker, so let's limit the intruder actions.

Stealthful Logging

In the diagram above, there is a machine that is crucial to the network and to the purpose of monitoring the honeypot. The Log and Intrusion Detection server has the important task of to analyze and to log all the traffic in the network and actions in all the machines in the DMZs, this allows to keep an eye on intruders and to take action if needed.

The configuration was thought to allow me to keep many layers of defense and logging. Although all the machines, including the cookerpot itself, will log to local media, they will also log everything to the log server, which will keep logs and traffic information in a central place, away from the attacker's fingers.

The machine acting as log server has three network cards (NICs), one connected to each network segment. The ones connected to both DMZs are configured without IP information and will operate in stealth mode, this will prevent connections from any machine in both DMZs to the log server.

Having NICs without IP information connected from the log server to both DMZs makes a lot of sense, however the log server receives data from the cookerpot or any other machine in those DMZs, it will never need to send any data back. Taking advantage from that we can protect the most important machine in the network from be potentially tampered by intruders, since they are unable to connect to it in the first place. The last NIC is connected to the LAN segment with IP information, this will allow the log server to be managed from within the LAN.

Given that, one important question arises. Since the log server is connected to the DMZ segment in an ipless manner, how the cookerpot machine will send the logs to it?

This all idea was taken from Mick Bauer's article on Stealthful Intrusion Detection and Logging [8].

This uses Snort [u4], an open source Network Intrusion Detection System, to act as stealth logger, and only will use syslog in machines sending logs to the server. Later in the text you will see that I have two Snort instances running in the log/ids server, one used for stealth logging and other used as standard network intrusion detection sensor. For now, let's focus on the one used for logging.

The idea is to have all machines sending log information to a bogus IP, i.e. an IP in that network segment that is not used by any real machine, and have snort in the log server sniffing and catching all UDP traffic to this IP on port 514, used by syslogd.

Let's suppose that DMZ_B network address is 192.168.5.0/24 and our bogus IP is 192.168.5.11.

The cookerpot machine needs to have syslog configured to send logs to a fake remote syslogd. For instance its `/etc/syslog.conf` file should look something like this:

```
*.info @192.168.5.11
```

With this, the cookerpot will send all info logs to our bogus machine with IP 192.168.5.11. This is a small and effective example for the syslog.conf file. Remember that logging is very cheap nowadays, so log everything you can, specially if you are configuring a honeypot, be free to extend your syslog configuration.

Let's now see how snort will catch this info on the wire and log it to the log server.

To configure snort to act as stealth logger, I run it with the following config file `/etc/snort/snort_logger.conf`:

```
var EXTERNAL_NET any
config dump_payload
config dump_chars_only
config logdir: /var/log/snort
preprocessor frag2

log udp 192.168.5.0/24 any -> 192.168.5.11/32 514 (logto:"logged-
packets";)
```

This tell snort to catch all the traffic sent to 192.168.5.11 from network 192.168.5.0/24 and log it to file /var/log/snort/logged-packets with full payload and no hexadecimal chars (only chars). I have another identical rules to catch syslog data from other network segments, but this example shows the idea very well. For a better understanding on snort rules please check Snort Users Manual [u5].

This will provide a better logging mechanism that could be used with the entire network to keep all syslog data from all systems in a central machine best hardened than all the others. Having a central log server is very important since allows us to follow better forensic procedures after an attack, keeping log data intact also allows to do a better crossing reference of all attackers moves which will provide a more deepely understanding of all steps of the attack.

Network Intrusion Detection

Additionally to the logging mechanism described above, I also have configured Snort to run as Network Intrusion Detection System (NIDS) [u4].

In the reality, I have chosen Demarc PureSecure as my NIDS [u6]. PureSecure is a solution based on Snort, although it's commercial, Demarc has a personal version available for free. I choose this for convenience, and because they really make the job easier with their web based console, that let me to keep a glance in what is happen in my network before I start digging through log files. This really makes my job easier. It's also written in Perl, allowing me to make small changes to fit my needs.

PureSecure uses MySQL as database to keep all events that occured in the network, and has an alert system and other great features. PureSecure installation is easy and I will not cover it here. An analog configuration is to use Snort with MySQL and Acid [u7].

One thing that should be stated is how are the NIDS sensors distributed through the network. Figure 2 shows that.

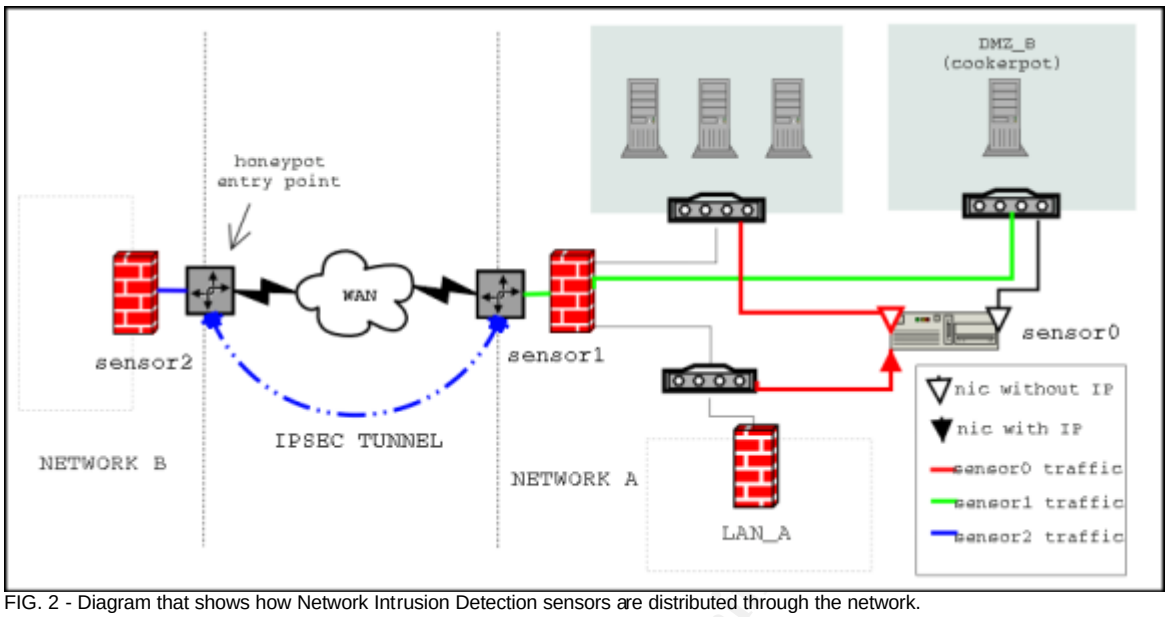

FIG. 2 - Diagram that shows how Network Intrusion Detection sensors are distributed through the network.

The main sensor was installed in the IDS server. In this sensor I have configured snort to listen traffic in the network cards connected to DMZ_A and LAN. It's also in this machine that the MySQL server will run, allowing connections from the LAN network segment to let other sensors to write network events data to the database.

In FW_A I have configured a secondary sensor (sensor1) listenning traffic flying to the cookerpot and logging it in the MySQL database at the IDS server. I choose to do that in the gateway because this would allow me to deploy snort inline in the future, to take active response to threats coming from the cookerpot.

In it's turn, FW_B in the remote network has another secondary sensor installed (sensor2), listenning traffic in both WAN and LAN interfaces, logging to the MySQL database at the IDS server and also locally, since the VPN connection can go down and this will allow the sensor to continue monitoring intrusion detection for the LAN at network B.

Remember, FW_B is the primary cookerpot entry point. The fact of this sensor is listening on the WAN interface is only for testing purposes, I really want to see for what the honeypot is proven from the Internet.

Limiting their moves

When setting up a machine with the purpose to have it rooted by malicious hackers, limiting what they do to all other machines in the same network or in the whole Internet is very important. It's here that Data Control appears.

In this aspect, I have configured the gateway FW_A to limit outbound connections from DMZ_B to all other zones in the firewall setup, this includes all other segments of the network and also the Internet. With that I pretend to break the majority of attacks starting in the cookerpot.

In the short future I plan to do more advanced hardening in this part of my network setup, one thing that I surelly will try is running Snort Inline [7] to block and modify packets based on attack signatures.

Cooking the Pot

Until here, the text described the network setup, now it's time to speak about the cookerpot installation itself.

The first thing to do when installing a cookerpot is to grab an Cooker image from one of the download mirrors available [u2]. After this one thing that must be done is to choose what type of cookerpot will be installed. The Cookerpot Project itself is more interested in probe server based installations rather than general user based ones (desktops and the likes). Within server based installations we can choose what level of security the machine will have, what servers should run on it, and so on.

I choose to install a standard server with some services running exposed to the Internet: the Apache webserver for both http and https services, OpenSSH for ssh, proftpd for FTP, Postfix as MTA for both POP3 and SMTP. All this programs come bundled with Mandrake Linux.

After the installation, I have created some regular users on the machine and some files on their home directories to let potentials attackers think that this machine is being used by someone. I also have uploaded a fake website designed by me to take attackers in error again and let them think that the machine is being used to host a true website. I left proftpd, the FTP server, with the anonymous account open and almost with the default configuration. Then I checked the Postfix installation and changed some things to not allow it to be used as an open SMTP relay, since I really don't want spammers using my cookerpot's resources, this will not be a spammer honeypot, oh no!...

One thing that Mandrake has that is really very nice is msec, also known as the Mandrake Security Package [9]. Msec hardens a system based on security level criteria, this is, from a given level from 0 to 5 the security will be set from poor to paranoia. Msec runs a lot of checks in a daily basis, depending on the level, that goes from standard file permissions to rootkits check up. It allows generated reports to be sent by email and logged into syslog.

I have choosen msec level 3 for this setup. This will give me a medium security level, and is the default in Mandrake's installation, and surelly it's how almost servers out there are configured. In the future I will configure it to run in higher levels to see how the system is vulnerable in more advanced security checking.

After the installation I have hacked the bash source and applied the patch that captures keystrokes and forwards them to syslogd [u8]. With this, all keystrokes typed in the bash shell will be sent to syslog, this will allow us to know all the commands that the evil hacker typed during an attack. Since the very first command a attacker will type once in the shell is "`ln -sf /dev/null ~/.bash_history`", this type of patch will prevent the bash history of his session to be lost. And since syslog is configured to send all data to the log/ids server, all his commands will be kept safe for latter analysis.

One thing that is important when time comes to do some forensic work after an attack, is the time settings on the server. I use the Network Time Protocol (NTP), to synchronize dates in all my machines. This will prevent logs with timestamps based on machine CMOS clock that differs from machine to machine, allowing to better cross reference all logs from all machines in the log server. The following crontab job will do the trick:

```
5 0,12 * * *  /usr/sbin/ntpdate -su ntp.example.com && /sbin/hwclock -w > /dev/null 2>&1
```

This will synchronize system's datetime and hardware clock with the NTP server ntp.example.com twice a day.

At the end I have configured Tripwire in the machine [u9]. Tripwire is a host based intrusion detection system (HIDS), that will check for intrusions checking files on the system against its file integrity database. The first thing to do when installing Tripwire is to generate this integrity database properly, then save all generated files to a safe medium or server and delete all policy files that are kept in plain text, since this will prevent an attacker to see what files are you checking for. Msec also runs tripwire checks, but I prefer to configure it myself, and put it to run at the crontab twice a day mailing the report properly encrypted with GnuPG to my mailbox.

Given that, the cookerpot is now ready to be hacked!

Rules of engagement

It's now time to talk a little about the Cookerpot Project's objectives and how the data produced from cookerpots like the one described in this paper will help to lead to final conclusions about the security of Mandrake Linux.

Once again, the main project's idea is best described from the Cookerpot Project Manifesto [1], which states the following:

> *[...] we will seek to establish a central reporting repository where honeypot maintainers will be able to contribute their data in an effort to establish as broad a picture as possible of what the Mandrake community faces in the way of outside threats when they connect to the internet. This repository once mature should serve as the basis for a Cookerpot Project white paper highlighting different outside threat trends that affect Mandrake users connected to the Internet and evaluate how Mandrake Linux deals with these threats and what can be done to better how Mandrake Linux and users deals with them.*

The data reporting attacks to cookerpots will be discussed on the Cookerpot Mailing List [u1]. If the method of such attack can be prevented by better configurations, disabling unwanted services, or any other possible solution, a conclusive post on the Cooker Mailing List [u2] will report such attack as well describe measures that can be take to prevent this kind of attack by default in future releases of Mandrake Linux.

As an example of a positive contribution by the Cookerpot Project, let's suppose that the cookerpot configured in this paper faces a successful attack to it's anonymous FTP account, the intruder gets system access and plays a little with it. After a little forensics work, analyzing logs and dumps of the tcp session the members of the Cookerpot Mailing List notice that the attack only was successful because of a bad practice on the default proftpd configuration that lets the anonymous account open by default.

This information is posted on the Cooker Mailing List describing that this kind of practice should be avoided since it's a major risk to the FTP server security. The Madrake's main developers notice that this information is correct and change the proftpd configuration to avoid this type attack.

Conclusion

This was the setup of my first cookerpot. Until now I haven't collected enough data to provide the project with some conclusions but I hope to have it in the near future. This is why this paper is not about data analysis and do not cover attacks already made, purely because my honeypot is very new and without enough days on the wild.

Concerning to the Cookerpot Project initiative, it's main idea is somewhat unusual, proposing a new method to aid software development, using honeypot technology to prevent and fix bad configurations issues and other insecure practices. Surely, this approach will lead to a better and more secure Mandrake Linux. We hope that, at least.

References

[1] The Cookerpot Project . "Cookerpot Manifesto". December 30, 2002
url: http://cookerpot.linsec.ca/bin/view/Main/CookerpotManifesto
(Last checked in February 04, 2003)

[2] The Honeynet Project. "Know Your Enemy". July 21, 2000
url: http://project.honeynet.org/papers/enemy
(Last checked in February 04, 2003)

[3] The Honeynet Project. "Know Your Enemy: Honeynets". January 07, 2003
url: http://project.honeynet.org/papers/honeynet
(Last checked in February 04, 2003)

[4] Spitzner, Lance. "Building a Honeypot". 20 Mar 2002
url: http://rootprompt.org/article.php3?article=210
(Last checked in February 04, 2003)

[5] Balas, Edward. "Honeypot Bandwidth Rate Limitation". May 13, 2002
url: http://www.honeynet.org/papers/honeynet/tools/dc.html
(Last checked in February 04, 2003)

[6] The Honeynet Project, "GenII Honeynets Definitions, Requirements, and Standards". January 07, 2003
url: http://www.honeynet.org/alliance/requirements.html
(Last checked in February 04, 2003)

[7] The South Florida Honeynet Project. "GenII Data Control for Honeynets, Understanding and Building Snort-Inline Data Control". January 08, 2002
url: http://www.sfhn.org/whites/gen2.html
(Last checked in February 04, 2003)

[8] Bauer, Mick. "Paranoid Penguin: Stealthful Sniffing, Intrusion Detection and Logging". Linux Journal, Issue 102. October 01, 2002.
url: http://www.linuxjournal.com/article.php?sid=6222
(Last checked in February 04, 2003)

[9] Danen, Vicent. "Introduction to msec". March 25, 2002
url: http://www.mandrakesecure.net/en/docs/msec.php
(Last checked in February 04, 2003)

[10] The Honeynet Project. "Know Your Enemy: VMware". January 27, 2003
url: http://project.honeynet.org/papers/vmware
(Last checked in February 04, 2003)

[11] Gubbels, Kecia. "Hands on the Honeypot". November 03, 2002
url: http://www.sans.org/rr/intrusion/hands.php
(Last checked in February 04, 2003)

[12] Sink, Michael. "The use of Honeypots and Packet Sniffers for Intrusion Detection". April 15, 2001
url: http://www.sans.org/rr/intrusion/honey_pack.php
(Last checked in February 04, 2003)

Useful URLs

[u1] The Cookerpot Project homepage
http://cookerpot.linsec.ca

[u2] The Mandrake Linux Cooker homepage,
http://www.mandrakelinux.com/en/cookerdevel.php3

[u3] The Honeynet Project homepage
http://www.honeynet.org

[u4] Snort IDS
http://www.snort.org

[u5] Snort Users Manual
http://www.snort.org/docs/writing_rules

[u6] Demarc PureSecure
http://www.demarc.com/products/puresecure

[u7] Installing Snort, MySQL and Acid,
http://www.snort.org/docs/snort-rh7-mysql-ACID-1-5.pdf

[u8] Bash patch that captures keystrokes and sends them to syslog.

The original by Antonomasia applied to bash-2.03:
http://www.honeynet.org/papers/honeynet/tools/bash.patch

The one I have applied to Mandrake's bash-2.05b-12mdk, here you will also find source and binary RPMs for it:
http://devfusion.net/~vsantola/packages/bash-syslog/

[u9] Tripwire
http://www.tripwire.org