



DEBIAN & UBUNTU SERVER HARDENING GUIDE

VERSION 1.0 - 11/09/2009

Ary Kokos

Secure Network S.r.l
ary[DOT]kokos[AT]securenetwork.it

Special thanks to :

Claudio Criscione
Quentin Lampin
Antonio Galante
Stéphane Poignant
Luca Carettoni

for their help and advices

Index

Forewords.....	7
Color Convention.....	8
Introduction.....	9
A story of paradigm, men and money.....	9
Hardening Linux.....	10
Fairy Tales.....	11
I. Foreword on security management.....	13
I.1 On risk and security management.....	13
I.1.1 On IT Security Management.....	13
I.1.3 Hardening.....	15
I.2 Lessons learned from real life.....	16
II. Hardening guide : MindMap.....	17
1. Installation.....	18
1.1 Partitioning scheme.....	19
1.2 Minimal Installation.....	21
2. Authentication and Access Restrictions.....	22
2.1 Authentication : password based.....	22
2.1.1 On password based authentication.....	22
2.1.2 Choosing good passwords.....	23
2.1.3 How an attacker will try to break your passwords.....	24
2.1.4 Insight on password quality parameters.....	24
2.1.4.1 Length.....	24
2.1.4.2 Entropy – password generator.....	26
2.1.4.3 Social.....	27
2.1.4.4 Easy to remember.....	28
2.2 Strong password policy – Password quality enforcers.....	29
2.3 Authentication : other means.....	30
2.4 Access restriction : physically access restriction.....	31
2.5 Access restriction : Bios and Bootloader.....	32
2.5.1 BIOS.....	32
2.5.2 Bootloader.....	32
2.6 Access restriction : system access restriction, unix user and groups.....	33
2.6.1 Users and groups, file permission.....	33
2.6.1.1 File permissions.....	34
2.6.1.2 Users and groups.....	34
2.6.2 Elevation scheme, using sudo.....	35
2.7 Access restriction : about advanced access restriction, SELinux.....	37
2.8 Access restriction : network access restriction, Firewalling ufw and Shorewall...37	
2.8.1 ufw.....	38
2.8.2 Shorewall.....	39
2.8.2.1 On netfilter and IPTables.....	39

2.8.2.2 Shorewall installation and configuration.....	40
2.9 Remote access, OpenSSH.....	43
3 Reducing the attack surface.....	47
3.1 Disabling unneeded daemons and services.....	47
3.2 Checking file permissions and system executables.....	48
3.3 Deleting or disabling unneeded user accounts.....	50
3.4 TCP/IP Stack hardening.....	51
3.4 Disabling ipv6	52
3.5 Filesystem mounting options.....	52
3.6 Misc and Kernel Hardening.....	53
4. Monitoring and detecting intrusions.....	55
4.1 Alert transport mechanism : Postfix.....	55
4.2 Host based Intrusion detection systems.....	58
4.2.1 OSSEC.....	58
4.2.2 Other HIDS.....	59
4.2.3 Rootkits hunters.....	59
4.2.3.1 Rkhunter.....	59
4.2.3.2 Chkrootkit	61
4.2.3.3 Unhide.....	61
4.3 Monitoring logs.....	62
5.2.1 Logwatch.....	62
5.2.2 Logcheck.....	63
4.4 Watchdogs.....	65
5. Keeping up to date and informed.....	66
5.1 Keeping Up To Date.....	66
2.2.1 Updating with apt.....	66
2.2.2 Automatic notifications.....	67
5.2 Informed.....	69
Chapter 6 : Mitigation and Confinement.....	70
6.1 Intro	70
6.2 Access control	71
6.2.1 SELinux	71
6.2.2 GRSecurity, AppArmor, TOMOYO, SMACK and co	74
6.3 Virtualization	75
Chapter 7 Advanced Hardening.....	77
7.1 SSH.....	77
7.1.1 Invisible.....	77
7.1.2 Dedicated connection.....	77
7.1.3 Portknocking.....	78
7.1.4 Airlock.....	78
7.1.5 Anti bruteforcing.....	78
7.1.6 Using keys.....	79
7.2 Anti scanning.....	79
7.3 Kernel Hardening.....	79
7.4 Delusion and obfuscation.....	81
7.4.1 Hiding Banner informations.....	81

7.4.2 Deluding Scans.....	82
chapter 8 : random little things.....	83
8.1 Bash history.....	83
8.2 Blowfish /etc/shadow.....	83
8.3 Absolute Path.....	84
8.4 Web Apps.....	84
8.5 About Bubbles.....	84
8.6 About dedicated management networks.....	84
8.7 Secure deletion.....	84
8.8 Gcc SSP.....	85
8.9 More links on IDS.....	85
8.10 Deception Networks	85
8.11 Bastille.....	85
8.12 PSAD.....	85
Appendix C : Some security links.....	86
Advisories :.....	86
Computer security experts Blogs :.....	86
Other source of information :.....	86
References.....	87
Books in French.....	87
Books in English.....	87
Other Hardening guides.....	88

Illustrations

Illustration 1: Foreword : MindMap.....	9
Illustration 2: Chapter 1 : MindMap.....	14
Illustration 3: Sample partitioning scheme.....	15
Illustration 4: Minimal install : nothing selected in this screen.....	17
Illustration 5: Chapter 2 : MindMap.....	18
Illustration 6: Password attack type.....	20
Illustration 7: Random password generation (/dev/urandom and tr).....	23
Illustration 8: Password quality checker control.....	26
Illustration 9: Privilege elevation scheme.....	32
Illustration 10: nmap output before and after firewalling.....	38
Illustration 11: Chapter 3 : MindMap.....	42
Illustration 12: sysv-rc-conf daemon configuration.....	42
Illustration 13: Chapter 4 : MindMap.....	50
Illustration 14: rkhunter --check.....	54
Illustration 15: Chkrootkit at work.....	56
Illustration 16: Unhide sys at work.....	57
Illustration 17: An extract of a Logcheck report.....	58
Illustration 18: Cron-apt.....	63

FOREWORDS

This document is meant to be two things at the same time.

On one side, an introduction to computer security for all the system administrators who are moving to Ubuntu in the last few years, but who are finding themselves in a new world where there's no official Security Guide. We hope to provide a jump-start to security best practices in the Ubuntu ecosystem: if you are going to work with these systems, then this is a good place to start. Keep in mind, however, that this guide is not meant to be a walkthrough in Linux administration: a basic understanding of how the Operating System work is required.

On the other side, we think that even experienced and security-wise system administrators, who have not yet developed their own standard for secure installation and hardening will find this document useful as a reference and guide. Maybe there will still be a trick or two you didn't know about.

Any remark, critic or error report is warmly welcomed: feel free to write to the author – [ary\[dot\]kokos\[at\]securenetwork.it](mailto:ary[kokos@securenetwork.it]) – or to me – [claudio\[dot\]criscione\[at\]securenetwork.it](mailto:claudio[criscione@securenetwork.it]).

I hope you will find this guide useful, and maybe that you will help to improve it!

Claudio Criscione

COLOR CONVENTION

The color idea was taken from the website of Bob Cromwell [Cromwell]: paragraphs are highlighted in different colors according to their “level”.

This manual may also be used as a quick reference guide: in this case the reader may go from highlighted part to highlighted part.

Green – 1 bar :


Green instructions correspond to a basic security level, requiring a minimal time investment but only granting a small improvement in security.

Orange – 2 bars :

Orange instructions represent a low-medium security level which may require a higher time investment.

Red – 2 bars (1 thin, 1 large) :

Red instructions correspond to a higher level of security, but have to be adapted to the reader' system, require good understanding of the technologies and their environment and probably a lot of testing before going live.

 Blue (2 fat bars) : Stories, anecdotes.

Purple (double line) :
Non finished parts.

INTRODUCTION

Hardening is a process which aims at securing a system ; absolute security is impossible to reach but reducing the surface attack and reaching an equilibrium between security and cost is possible.

Hardening a server means, at the practical level, **reducing** as much as possible the **attack surface** and monitoring the exposed part to **detect intrusions**.

This guide will describe and explain how to secure an Ubuntu Server (and, to some degree, a Debian system as well) up to reasonable level of security: a compromise between security, usability, time and costs.

All the instructions present in this guide were done and tested on **Ubuntu Server 8.04** or **Debian Etch**.

Before reading this guide, we would like the reader to keep in mind two important points :

- Every attack, if your enemy is decided enough, is successful

If an enemy is decided enough, he will break in. It is only a matter of how much resources he is able to invest ; and time.

- Linux is not a secure operating system, and will never be one

Most of our information systems are fundamentally insecure. This happens because they were never made with security as a primary aim.

Popular Operating systems such as Unix/Linux or Windows are not an exception.

A story of paradigm, men and money

Once upon a time, we started building information systems.

At this time they were made to be used by a limited number of skilled and trusted persons. The systems were rarely interconnected and when they were, the network was considered as secure.

The systems were rarely standard and only a limited number of skilled engineers could understand and use them.

The security mechanisms were mostly conceived to avoid human mistakes or buggy programs. And they were effective.

Then the world changed.

Now we live in a world where systems become more and more complex, ubiquitous, standard and interconnected while users are less skilled and often not trustworthy.

Banks, industries and governments which used to operate mainframes and custom systems, now tend to use standard systems, due to the economical pressure.

And all those systems are interconnected.

The problem is that nowadays our systems can not only be attacked by a limited number of skilled attackers but also by the first sheep who is able to use a search engine and a mouse. Communication interception and information systems espionage and disruption which were reserved to government entities or some powerful organization, becomes now feasible by a single non skilled person.

And we still use systems based on the old paradigm, that we just patched to achieve a minimal security.

Hardening Linux

Using a linux system is not a protection by itself. It is as vulnerable as a windows or solaris operating system, and was never conceived to be a secure operating system.

Of course some projects try to enhance its security, some of them by limiting the attack surface, other aiming at the correctness of the configuration or trying to retrofit the security monitor concept. But as its conception and development does not aim at producing an high security system, this is quite useless against a decided attacker. There is not even a security kernel, nor verified code nor mandatory access control by default. Even with a SELinux patch, which greatly improves the security, it is not sufficient as it does not work on a security kernel. Huge flaws such as the possibility to patch the kernel via /dev/kmem, running network facing demons as root thus requiring a huge TCB or shared resources are still present on default installs. Unix systems just fail to verify the complete mediation, tamperproof and verifiability properties. For more information on secure systems, we advise the reader to refer to Operating System Security of Trent

Jaeger chapter 4 part 2, Morgan and Claypool Publishers; 1 edition (October 7, 2008).

Hardening a system is like trying to secure a building. Of course the first things that you should do is to check that all the doors are closed, force the person to use one entry, identify them, ask for their ID card and even put a camera and guards. This system will never stop a trained attacker, he will just find a way to enter the building.

But maybe it is sufficient to stop certain attackers : the non skilled one, and maybe that your threat model only considers them and that you accept the risk represented by the trained one.

All this diatribe against the current security state shall not stop you from hardening it. Hardening a system, especially based on linux is an hard but feasible task : it consists in finding the equilibrium between security and costs (including also usability and innovation).

The aim of this guide is not to protect you against very determined or competent attackers - which requires a real security management, very competent persons and a lot of resources - but to provide you some weapons to defend your systems against the most common attacks. The whole idea is to raise up the cost of attacking your systems to discourage most of the non competent attackers.

Competent or decided one will ever find a way to enter, if it is not using the computers it will be using the human weaknesses, but non-competent one will pass to the next target.

It is like putting a lock on your luggage at the hotel when you go out : it will never stop a competent attacker who will just locket it nor a decided one who will just take the whole luggage and break the lock at home, but it will keep most of the masses honest. And for most cases, it is sufficient. If it is not, find another way to secure your belongings.

Fairy Tales

Here are a few stories that happened during our audits.

#1 IIS.vbs

During an audit, nessus reported us an open ftp service on a windows server among hundred of other ones. Inside there were a few files and a strange iis.vbs. Opening it we found a couple of credentials : Administrator/companyName=000. As we could not believe it we tried it and got full admin access on this machine.

Afterward we discovered that this computer was used to automatically deploy/updated applications on all the domain.

#2 Oracle forms

During a pentest, after 10 days exploiting SQL injections, we casually discovered that the developers left a login/password in the html comments of the oracle forms web page. These credentials gives full access to all the database.

#3 Jboss and backup.tar

During a VA, we found a Jboss admin interface left open. Loading a shell.war, we started walking accross the directories. In /tmp we found credentials for an ftp service, a lot of deployment scripts with 5 user credentials and a 6 Go backup.tar.

From the credentials we learned the password scheme of the company :
companyNameService, let say that the company name was foo, the "webservers" account password on the "webserver" host was fooWeb, the oracle account on the oracle server was fooOra, the tomcat fooTom and so on. We quickly got access to most of the company servers. Of course all the kernel were old, an vulnerables, especially to the vmsplice() local root exploit.

From the backup.tar, we got access to all the compagny software source code.

In other cases we got access to all the customers database by a single telnet -l "-froot" on an old solaris system. Other times it was just by using john on /etc/shadow obtained via a directory trasversal (the admin left the file world readable).

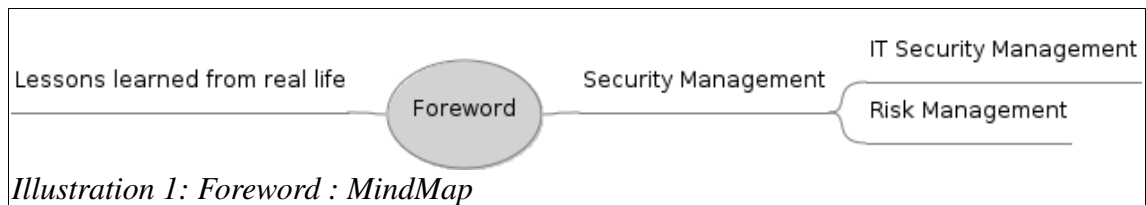
All those attacks could have been avoided by very simple security measures.

Those are the one described in this guide.

I. FOREWORD ON SECURITY MANAGEMENT

GOAL :

THIS CHAPTER WILL PROVIDE A SHORT INTRODUCTION TO IT SECURITY MANAGEMENT, PRESENTING SOME IMPORTANT NOTIONS, MOSTLY USEFUL TO A SYSTEM ADMINISTRATOR IN ORDER TO UNDERSTAND THE NEXT PARTS. IN A SECOND PART, SOME REAL LIFE STORIES. EXPERIENCED SECURITY SPECIALISTS MIGHT WISH TO SKIP THIS PART.



I.1 On risk and security management

I.1.1 On IT Security Management

Hardening is a part of IT security Management which is itself, in most models, a part of risk management.

In the last years IT security evolved from a technical science, involving mostly the IT department, to a management activity driven by the top management: from computer security to **information** security.

A few years ago, threats rarely came from the network and where mostly coming from the inside. Securing the system was a matter off controlling access and preventing physical disasters. Nowadays the major part of the information is digitalized on complex inter-operable ubiquitous systems :

- systems need to be interconnected and inter-operable
- the information is stored anywhere, from a mainframe to a PDA and we expect to easily access every information from everywhere

•systems become more and more complex and hence harder to secure.

Due to the challenges posed to the last three points, which completely change the scale – both in numbers and in size – of the involved technologies, , security is in a first place a matter of management, and then a technical matter.

The classical cycle of IT governance is very similar to a Deming's wheel, or a **Plan-Do-Check-Act** cycle: :

1- Identifying risks and objectives

Identify which are the risks, which is usually done regarding to :

- Confidentiality
- Integrity
- Availability
- Traceability

We will not go into more details on what a risk is and how it should be identified in this introduction, but we will use the term Risk in the rest of the guide as a synonym of Threat, even if we know that a Threat is only a part of a Risk.

Once the risks are identified, for each of them, you will usually need to **evaluate** :

- The **occurrence** : what is the probability of this risk happening
- The **importance (or impact)** : what are the consequences regarding this risk (direct and indirect losses in terms of productivity, reputation, confidential informations, etc.)

Once identified, an action should be taken to deal with each risk, from the following list. As we will soon see, these actions are merged into a global Plan.

Avoidance (eliminate)

Reduction (mitigate)

Transfer (outsource or insure)

Retention (accept and budget)

from [W risk mana]

2-Planning

This part consist in studying and choosing the best solution for each risk, which will consist both in technical and human solutions.

This part results in an organization (or business unit) plan.

3-Deployment

Realizing the plan , by **setting rules and regulations**, which shall be enforced as any

other existing company policy: when it comes to information security, however, **training for the users** has to be considered as a core part of the solution, since security always requires a trade-off with usability and users are not always happy to be part of the process!

Once policies and training is in place, **technical means** to enforce them should be devised and applied: every security system and technical procedures falls in this category.

4-Use

Governance of usage includes two different kind of **operations: every day task**, that is updating, processing logs, backups, keeping administrators and users informed on new trends and up to date, adding new users and so on; and **exceptional tasks**, which include reacting to an intrusion, checking an anomaly and so on.

5- Check

This step has to be performed in order to evaluate the overall efficiency of the system. This is usually achieved through two different means, that is by leveraging **active checks** and **passive checks**. Audit are the most well known of the active checks, and can be both internal and external. Each type of audit has his own pros and cons, external audit being often less pervasive but far more objective and trustable and internal audit being able to reach greatear details (due to the knowledge of the audited infrastructure) but being subject to greater biases from the context. Passive checks encompass every control which is routinely performed by manual or automated means, like log checking or automatic detection of attacks.

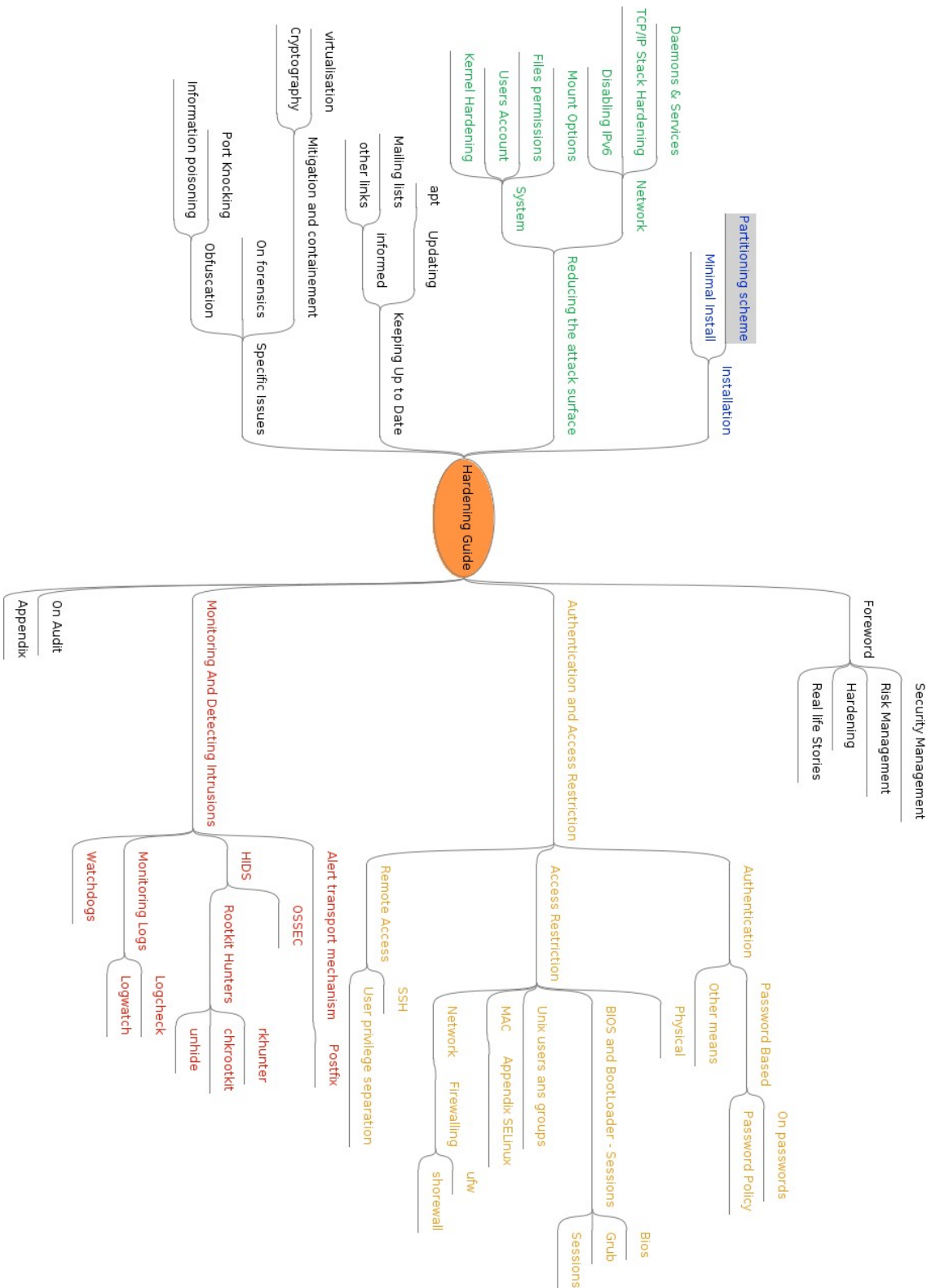
For further reading, we advice the reader to refer to the pointers in the reference part and to well designed methods for risk assessment and management such as EBIOS, MEHARI, CRAMM or OCTAVE. [Mehari]

I.1.3 Hardening

Hardening is a process which aims at securing a system ; absolute security is impossible to reach but reducing the surface attack and reaching an equilibrium between security and cost (where with cost we refer to both implementation, manteinance and usability costs) is possible.

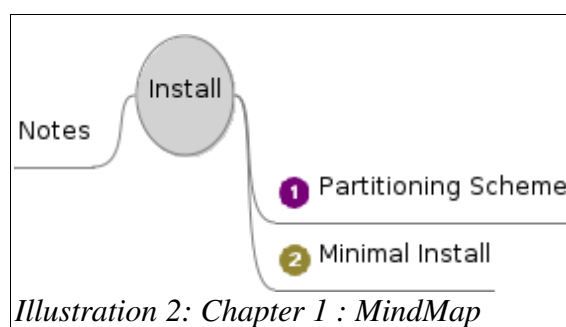
Hardening a server means, at the practical level, reducing as much as possible the attack surface, and monitoring what is exposed to detect intrusion.

II. HARDENING GUIDE : MINDMAP



1. INSTALLATION

IN THIS FIRST CHAPTER WE WILL FOCUS ON THE SYSTEM'S INSTALLATION, AIMING AT A MINIMAL INSTALL.



Hardening a system is much easier if you start from a **minimalistic system** and then add only the needed services. Hardening a complex system is possible but has a higher cost and is much more complicated, since it is easy to forget some (apparently) harmful piece of software somewhere in the machine. Even with modern packet managers, handling installed packages isn't an easy task.

Important note : A very important point is to only perform atomic operations which can be easily reverted in case of a mistake. This means that if you are editing your OpenSSH Server configuration files you should not modify 5 parameters at the same time: in case of a mistake you won't be able to individuate which one is responsible of the failure. Instead change the first one and test, then proceed to modify the second one and so on.

Virtual machines: If possible, and relevant regarding your requirements, use virtualization to your advantage :

- for testing purpose : reverting to a previous consistent state using snapshots is very simple
- copying, making a backup, moving, deploying virtual machines is done in a very short amount of time

There are a lot of different solutions on the market, for more information we invite the reader to read the corresponding Wikipedia article [Virt]. For informational purposes, the author used VMware Server [VMware] while testing the setup in this guide.

Ubuntu server installation is straightforward and well documented [Ubuntu Server Install], and there are only 2 points to which you may take care of : partitioning and package selection.

For Debian installation, the reader is advised to refer to this document [Debian Install]

1.1 Partitioning scheme

During the installation, you will be asked for a partitioning scheme ; a partition is a logical part of your hard disk on which you will put a file-system, which in turn will contain your files.

The easiest solution is to put all in a single partition, but it doesn't provide as much **granular control** as having different partitions.

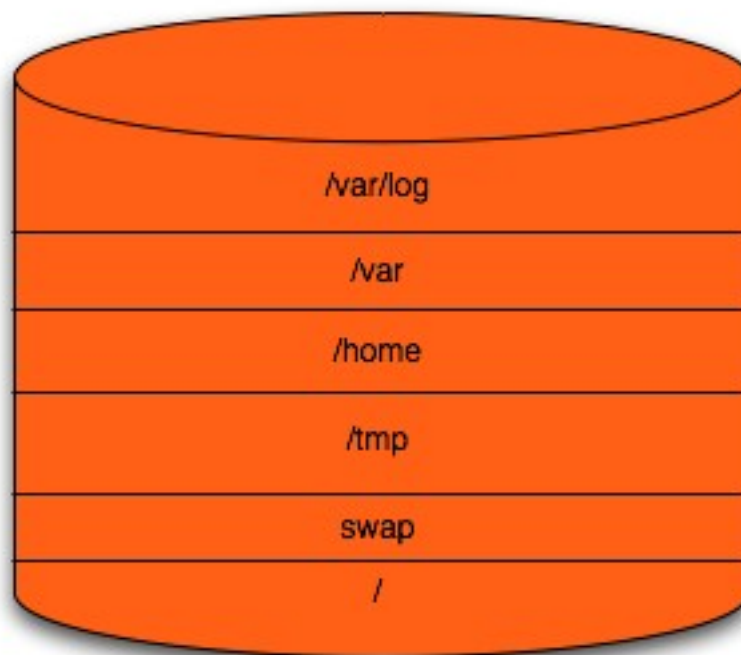


Illustration 3: Sample partitioning scheme

Different partitions can be used for different means, let see it through some examples :

- **Performance** : if your server acts as a virtual machine host, having /var on a second partition (or better a second disk) with the *noatime* option (not writing the last access time of the files) improves performances noticeably. (nb putting noatime on the whole system is very bad for accountability, since an ext3 file system with the noatime option will not record access time to files.).

- **Security** : some attacks rely on being able to execute binaries in /tmp, mounting tmp with the noexec option will prevent this family of attacks

- **DOS defense** : if for some reason a daemon starts to generate a lot of logs and completely fill your drive, there will not be enough place for normal operations: this will most likely result in a DOS for your system, which is generally named “Starvation DOS”. Here a solution is to put /var/log in a different partition.

To choose a partitioning scheme, the first step is to identify the function of the server and what are the requirements of the softwares which will run on it.

For example a mail or a web server will require a big /var but not a big /home. A VMware host will require a big /var/lib/vmware.

Making this choices is not easy, but we can identify 2 approaches :

- based on calculus
- based on experiment

Building a test machine and monitoring actual, real-world disk usage may be a good option if enough time is available, as would be analyzing an existing machine covering a similar role.

As a rule of thumb, most network deamons leverage a subdirectory of /var as their main storage in Ubuntu.

Some examples of partitioning schemes follow; mount options will be described later.

Example for a mail or webserver (a lot of data in /var):

```
/
swap
/tmp
/home
/var
/var/log
```

Example for a VMware host (it is better to have a dedicated partition for VM):

```
/
swap
/tmp
/home
/var
/var/lib/vmware
```

1.2 Minimal Installation

Choose the **minimal install**, just the basic system, nothing more: in this way, even if you will have to install more packages at a later stage, you will retain full control of what is installed in your system. Hardening is also about control, since attackers don't really whether you are using a vulnerable software installed on your machine or not.



Illustration 4: Minimal install : nothing selected in this screen

2. AUTHENTICATION AND ACCESS RESTRICTIONS

IN THIS PART WE WILL PRESENT SOME AUTHENTICATION PARADIGMS, WITH A SPECIAL INSIGHT ON PASSWORD QUALITY. THEN WE WILL CONTINUE ON ACCESS RESTRICTION AT SYSTEM AND NETWORKS LEVELS.

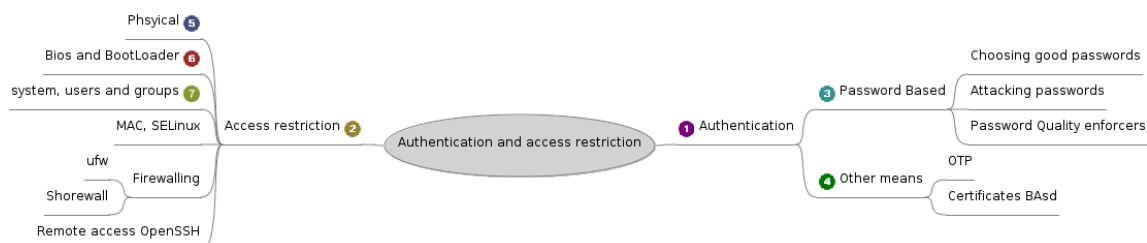


Illustration 5: Chapter 2 : MindMap

2.1 Authentication : password based

One of the key to security can be summarized as follows:

Choose good passwords.

It is that simple. Choose good passwords, for any user of any service, and you will noticeably improve the security of your system.

From all the possible authentication schemes, **password** based ones are the most common one as they are the **default mechanism** for a large majority of systems and are very **low cost**, since they most likely only require a keyboard to be used.

Good passwords are of tremendous importance as they are used to gain access or use privileges.

Weak passwords are still one of the most widely present and exploited vulnerability of a system.

2.1.1 On password based authentication

Before describing the different parameters which make good passwords, it is important

to understand what “a good password” actually means:

A good password is a password which is not easily guessable nor easily forced.

Not easily forced means that if an attacker tries a lot of passwords, without any other knowledge, it will require him a long enough time to find the right one.

Here the quality of the password is not the only parameter, the system which will check it is as important as the password.

If there is any **flaw in the authentication mechanism**, the quality of the password becomes useless. Here we will take an example of an exploit found on milw0rm :

```
=====
3Com OfficeConnect Wireless Cable/DSL Router Authentication Bypass
Original Advisory: http://www.ikkisoft.com/stuff/LC-2008-05.txt
luca.carettoni[at]ikkisoft[dot]com
=====
An unauthenticated user may directly invoke the "SaveCfgFile" CGI
program and easily download the system configuration containing
configuration information, users, passwords, wifi keys and other
sensitive information.
http://<IP>/SaveCfgFile.cgi
```

This kind of errors can also be found at the operating system level, for example the **telnet – l “-froot”** command which, on some versions of the Solaris operating system, lead to a remote root shell.¹

The time required to check the password is also important :

A system with 6 char password which requires almost 20 seconds to check a password for correctness is much safer than a system with 8 chars where checking one password will require 1/10000s (time to check all the key-space $26^6 \cdot 20 = 6e9$ vs $26^8 / 10000 = 2e7$). Obviously, speed check can be tremendously influenced if the authentication is performed over a network.

Last but not least having a complex password, but easily guessable, is useless.

During a Vulnerability Assessment, for instance, we saw system administrators using their company name written in “pseudo leet” as root passwords. If the company name was Artichaud, the password was 4rt1ch4ud, which technically is not a bad password, but is definitely easily guessable by an attacker.

¹ <http://milw0rm.com/exploits/3293>

2.1.2 Choosing good passwords

Do	Don't
At least 8 characters containing special char such as ,:~_&/ is a minimum 10 to 12 is advised or passphrases	Generic passwords such as admin, Password0, root, password, etc
Choose passwords which will be accepted by your users and easy to remember.	Passwords related to publicly available information such as company's name, server function, personal information
Use password quality enforcers to check the quality of the passwords.	Words present in any dictionary of any language

2.1.3 How an attacker will try to break your passwords

When an attacker tries to break a password, he will:

- 1 : try **generic** passwords like admin/admin, Administrator/Password or default ones.
A nice list of generic passwords is available here : [DefPassList]
Even some worms will try to leverage default or trivial passwords in order to gain access to remote systems.
- 2 : identify and generate a list of passwords **related** to you or your company, or to the function of the server. *Database* is never a good password. This is generally called a “targeted” attack.
- 3 : leverage a **dictionary**, submitting any and all words in it, or even multiple dictionaries in different languages.
- 4 : if everything else fails, perform a **bruteforce**, that is trying all passwords, covering all the so called “key space”, that is the sum of all the possible combinations which can be generated using all the characters supported by the system up to the maximum password lenght. This will require time, depending on the speed the check is performed and the size of the key space.

2.1.4 Insight on password quality parameters

The quality of a password can be judged by many different parameters. In this section we will discuss some of them and how they interact to create a strong password.

2.1.4.1 Length

One of the most important parameter is the keyspace : the quantity of possible passwords. A bigger keyspace means a bigger number of passwords to test, and so a longer time : if cracking it requires 10 days for an admin password, there is no security ; instead if it requires 200 years it is safe for a normal use.

The keyspace size depends on the length of the passwords and the different characters which can be used.

Here are some numbers just to illustrate the principle.

Type	Password	Keyspace
Numbers, 4	3251	$10^4 = 1000$
Char, 8	fkjduhad	$26^8 = 2.1e11$
Char, num, 8	g5su96po	$36^8 = 2.8e12$
Char, maj, num, 8	kL89klI9	$62^8 = 2.2e14$
Char, maj, num, special (most used), 8	Jl@K:5&	$92^8 = 5.1e15$
Char, 8	fkjduhad	$26^8 = 2.1e11$
Char, 12	jatnpkcfszlz	$26^{12} = 9.5e16$
Char, maj, num, special (most used), 12	d=dLt@9p:kS7	$92^{12} = 3.7e23$
Char, 17	urjanhdpemkjtqplm	$26^{17} = 1.2e24$

Long story short: if you don't use special char, use a longer password to compensate.

Rainbow Tables : In some cases having a long password is not sufficient.

Normally a password is never stored in cleartext: there is no need for that, since every check and comparison can be performed by the operating system by only leveraging its *hash*. A hash is a mathematical operation which can associate any data to a fixed-size chunk of information: a good hash should be resistant to collisions (i.e. it should be very difficult to identify an input which will produce a given output) and this is the most important requirement in the security framework.

Here is an example with a well known hashing algorithm, **sha1** :

```
a:~ a$ echo "hello" | openssl sha1
f572d396fae9206628714fb2ce00f72e94f2258f
a:~ a$ echo "hella" | openssl sha1
1519ca327399f9d699afb0f8a3b7e1ea9d1edd0c
a:~ a$ echo "This is a long phrase" | openssl sha1
18cfab92cf66ef9f2ae4c5302ba3d500d307377e
```

Notice how the hash of *hella* is quite different from the one of *hello*. Since it is not possible to invert the function, for any good hashing function (where collisions are not

easily achievable) an attacker needs to calculate the hash for every password he tries and then compare it. This calculus is very time consuming when performed on such a huge amount of data. However, every time a given string, for instance *hello*, is hashed with the same algorithm the output is the same : f572d396fae9206628714fb2ce00f72e94f2258f. An attacker could just build a database of all possible passwords, the so-called Rainbow Tables, dramatically reducing calculus time from a hash to a research in a database. That's why modern systems use something called “salt”, a random input added to the calculus. This is just a dummy example to illustrate the concept :

```
password is hello ; salt =12 ; hash(pass, salt) =  
8f28e3ba39c87dacc148385e58ffe3a772b624b2
```

```
password is hello ; salt =18 ; hash(pass, salt) =  
9e720243cd309b471faad1259cfe14ca4c4190ab
```

Some non-Linux operating systems (most notably Windows) are vulnerable to Rainbow Table attacks, and tools/rainbow tables are available such as ophcrack² or the Free Rainbow Tables project.

2.1.4.2 Entropy – password generator

Having a big keyspace is not sufficient: the security of the password also depends on the “probability” that you choose a “random” password. Let's take an example of a sysadmin who used to choose his passwords like this (*pippo* here is his own company name) :

Pippo1927?

Pippo5672!

Pippo9232.

Pippo1532;

Pippo3427:

His passwords were quiet complex, but were always done on the same scheme : 10 chars, starting with Pippo, 4 numbers and 1 special char: the entropy of the passwords is very low.

So here you have two solutions : either use long passwords or get them from a good source of randomness, for ex /dev/urandom.

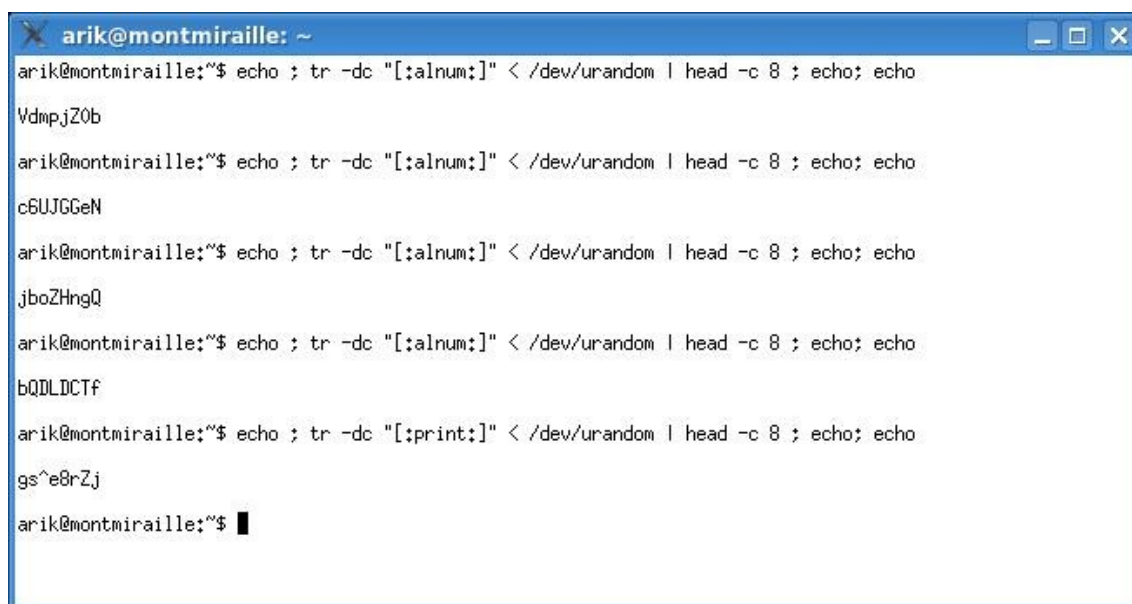
#SCRIPT : Generation of a random password in bash

```
password="$( tr -dc "[:alnum:]" < /dev/urandom | head -c 10 )";  
echo "Password is : ${password}"
```

(use password="\$(tr -dc "[:print:]" < /dev/urandom | head -c 10)"; for a bigger keyspace (:print: includes more chars than :alnum:), see “man tr” for details)

² <http://ophcrack.sourceforge.net/>

Some outputs :



```
arik@montmiraille: ~  
arik@montmiraille:~$ echo ; tr -dc "[:alnum:]" < /dev/urandom | head -c 8 ; echo; echo  
VdmpjZ0b  
arik@montmiraille:~$ echo ; tr -dc "[:alnum:]" < /dev/urandom | head -c 8 ; echo; echo  
c6UJGGeN  
arik@montmiraille:~$ echo ; tr -dc "[:alnum:]" < /dev/urandom | head -c 8 ; echo; echo  
jboZHngQ  
arik@montmiraille:~$ echo ; tr -dc "[:alnum:]" < /dev/urandom | head -c 8 ; echo; echo  
bQDLDTf  
arik@montmiraille:~$ echo ; tr -dc "[:print:]" < /dev/urandom | head -c 8 ; echo; echo  
gs^e8rZj  
arik@montmiraille:~$
```

Illustration 6: Random password generation (/dev/urandom and tr)

For further informations, the reader can refer to a NIST document³ or the SANS password policy⁴

2.1.4.3 Social

As we said before while describing the targeted attack type, an aggressor will try passwords related to the server context, such as name of the company or personal information related to the administration staff. As for dictionary attacks, there are huge wordlists available on the Internet, for example the one provided by openwall⁵

..... ;)

Are those passwords strong ?

“balderdash/Quatsch”

“window gardening”

“Sie ist franzoesischer Herkunft!”

“!@#%&^*”

“Legen Sie etwas fuer schlechte Zeiten zurueck!”

“maxtrix4x”

“uranium 235”

“SPSPARC&CACHE”

³http://csrc.nist.gov/publications/nistpubs/800-63/SP800-63V1_0_2.pdf

⁴ www.sans.org/resources/policies/Password_Policy.pdf

⁵[ftp://ftp.openwall.com/pub/wordlists/all.gz](http://ftp.openwall.com/pub/wordlists/all.gz)

|| Solution : **no** I got them from the all.gz wordlist

2.1.4.4 Easy to remember

Strong, complex passwords are not suitable for all situations: if they are too complex to remember the user will write them down on a paper, under their keyboard or simply forget them. 3B9d31c6e3d10ebf is a good password, but a very difficult one to remember.



ALWAYS CONSIDER THE PSYCHOLOGICAL ASPECT OF SECURITY. IF SECURITY IS NOT ACCEPTED BY YOUR USERS, THEY WILL TRY TO FIND A WAY TO BYPASS IT.

AN EXAMPLE : SOME COMPANIES HAVE THEIR USERS CHANGING THEIR PASSWORD EVERY MONTH WHERE THERE IS NO NEED TO DO IT. THIS ALWAYS RESULTS INTO DUMB PASSWORDS, SUCH AS AUGUST2008, SEPTEMBER2008 (MORE THAN 8 CHARS, 1 CAP + NUMBERS).

ADMIN PASSWORDS

► **Important** passwords (root/admin/encryption) :

A good password should be a minimum of **12 chars** long, with numbers, upcase, letters and **special chars**, or a long **passphrase** (more than **24 chars**).

As we said before, you can compensate lack of entropy by length

To remember, a solution may be to use a short story :

passphrase : marry gave me three dollars and a rose

password : marry3\$&1ROSE or marry3\$ROSE

► **Common** passwords :

12 chars should be used too, but if your users are not able to remember them, **8 to 10 chars** may be used.

USER PASSWORDS

Outside an high security context, it is very difficult to make users choose good passwords. The best solution would be a 8 to 10 characters long password which includes both alphanumeric and special characters, but as they are difficult to remember, a good solution is the **use of passphrases**.

An empirical study is available here: [Ross Anderson SHB 2008]

2.2 Strong password policy – Password quality enforcers

A good password policy – that is, a written rule requiring users to have strong passwords – is necessary, but it will never force an user to use good passwords.

To enforce password quality, we suggest 2 simple solutions :

- retroactive : regularly audit the passwords
- proactive : forbid every weak password when the users try to set one

Proactive password enforcing can be done easily via a package included in Ubuntu, **libpam-passwdqc**, which will check the quality of the proposed password and refuse it if it is too weak according to its configuration.

```
# apt-get install libpam-passwdqc
# nano /etc/pam.d/common-password
```

add :

For a high security system :

```
"password required pam_passwdqc.so max=72 similar=deny enforce=everyone
retry=2 ask_oldauthtok=update check_oldauthtok"
```

This is a default behaviour, a bit paranoid and sometimes a bit annoying for systems which do not require an high security, so you should lower the requirements.



USE PASSWORD QUALITY CHECKERS TO LIMIT THE USE OF WEAK PASSWORDS

```
You can now choose the new password or passphrase.
```

```
A valid password should be a mix of upper and lower case letters,
digits, and other characters. You can use an 8 character long
password with characters from at least 3 of these 4 classes, or
a 7 character long password containing characters from all the
classes. An upper case letter that begins the password and a
digit that ends it do not count towards the number of character
classes used.
```

```
A passphrase should be of at least 3 words, 12 to 72 characters
long and contain enough different characters.
```

```
Alternatively, if noone else can see your terminal now, you can
pick this as your password: "pursue-danger-neural".
```

```
Enter new password:
Weak password: too short.
passwd: Authentication token manipulation error
passwd: password unchanged
Try again? [y/N] █
```

Illustration 7: Password quality checker control

For a multiuser system, medium security

Example : password required pam_passwdqc.so min=disabled,12,8,6,5 max=40

From the manpage http://linux.die.net/man/8/pam_passwdqc :
min=N0,N1,N2,N3,N4

(**min=disabled**,24,12,8,7) The minimum allowed password lengths for different kinds of passwords/passphrases. The keyword **disabled** can be used to disallow passwords of a given kind regardless of their length. Each subsequent number is required to be no larger than the preceding one.

N0 is used for passwords consisting of characters from one character class only. The character classes are: digits, lower-case letters, upper-case letters, and other characters. There is also a special class for non-ASCII characters which could not be classified, but are assumed to be non-digits.

N1 is used for passwords consisting of characters from two character classes which do not meet the requirements for a passphrase.

N2 is used for passphrases. A passphrase must consist of sufficient words (see the **passphrase** option below).

N3 and *N4* are used for passwords consisting of characters from three and four character classes, respectively.

For further reading, check <http://www.openwall.com/passwdqc/>.

2.3 Authentication : other means

In some situations password based authentication is not appropriated :

- In environment requiring a very high level of security, policies may require multiple factor authentication.
- When users use weak passwords, write them down or pass them, and there is no “non-technical” mean to avoid this behaviour.

Generally speaking, authentication can be performed through three different means

What you know

Examples of this kind of authentication are password based authentications, but other examples include, for instance, a set of questions which the user is the only one able to answer.

What you have

This authentication schema is performed by showing the system the user owns something: it can be a physical token, a smart card or a digital certificate itself.

What you are

While this is mostly used for identification purposes, “what you are”, biometric authentication means are becoming increasingly common in many scenarios.

With strong authentication in most of the cases we mean *multiple factor authentication*, i.e. leveraging multiple authentication means at the same time. For example a smart card (or a token, or “something you have”) which will require a PIN (“something you know”) in order to be unlocked and be able to perform authentication.

The following table presents a short, by no mean exhaustive listings of authentication means:

Technology	Free Solution	Commercial Solution	Pros	Cons
One time password A one time password is generated, the user has to enter it to authenticate	List of passwords on a paper sheet OPIE or S/Key based with software on cellphone OTP over SMS	Using a token such as RSA SecurID, Entrust identity guard, etc	Low cost Easy to use	Man in the middle Social engineering
Biometrics				Can be tricked, since it's often undertrained due to overfitting issues.
Public key cryptography- Certificates	Openssl		With PIN on smartcard, very strong auth	Costs Complex to set up

2.4 Access restriction : physically access restriction

Physical security should be considered with great care, if an attacker has physical access to the servers, he owns them: so many attacks are available with local access to the system that it is almost impossible to void them all, without imposing a great burden on the server performance.



NO PHYSICAL SECURITY = NO SECURITY

2.5 Access restriction : Bios and Bootloader

2.5.1 BIOS

By default, BIOS access is not restricted in most computers, which may be an issue in certain cases :

- If the attacker has a physical access to the machine, he can boot on a live cd and access all the disk content in both read write modes without being stopped by file permissions, access controls or being subject to any restriction. It is thus trivial to inject trojans or malware in the system.
- He can also set an admin password which is required at boot time and thus impact on the server availability.

Set an admin **password** on your **BIOS**, and only allow modification if the password is given.

Set HDD as the first boot device, ensuring boot order cannot be manipulated. If possible deactivate Network Boot

The procedure is vendor-specific, so check the BIOS manual for detailed instructions.

If the situation requires it, in some rare cases you may set a boot password which will be required at every boot. If this usage is absolutely logic for a laptop, it may impact the availability of your server as there always should be someone to enter this password.

2.5.2 Bootloader

For similar reasons, your bootloader should be password protected against any change: typically an attacker will try to boot in single mode in order to get full administrative access, boot on an other kernel or an other device.

Password protecting the bootloader (which is most likely Grub on any new installation) is thus of primary importance: a trivial “init=/bin/bash” at the end of the root boot line is enough to bypass any password and gain root access to the system. Using a password will prevent the attacker from tampering the Grub configuration.

Password protecting grub :

First we shall calculate a salted hash of the password: it's not a good idea to use a plaintext password, since should an attacker manage to read /boot/grub/menu.lst he would easily retrieve the password.

```
arik@montmiraille:~$ grub
grub> md5crypt
md5crypt
Password: password1234
password1234
Encrypted: $1$dNSZt$c/k5TILRxiWsc1SxOrfnJ/
```

Now, editing the /boot/grub/menu.lst file , we will add the following line, in the first section, as the first line:

```
password --md5 $1$dNSZt$c/k5TILRxiWsc1SxOrfnJ/
```

This will disable any interactive editing control and entries protected by the keyword lock: for all the entries which do not have to be activated by an unauthorized user add the “lock” specifier as follows :

```
title Boot dos
lock
rootnoverify (hd0,1)
makeactive
chainloader +1
```

If you use the keyword **password** instead of **lock**, then grub will ask for the password at boot time.

Last as menu.lst is by default world readable, make it readable only by root :

```
# chmod 600 /boot/grub/menu.lst
```

2.6 Access restriction : system access restriction, unix user and groups

2.6.1 Users and groups, file permission

On any Unix system, users are identified within the kernel by a unique number called UID (User Identifier), which is then linked to a username. Any user can belong to any number of groups, with a minimum of one. The combination of the username and the set

of groups the user is a member of is leveraged to allow or deny actions (i.e. file access) to the user.

2.6.1.1 File permissions

On a traditional un*x like system a file will have a set of permissions configured for the owner user, for the owner group and for everyone else in the system. Privileges are expressed using 9 bits, which can be easily retrieved using a simple `ls-l` command.

For example a `ls -l` on `texto` will give :

```
-rwx r-x r-- 1 user1 wheel    80 2009-01-28 19:05 texto
```

- : the first dash is signaling the listed file is indeed a file and not a directory, which would result in a `d`.
- `rwx` : the first 3 chars are for the user `user1`, owner of the file: he may read, write or execute this file
- `r-x` : applies to the group, here `wheel`: users member of the group may execute or read the file
- `r--` applies to all the other users of the system: they may only read the file

Only the owner of a file may change its rights: this action is performed through the `chmod` command.

`Chmod` can be used :

- Using the peculiar “x+y” way
 - to allow “other” to execute the file : `chmod o+x file`
 - to remove the right : `chmod o-x file`
 - for the first parameter “u” “g” “o” can be used and “r” “w” “x” for the second one
- with the XXX way calculated according to :

r	w	x	r	w	x	r	w	x
400	200	100	40	20	10	4	2	1

For example, to put file in `-rwx r-x r-x` `chmod 755 file`

Directories can also be `chmoded`, and recursively too by `chmod -R XXX directory`

2.6.1.2 Users and groups

Ownership of a file can be changed by `chown` and `chgrp`, respectively managing the owner user and group.

With rights and groups, we can allow only certain entities to do certain actions.

For example if I want to allow only John, Jack and Willie to reboot my SMS server :

- add john, jack and Willie to a `smsAdmin` group
- make `sms-manager.sh` belong to me and define his group to `smsAdmin`
- `chmod it -rwx -x ---`

For more information on SUID see appendix on `chroot`

2.6.2 Elevation scheme, using sudo

To perform important operation, you need to act as an administrator :

On most **linux** distributions we have a classical user/root model : users have limited privileges and shall “su” to root for privileged operations.

The elevation scheme is :

login as **root** and execute commands OR login a normal user, **su**, execute commands

Ubuntu's developers adopted a different model, based on sudo : for administration purposes, one shall connect as an user member of the admin group and then perform privileged operations with sudo.

The elevation scheme is :

login as a privileged user, than **sudo** command.

This scheme is suitable for a laptop/working computer but may present some risks for a server :

- What happens if the admin (or root) password is compromised (keylogger, shoulder surfing, etc) ? Anyone can directly log in.
- What if the log in mechanism is flawed or if there is a possible attack on the authentication mechanism. (This do happened, think of the OpenSSL Debian issue⁶)

The first thing to assume is that the admin password is always compromised : direct root login out of the console should not be possible (for example via SSH). Of course you will need to log in as administrator to manage your server, but we need to add an extra layer of authentication to do so.

Here we propose a model similar to the BSD one and integrated with the sudo way of Ubuntu : use sudo as it permits a finer grained control and logging facilities while using a wheel group : only members of the wheel group can use sudo.

With this model we have 2 kind of users : normal users and administrators. Only administrators can use sudo (su to root doesn't work on a standard Ubuntu installation), but, out of the local console, logging as administrator is not allowed.

So a user shall first log as a normal user, then su to administrator and then use sudo. If the administrator password is compromised, the attacker will need to compromise an other account to be able to use the password. If he compromises a normal account, he

6 Debian developer “patched” OpenSSH which reduced the entropy of the keys to 16 bit and so permit to “bruteforce” them quickly. More information here : <http://www.metasploit.com/users/hdm/tools/debian-openssl/>

won't have any special rights.

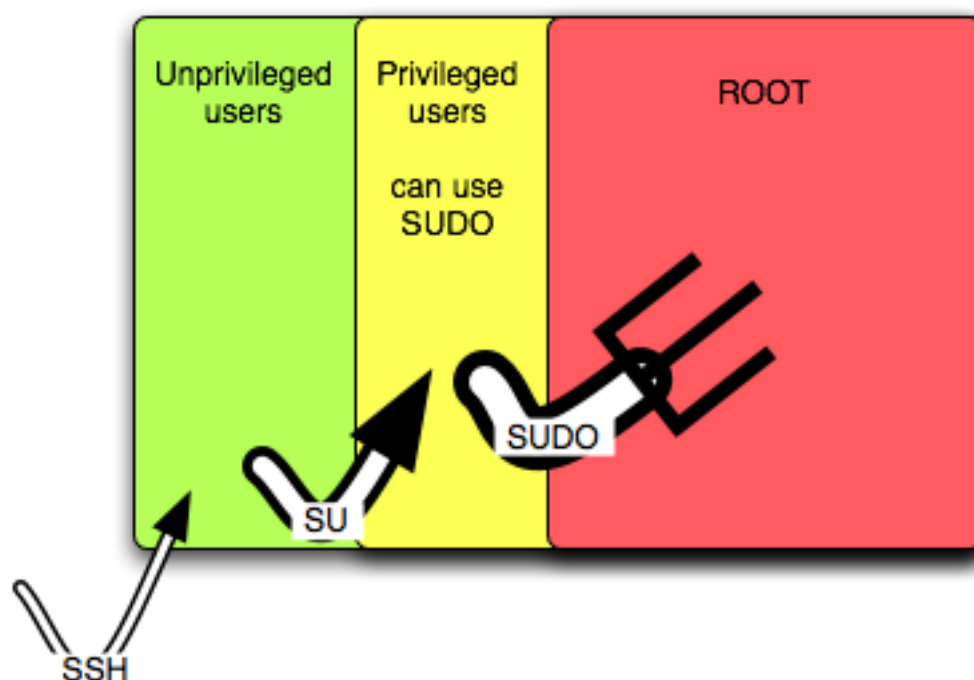


Illustration 8: Privilege elevation scheme

Here, we will set up a model where :

- users shall first log in as normal users
- then su to a privileged account if they need to execute sensible commands.

First add a new user :

```
$ sudo adduser myuser
```

Now we create a wheel group (here, group of users allowed to use su)

```
$ sudo groupadd wheel
```

Then add the users whom need to su, in this case :

```
$ sudo adduser secadmin wheel
```

```
$ sudo adduser myuser wheel
```

Check that your users belongs to the correct groups :

```
$ groups user
```

Now, restrict the use of su to wheel members :

```
$ sudo chgrp wheel /bin/su
```

```
$ sudo chmod 4750 /bin/su
```

Here user myuser can use su but not sudo as he's not in the admin group. To execute a sensible command he first needs to su (switch user) to secadmin, and then use sudo.

► **Privilege elevation schema :**

`ssh myuser@host ; su secadmin ; sudo command`

For tighter control, you can configure the sudoers file more granularly : it is possible to configure rights using vi sudo, for example if you want to delegate the apache administration part to an other administrator without giving him full administration rights or if you want to log actions.

A detailed how to is available on ubuntu website⁷

2.7 Access restriction : about advanced access restriction, SELinux

See appendix : MAC-SELinux

2.8 Access restriction : network access restriction, Firewalling ufw and Shorewall

A firewall is just a software which allows/denies traffic according to a policy. It can have a lot of different uses, but we won't explore the ins and outs of firewalls in this document. Here we will only use it to restrict TCP/IP traffic, in order to shrink the attack surface an Ubuntu server presents to attackers.

A firewall is not always necessary, but it can provide a good security enhancement : if your web server also runs a ssh server, in order to avoid any connection to be made on ssh, you can configure your firewall it to act like :

“Deny all by default

allow traffic to Webserver on ports 80 and 443

allow traffic to ssh on port 22 only from the IT staff room IP”

Someone performing a scan on this machine will only see port 80 open and so will not be able to attack the ssh daemon. While this is a very basic security rule and common sense configuration, an high number of servers on the internet are currently exposing critical services to everyone.

There are 2 policies type : allow all then restrict OR deny all and allow certain things.

⁷ <https://help.ubuntu.com/community/Sudoers>

As it goes for blacklisting in general, it's not a good idea to try to identify all sort of attacks and possible connections. It's far easier to identify everything which should be allowed starting from a deny all perspective in the beginning.

If you allow all then remove certain permissions, an attacker may easily find a way to circumvent the policy.

2.8.1 ufw

For a standard, basic use we recommend *ufw* as your firewall configurator : it is present by default on the system and very easy to configure. Under the hood, ufw will configure the iptables firewall your Ubuntu system is already running.

//First enable ufw :

```
# ufw enable
```

Firewall started and enabled on system startup

//Specify default policy, here will deny all incoming traffic and allow all outgoing

```
# ufw default deny
```

Default policy changed to 'deny'

(be sure to update your rules accordingly)

//Now let see the usage, from man :

```
//ufw [--dry-run] enable|disable
```

```
//ufw [--dry-run] default allow|deny
```

```
//ufw [--dry-run] logging on|off
```

```
//ufw [--dry-run] status
```

```
//ufw [--dry-run] [delete] allow|deny PORT[/protocol]
```

```
//ufw [--dry-run] [delete] allow|deny [proto protocol] [from ADDRESS [port PORT] [to ADDRESS [port PORT]]
```

// Ok let's enable ssh on port 22 and apache on port 80

```
# ufw allow 22
```

Rule added

```
# ufw allow 80
```

Rule added

// Now let see the status

```
root@ubuntu:/home/ary# ufw status
```

Firewall loaded

To	Action	From
----	--------	------

```
--
-----
22:tcp      ALLOW  Anywhere
22:udp      ALLOW  Anywhere
80:tcp      ALLOW  Anywhere
80:udp      ALLOW  Anywhere
```

```
# ufw allow 22 : to allow entering connections on port 22
# ufw allow ssh : equivalent if you use standard ports cf /etc/services
# ufw allow 22/tcp : to allow only tcp
# ufw allow proto tcp from 192.168.229.100 to 192.168.229.78 port 80 : to allow access
only from 192.168.229.100
# ufw delete allow proto tcp from 192.168.229.100 to 192.168.229.78 port 80 : to delete
the previous rule
(address can also be something like w.x.y.z/24)
#ufw allow proto tcp from 192.168.229.100 to any port 22 : to allow ssh access only
from 192.168.229.100
```

You can also modify `/etc/ufw/before.rules` (and `after.rules`) (ex : by default it leaves icmp, multicast and some other stuff, it may not be relevant in your case)

2.8.2 Shorewall

Ufw is perfect for very basic filtering on an Ubuntu system, even if it is possible to install it on a debian system, as for now it is still not packaged for debian and so we rather advice to make your own scripts or use shorewall.

Making a fine grained control home made scripts is an excellent solution, this one⁸ is a good example, but will require a non negligible time investment.

In most of the cases, a specially for a bastion host which is already behind a firewall, using special configuration tools for netfilter will be as efficient as home made scripts while reducing the time investment.

Here we will use a software called shorewall (shoreline firewall) : a high level tool for configuring Netfilter.

2.8.2.1 On netfilter and IPTables

“Netfilter is a framework that provides a set of hooks within the Linux kernel for intercepting and manipulating networks packets”[NF wi].

[8http://www.devarticles.com/c/a/JavaScript/Detecting-and-Countering-Server-Intrusions/](http://www.devarticles.com/c/a/JavaScript/Detecting-and-Countering-Server-Intrusions/)

Netfilter can be used as a firewalling facility, but not only (as for example it can be used for OS fingerprinting disruption).

Iptables is the user space tool used to create the rules. Iptables syntax is very logic but quiet obscure if you are not used to it. For example to allow inbound packets which initiates a HTTP connection, the rule is :

```
iptables -A INPUT -p tcp -j ACCEPT --dport 80 -m state --state NEW
```

2.8.2.2 Shorewall installation and configuration

First install shorewall :

```
# apt-get install shorewall shorewall-doc
```

The shorewall team provides a set of standard rules regarding particular cases. Here we want to build a bastion host

```
# cp /usr/share/doc/shorewall-common/examples/one-interfaces/* /etc/shorewall/
```

```
# cd /etc/shorewall
```

```
# gzip -d *.gz
```

Shorewall is configured with a set of files :

- **zones** : First you define your zones (does make much more sense if you consider the case a firewall with 3 or 4 interfaces where you have the “net”, “FW”, “local” and “DMZ” zones).
Here we have only two zones : fw : the firewall (see it as your computer) and net : the internet (the outside world).
Here nothing to change
- **policy** : where you define the default policy. By default :
from the inside to the outside : ACCEPT
from the outside to the inside : DO NOT ACCEPT
The default policy should always be a **default deny** one
Here nothing to change
- **rules** : where you specify specific rules
- **interfaces** : where you define interfaces related parameters. (if shorewall complains about rfc1918, take of the parameter in the interfaces file)
- **shorewall.conf** : general settings

IMPORTANT : if you are using a private class A, B or C network (10.0..., 172.16... and 192.168...) take of the norfc1918 in /etc/shorewall/interfaces

Aas all inbound traffic is not allowed by default, configuring shorewall consist in

authorizing the entries regarding to the services you want to allow the connection to :

Here you have 2 solutions : use macros which are prebuilt instructions provided by the shorewall team or specify by hand the rules.

Here we will take the example of a webserver :
edit /etc/shorewall/rules and add at the bottom :

#Action	SOURCE	DESTINATION	PROTO	DEST PORT(S)
Web/ACCEPT	net	\$FW		
IMAP/ACCEPT	net	\$FW		
ACCEPT	net	\$FW	tcp	80
ACCEPT	net	\$FW	tcp	143

Here the 2 first lines are equivalent to the two seconds : in the first cases the specifications done via macros and in the second case by hand.

To list the available macros : `# shorewall show macros`
(some examples : FTP, SSH, DNS, POP3, SMB, BitTorrent)

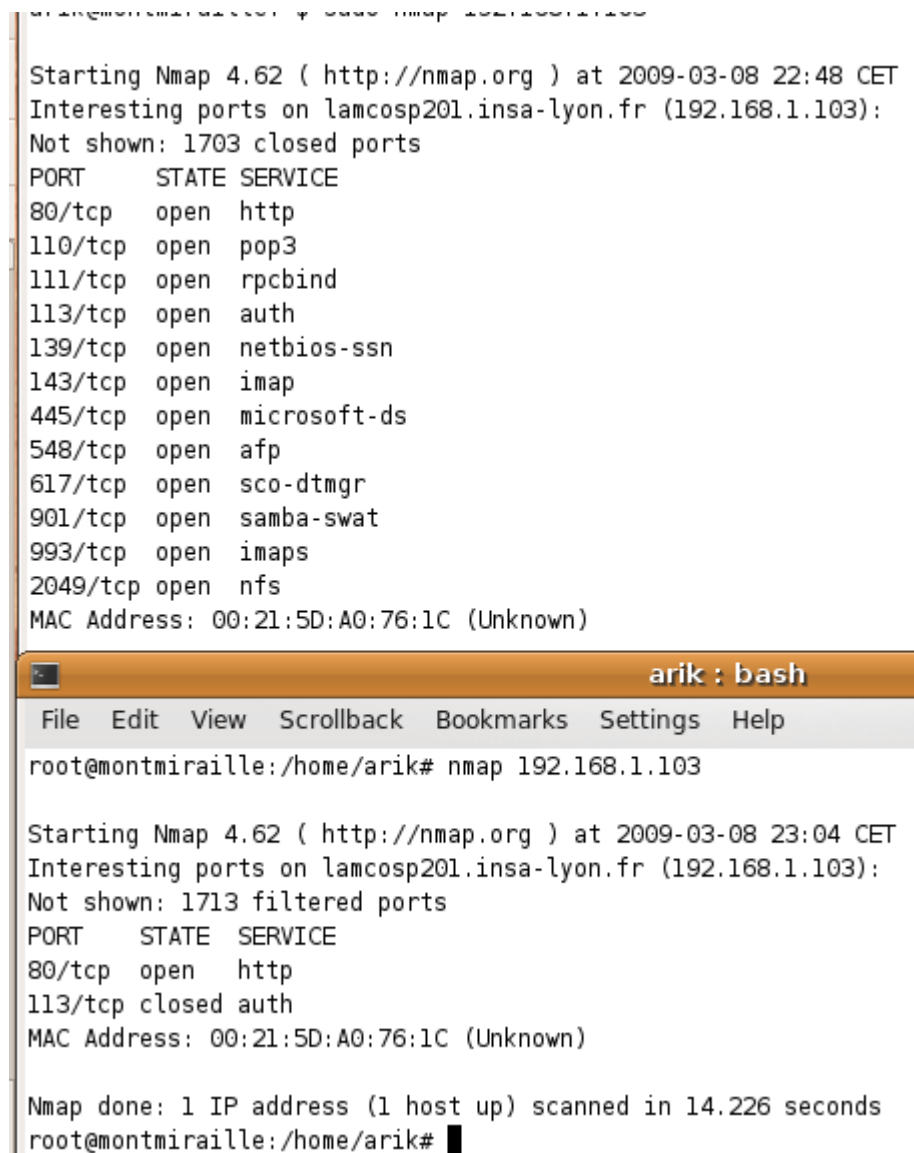
Last edit your `/etc/default/shorewall` and change `startup=0` to `startup=1`
and start shorewall :
`# /etc/init.d/shorewall start`

To test your rules, you can use `nmap`

In this configuration outbound traffic is not strictly restricted, for a higher security, outbound connections can be filtered in a tighter way.

For more details on firewalling and shorewall configuration, the reader may refer to :
[Shorewall bastion]

[Shorewall start]
[Netfilter How to]



The screenshot shows a terminal window with a title bar 'arik : bash'. It displays two nmap scans of 192.168.1.103. The first scan, at 22:48 CET, shows 1703 closed ports and 13 open ports: 80/tcp (http), 110/tcp (pop3), 111/tcp (rpcbind), 113/tcp (auth), 139/tcp (netbios-ssn), 143/tcp (imap), 445/tcp (microsoft-ds), 548/tcp (afp), 617/tcp (sco-dtmgr), 901/tcp (samba-swat), 993/tcp (imaps), and 2049/tcp (nfs). The second scan, at 23:04 CET, shows 1713 filtered ports and only two open ports: 80/tcp (http) and 113/tcp (auth). The terminal output is as follows:

```
Starting Nmap 4.62 ( http://nmap.org ) at 2009-03-08 22:48 CET
Interesting ports on lamcosp201.insa-lyon.fr (192.168.1.103):
Not shown: 1703 closed ports
PORT      STATE SERVICE
80/tcp    open  http
110/tcp   open  pop3
111/tcp   open  rpcbind
113/tcp   open  auth
139/tcp   open  netbios-ssn
143/tcp   open  imap
445/tcp   open  microsoft-ds
548/tcp   open  afp
617/tcp   open  sco-dtmgr
901/tcp   open  samba-swat
993/tcp   open  imaps
2049/tcp  open  nfs
MAC Address: 00:21:5D:A0:76:1C (Unknown)

Starting Nmap 4.62 ( http://nmap.org ) at 2009-03-08 23:04 CET
Interesting ports on lamcosp201.insa-lyon.fr (192.168.1.103):
Not shown: 1713 filtered ports
PORT      STATE SERVICE
80/tcp    open  http
113/tcp   closed auth
MAC Address: 00:21:5D:A0:76:1C (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 14.226 seconds
root@montmiraille:/home/arik#
```

Illustration 9: nmap output before and after firewalling

2.9 Remote access, OpenSSH

OpenSSH permits you to remotely login in a safe way : all the traffic is ciphered which avoids sniffing attacks.

Even if OpenSSH is written with security as main objective, having an SSH daemon running is one more possible open door for an attacker. So install it only if you need remote administration.

Installation :

```
# apt-get install openssh-client  
# apt-get install openssh-server
```

Now we will restrict direct access to privileged users via openssh :

edit `/etc/ssh/sshd_config` change :

```
# vim /etc/ssh/sshd_config  
"PermitRootLogin yes" to "PermitRootLogin no" (without the "")  
and add  
"DenyGroups admin"  
  
and restart ssh daemon :  
# /etc/init.d/ssh restart
```

Change OpenSSH default port (to avoid some automatic attacks and force an attacker to perform a scan)

```
# vim /etc/ssh/sshd_config  
-> Port 22 to Port 21021  
# /etc/init.d/ssh restart
```

:: ;) ::

Short story : password stealing in a university⁹ :

A teacher founds strange that some students had a mark of 04/20 at the first exam and 16/20 in the second. He thinks that those students had access to the exam's subject but he doesn't understand how.

After inquiry (logs, ask student, etc) it appears that the students used their notebook's webcam to film the teacher entering his password.

This doesn't not only happened in films.

:: ;) ::

In an other university : computer in self service room. The computer is locked down : password on the bios, boot only on hard disk, password on grub, compute physically locked and up to date.

But the sysadmin used to install some computers over PXE and forgot to disable it.

And one day a student seeing "pxe" on the screen when rebooting the computer used it to boot on a live cd on an other computer and cat `/boot/grub/menu.lst`.

The password was in clear. It was the same as the bios password... and root password.

⁹ <http://zythom.blogspot.com/2008/09/slap-happy.html> (In French)

Of course one can restrict more the access conditions :
allowing or banning users via :

- **AllowUsers**
- **DenyUsers**
- AllowGroups**
- **DenyGroups.**

The syntax is the same as the above example : to deny an untrusted users group, add “DenyGroups untrusted” in your sshd_config.

Example : allow users toto and tata
edit /etc/ssh/sshd_config and add :
AllowUsers toto tata

If the computer doesn't need to be contacted from every where, limit ssh access to a limited subset of IP

This can be done at 3 levels :

- 1 Packet filtering → See Iptables part
- TCP Wrapper -- /etc/hosts.allow /etc/hosts.deny
- PAM

The simplest way to limit access from certain IP is via the server's firewall, for example with ufw :

```
#ufw allow proto tcp from 192.168.229.100 to any port 22 : to allow ssh access only from 192.168.229.100
```

For more details, see part 4 on acces restriction.

If the risk of being attacked via SSH is very high, don't allow direct SSH access from the internet. If an attacker can't see an open port, he's not able to attack it. For this you can use :

- portknocking (see part 7) : by default the firewall blocks all, and if a client sends a predefined type of messages, the firewall dynamically opens the port for the specified IP.
- a second channel : this is for very high security purpose ; you can connect a cellphone to your computer and use it as a second channel to send orders to your server.

Part 3 : Reducing the attack surface

3 REDUCING THE ATTACK SURFACE

IN THIS CHAPTER WE WILL SEE HOW TO REDUCE AN ATTACKER ATTACK SURFACE. MOSTLY BY DISABLING UNNEEDED DAEMONS, MODULES, SETTING TIGHT MOUNT OPTIONS, TCP/IP STACK AND KERNEL HARDENING.

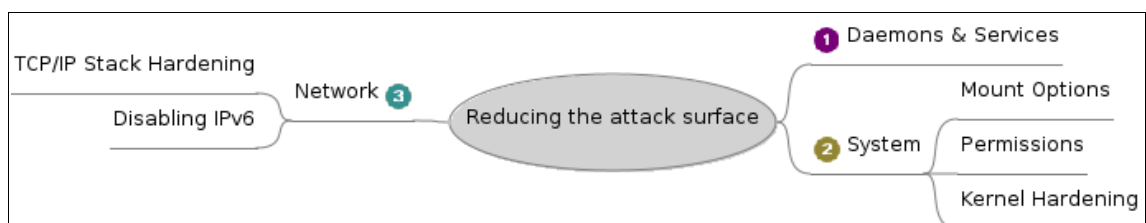


Illustration 10: Chapter 3 : MindMap

3.1 Disabling unneeded daemons and services

Standard installation provides only a few **daemons**, however as they may be used as an **entry point** for an **attacker**, we will **disable** the one which are **not needed**.

Disabling daemons can be done by hand, but it is much more easier via sysv-rc-conf, a tool done for this purpose

SysV Runlevel Config -[: stop service =[: start service h: help q: quit								
service	1	2	3	4	5	0	6	S
apparmor	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
atd	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
bootlogd	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
console-s\$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
cron	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
dbus	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
dns-clean	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
halt	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
keyboard-\$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
killprocs	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
klogd	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Use the arrow keys or mouse to move around. ^n: next pg ^p: prev pg
space: toggle service on / off

Illustration 11: sysv-rc-conf daemon configuration

```
To install :  
# apt-get install sysv-rc-conf  
To launch :  
# sysv-rc-conf
```

Look at runlevel 2 column

```
On a standard install (ubuntu server 8.04) :  
atd -> "run jobs queued for later execution" : May be useful, leave  
cron -> daemon to execute scheduled commands : May be useful, leave  
klogd -> Kernel Log Daemon : leave  
rc.local -> Leave  
rmnologin -> leave  
rsync -> no  
ssh -> if needed  
sysklogd -> leave
```

(configuration for Debian is very similar)

3.2 Checking file permissions and system executables

On modern linux systems, passwords are not stored any more in clear text in /etc/passwd, but you can find a hash in /etc/shadow. If an attack can read /etc/shadow, he can try to attack your password using password cracking tools like john the ripper. That is why we will check that only root and shadow group members can read it.

Check /etc/shadow permissions (on a fresh install permissions should be right, check it if you harden an already installed system) :

```
# ls -l /etc/shadow  
-rw-r----- 1 root shadow 736 2008-09-22 12:53 /etc/shadow
```

Check for unauthorized SUID/SGID System Executables

```
# find /\( -perm -4000 -o -perm -2000 \) -type f -print
```

to remove the set-ID : `chmod -s file`

This section is in orange : by default there are not a lot of SUID/SGID System executables but deactivating them depends on the cases, which has to be done specify

for your system.

On my default installed system, I have :

```
/bin/fusermount
/bin/su
/bin/umount
/bin/ping6-----> if no ipv6 needed
/bin/mount
/bin/ping
/var/local
/var/lib/libuuid
/var/log/news
/var/mail
/var/cache/man
/usr/src
/usr/local/lib/python2.5
/usr/local/lib/python2.5/site-packages
/usr/bin/expiry
/usr/bin/wall -----> not really used nowadays
/usr/bin/traceroute6.iputils
/usr/bin/sudoedit
/usr/bin/passwd
/usr/bin/mtr
/usr/bin/chsh
/usr/bin/chage
/usr/bin/ssh-agent
/usr/bin/newgrp
/usr/bin/sudo
/usr/bin/gpasswd
/usr/bin/at
/usr/bin/crontab
/usr/bin/mlocate
/usr/bin/chfn
/usr/bin/arping
/usr/bin/bsd-write
/usr/lib/openssh/ssh-keysign
/usr/lib/pt_chown
/usr/lib/eject/dmccrypt-get-device
/usr/sbin/pppd -----> Is it needed
/usr/sbin/uudd
/lib/dhcp3-client/call-dhclient-script
/etc/ppp/peers
/etc/chatscripts
/sbin/unix_chkpwd
```


Find files without any owner :

```
# find /\( -nouser -o -nogroup \) -print
```

Normally you should not find something, if it is the case you'll need to investigate it.

Check that `randomize_va_space` is 1 (default ok) :

```
root@gamon:/proc/sys/kernel# sysctl kernel.randomize_va_space
kernel.randomize_va_space = 1
```

check if Execute Disable (XD) or No Execute (NX) is supported (x86 systems)

\$ cat /proc/cpuinfo and search for “pae” or “nx” flags

(by default the ubuntu kernel supports pae)

3.3 Deleting or disabling unneeded user accounts

This step is not really mandatory on a freshly installed system, but is useful if you harden an existing one. First check that only legitimate accounts are present : user who can't log has a * after the first :

If there is text, that means that they can log in (usually with a password), so check them carefully. Having myuser on a standard install is ok. Check if there are not old accounts or testing/dev accounts (which may have a weak password).



USE ONLY WHAT IS NECESSARY, ADD FEATURES
ONLY WHEN THEY ARE NEEDED. IF POSSIBLE AVOID
WEB INTERFACES.

Look in `/etc/shadow`, for every line with a * after the first “:” (* or ! Means that the user can not login with his password), modify your `/etc/passwd` like this :

In shadow :

```
syslog:*:14148:0:99999:7:::
```

In passwd modify to have :

```
syslog:x:102:103::/home/syslog:/bin/false
```

By default all non users account are blocked with *, but it is better to lock passwd too.

For further readings, refer to this link.¹⁰

3.4 TCP/IP Stack hardening

Detailed explications are available on Cromwell's web site¹¹

Ubuntu has by default a lot of security options configured in a safe way

You need to edit your sysctl.conf , and make the following changes :

```
# vim /etc/sysctl.conf
add the following lines :
#ICMP redirect should not be used as it may permit to inject routes ; disable them
net.ipv4.conf.all.accept_redirects=0
net.ipv6.conf.all.accept_redirects=0
net.ipv4.conf.all.send_redirects=0
net.ipv6.conf.all.send_redirects=0

#Log "strange packets with source/destination invalid" ; this should also log rejected
#source routed packets and spoofed packets
net.ipv4.conf.all.log_martians=1

#Last enable syncookies to protect against synflooding
net.ipv4.tcp_syncookies=1
```

Note : some users reported that the sysctl.conf way didn't work on debian, the solution is to put a script in /etc/network/if-up.d/ with

```
#!/bin/sh
echo 0 > /proc/sys/net/ipv4/conf/all/accept_redirects
and so on. (the previous line corresponds to net.ipv4.conf.all.accept_redirects=0)
```

¹⁰http://ferry.eof.eu.org/lesjournaux/11/public_html/ch16s03.html

¹¹<http://www.cromwell-intl.com/security/security-stack-hardening.html>

3.4 Disabling ipv6

If you are not using ipv6 (most of the cases now day) disable it as it may be a potential vector for attacks.

```
# vim /etc/modprobe.d/aliases and coment this line
alias net-pf-10 off
#alias net-pf-10 ipv6

add in /etc/modprobe.d/blacklist
blacklist ipv6

check :
lsmod | grep ipv6
```

3.5 Filesystem mounting options

Mount options can be used to reduce the way an attacker can use in case the system is compromised. In some case it will prevent the attack, in other no ; here he goal is just to reduce the attacks ways.

Nodev : a file can not be used as a device. In /tmp, /var and /home, it should not happen on a normal system.

noexec : do not execute binaries. For a standard use, binaries should not be executed in /tmp and /var

nosuid : do not execute suid binaries. For a standard use, SUID binaries should not be esecuted in /tmp and /var or /home

Edit /etc/fstab :

#vim /etc/fstab

an add for each partition :

/tmp	nosuid,nodev,noexec
------	---------------------

/var	nosuid,nodev,noexec
------	---------------------

/home	nosuid,nodev
/usr	nodev

nosuid,nodev,noexec on /var, may leave you unable to use some applications such as apt or VMware. Setting nosuid, nodev, noexec on /var should be done only after proper testing.

3.6 Misc and Kernel Hardening

Check core dump are disabled : `sysctl fs.suid_dumpable`
0 by default : ok

For more information on kernel hardening see Appendix G : Kernel Hardening with GRSecurity/PaX

Part 4 : Monitoring and detecting intrusions

4. MONITORING AND DETECTING INTRUSIONS

ONCE THE ATTACK SURFACE REDUCED TO IT'S MINIMUM, WE SHALL MONITOR THE EXPOSED ONE. IN A FIRST TIME WE WILL INSTALL AN ALERT MECHANISM THEN INSTALL SOME HOST BASE INTRUSION DETECTION SYSTEMS.

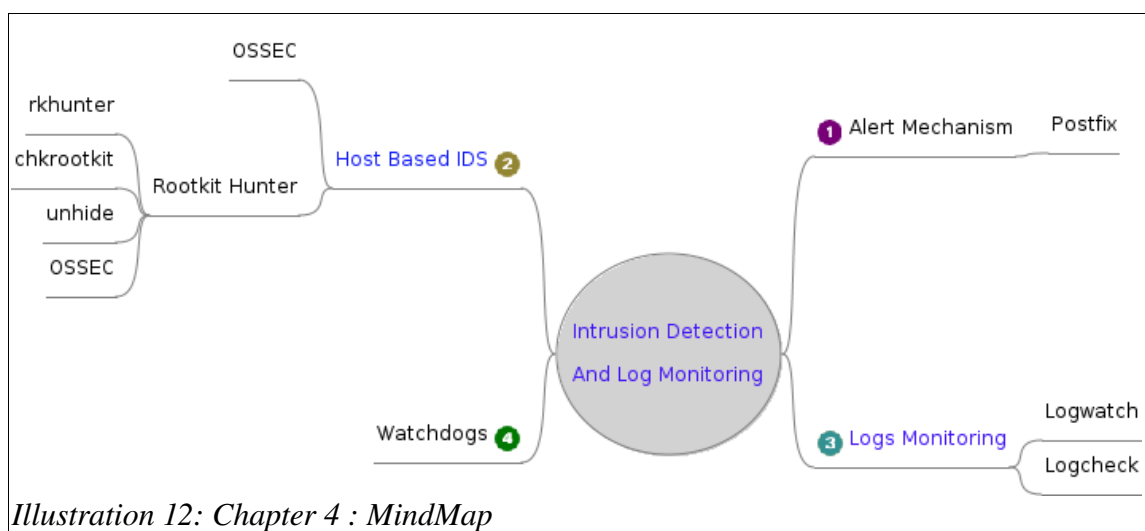


Illustration I2: Chapter 4 : MindMap

Note that this part focuses on a *very basic* configuration, a bare minimum and need to be adapted to your needs

4.1 Alert transport mechanism : Postfix

Some softwares will need to send reports or alerts, which are usually sent by mail. As the default install do not provide a MTA, we will install postfix. We chose postfix for his easy configuration and his good security records.

As we only need to send mail and not receive any message, we will configure it to listen only on the loopback interface. This is done for security reasons : leaving a daemon listening on port 25 which is not needed may provide a potential entry point for an attacker.

```

First, install postfix
# apt-get install postfix
A configuration screen will appear, select none

# apt-get install mailx

//Now create a file /etc/postfix/main.cf with rights -rw-r--r-- root root
$sudo touch /etc/postfix/main.cf
$sudo chmod 611 /etc/postfix/main.cf

//Copy past this in /etc/postfix/main.cf
#####
#
# See /usr/share/postfix/main.cf.dist for a commented, more complete version


# Debian specific: Specifying a file name will cause the first
# line of that file to be used as the name. The Debian default
# is /etc/mailname.
#myorigin = /etc/mailname

smtpd_banner = $myhostname ESMTP $mail_name (Ubuntu)
biff = no

# appending .domain is the MUA's job.
append_dot_mydomain = no

# Uncomment the next line to generate "delayed mail" warnings
#delay_warning_time = 4h

readme_directory = no

# TLS parameters
smtpd_tls_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
smtpd_tls_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
smtpd_use_tls=yes
smtpd_tls_session_cache_database = btree:${data_directory}/smtpd_scache
smtp_tls_session_cache_database = btree:${data_directory}/smtp_scache

# See /usr/share/doc/postfix/TLS_README.gz in the postfix-doc package for
# information on enabling SSL in the smtp client.

myhostname = ubuntu
alias_maps = hash:/etc/aliases

```

```
alias_database = hash:/etc/aliases
mydestination = ubuntu, localhost.localdomain, localhost
relayhost =
mynetworks = 127.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128
mailbox_size_limit = 0
recipient_delimiter = +
inet_interfaces = loopback-only
default_transport = smtp
relay_transport = smtp
inet_protocols = ipv4
#####
```

Here in fact it is a “local” setup with possibility to send mails :

```
default_transport = smtp
```

```
relay_transport = smtp
```

With postfix only listening on local interface :

```
inet_interfaces = loopback-only
```

You can change change the hostname :

```
myhostname = ubuntu
```

Note : when apt-get install postfix, it will also install :
openssl openssl-blacklist postfix ssl-cert

Restart postfix

```
# /etc/init.d/postfix restart
```

As for now mails sent to root will be locally delivered, which may not ne so convenient.

We will instruct the system to send them to user@yourdomain.com by editing /etc/aliases and add

```
postmaster: root
```

```
root: user@yourdomain.com
```

```
#sudo newaliases
```

Now test it by :

```
$ mail -s “Test” you@yourdomain.com
```

```
TEST !
```

```
.
```

```
$
```


4.2 Host based Intrusion detection systems

4.2.1 OSSEC

OSSEC is an host based intrusion detection system which performs log analysis, file integrity checking, rootkit detection and active response.



AN HOST BASED INTRUSION DETECTION SYSTEM IS
WARMLY ADVISED BUT DO NOT CONFIGURE IT TO THROW TOO
MUCH ALARMS : DO REMEMBER THAT THE PSYCHOLOGICAL
ASPECT OF SECURITY IS OF FIRST IMPORTANCE, TOO MANY
WARNINGS WILL FINISH IN THE SPAM FOLDER.

OSSEC runs on GNU/Linux (Ubuntu, Debian, Redhat, Gentoo, etc) Open/Free/NetBSD, solaris, aix, hp ux, MacOS X, Vmware ESX and Windows (2000, XP, 2003, Vista and 2008) which represent a big advantage when managing a heterogeneous parc of machines.

OSSEC can work in agent/server mode or in local mode.

OSSEC does the same things as a combination of tools such as logcheck, chkrootkit, etc but all integrated in one and has active response capacities. This last point if well configured can be useful : it permits to execute certain commands when certain triggers are activated. One side it permits you to react as soon as the attack happens, thus limiting the damages ; but on the other a false positive may block your system and an attacker can use it to DOS your system.

```
# apt-get install build-essential
Get the last version on http://www.ossec.net/main/downloads
# wget http://www.ossec.net/files/ossec-hids-1.6.1.tar.gz
# tar zxvf ossec-hids-1.6.1.tar.gz
# cd ossec-hids-1.6.1
# ./install.sh
Follow install instructions, here are one for a single host :
en | local | accept default options, do not enable active response (this option need to be
tested)
start ossec :
#/etc/init.d/ossec start
```

```
Uninstall build-essential :  
#apt-get remove --purge build-essential
```

Now you should have this process running :

```
ossecm      /var/ossec/bin/ossec-maild  
root        /var/ossec/bin/ossec-execd  
ossec       /var/ossec/bin/ossec-analysisd  
root        /var/ossec/bin/ossec-logcollector  
root        /var/ossec/bin/ossec-syscheckd  
ossec       /var/ossec/bin/ossec-monitord
```

4.2.2 Other HIDS

The reader may consider other HIDS such as samhain, Tripwire or AIDE.

For more information, the reader is invited to consult the appendix “More HIDS”

4.2.3 Rootkits hunters

4.2.3.1 Rkhunter

Rkhunter will check for the presence of rootkits and backdoors via md5 signatures and triggers (anomalities, ex interface in promiscuous mode).

```
SunOS / NSDAP Rootkit      [ Not found ]  
Superkit Rootkit          [ Not found ]  
TBD (Telnet BackDoor)     [ Not found ]  
TeLeKiT Rootkit           [ Not found ]  
TOrn Rootkit              [ Not found ]  
Trojanit Kit              [ Not found ]  
Tuxendo Rootkit           [ Not found ]  
URK Rootkit               [ Not found ]  
VcKit Rootkit             [ Not found ]  
Volc Rootkit              [ Not found ]  
X-Org SunOS Rootkit       [ Not found ]  
zaRwT.KiT Rootkit         [ Not found ]  
  
Performing additional rootkit checks  
Suckit Rootkit additional checks [ OK ]  
Checking for possible rootkit files and directories [ None found ]  
Checking for possible rootkit strings [ None found ]  
  
Performing malware checks  
Checking running processes for suspicious files [ None found ]  
Checking for login backdoors [ None found ]  
Checking for suspicious directories [ None found ]  
Checking for sniffer log files [ None found ]  
  
Performing Linux specific checks  
Checking kernel module commands [ OK ]  
Checking kernel module names [ OK ]  
  
Press <ENTER> to continue]
```

Illustration 13: rkhunter --check

```
# apt-get install rkhunter
Cron job :
Edit /etc/rkhunter.conf
Uncomment :
MAIL-ON-WARNING=you@yourdomain.com
USE_SYSLOG=authpriv.notice
# rkhunter --update
# rkhunter -c --createlogfile
```

Now launch it with “--cronjob --report-warnings-only” options : no interactive, only warnings. By default on Ubuntu Server 8.04 you have this warnings :

```
root@ubuntu:/etc/postfix# rkhunter -c --cronjob --report-warnings-only
Warning: Hidden directory found: /dev/.static
Warning: Hidden directory found: /dev/.udev
Warning: Hidden directory found: /dev/.initramfs
```

One or more warnings have been found while checking the system.
Please check the log file (/var/log/rkhunter.log)
root@ubuntu:/etc/postfix#

We'll have to whitelist them :
#vim /etc/rkhunter.conf

```
Search for those and uncomment :
#ALLOWHIDDENDIR=/etc/.java
#ALLOWHIDDENDIR=/dev/.udev
#ALLOWHIDDENDIR=/dev/.udevdb
#ALLOWHIDDENDIR=/dev/.udev.tdb
#ALLOWHIDDENDIR=/dev/.static
#ALLOWHIDDENDIR=/dev/.initramfs
#ALLOWHIDDENDIR=/dev/.SRC-unix
```

(If you use unhide, /usr/sbin/unhide and /usr/sbin/unhide-linux26, will generate warnings
rkhunter --propupd to fix this- only in a clean system)

Now you have a choice between :

- Heart beat like report : each day a mail, even when no abnormalities are detected. A good solution if you manage a few machines, but impossible to use if there are more than a few machines.

- Anomaly report : get a mail only when there is a problem detected... fine, but when an attacker owns your machine, he will certainly disable your HIDS and rootkit hunters.

Edit /etc/cron.daily/rkhunter, and replace the content by :
#!/bin/sh
/usr/bin/nice -10 /usr/bin/rkhunter --cronjob --report-warnings-only

4.2.3.2 Chkrootkit

Chkrootkit is an other rootkit hunter which is complementary to rkhunter.

```
Checking `su'... not infected
Checking `ifconfig'... not infected
Checking `inetd'... not infected
Checking `inetdconf'... not infected
Checking `identd'... not found
Checking `init'... not infected
Checking `killall'... not infected
Checking `ldsopreload'... not infected
Checking `login'... not infected
Checking `ls'... not infected
Checking `lsof'... not infected
Checking `mail'... not infected
Checking `mingetty'... not found
Checking `netstat'... not infected
Checking `named'... not found
Checking `passwd'... not infected
```

Illustration 14: Chkrootkit at work

```
#apt-get install chkrootkit
```

to make a test :

```
# chkrootkit -q
```

4.2.3.3 Unhide

“unhide is a forensic tool to find processes hidden by rootkits, Linux kernel modules or by other techniques. It detects hidden processes using three techniques:

The proc technique consists of comparing /proc with the output of /bin/ps.

The sys technique consists of comparing information gathered from /bin/ps with information gathered from system calls.

The brute technique consists of bruteforcing the all process IDs. This technique is only available on Linux 2.6 kernels.

unhide-tcp is a forensic tool that identifies TCP/UDP ports that are listening but are

not listed in /bin/netstat through brute forcing of all TCP/UDP ports available”
From the man page of unhide and unhide-tcp

```
root@ubuntu8:~# unhide sys
Unhide 20080519
yjesus@security-projects.com

[*]Searching for Hidden processes through getpriority() scanning
[*]Searching for Hidden processes through getpgid() scanning
[*]Searching for Hidden processes through getsid() scanning
[*]Searching for Hidden processes through sched_getaffinity() scanning
[*]Searching for Hidden processes through sched_getparam() scanning
[*]Searching for Hidden processes through sched_getscheduler() scanning
[*]Searching for Hidden processes through sched_rr_get_interval() scanning
[*]Searching for Hidden processes through sysinfo() scanning

root@ubuntu8:~# █
Illustration 15: Unhide sys at work
```

```
# apt-get install unhide
```

An interesting feature of unhide is that it can work with rkhunter : remove “hidden_procs” from DISABLE_TEST in /etc/rkhunter.conf XXXXX TEST

```
#!/bin/bash
unhide brute
unhide proc
unhide sys
unhide-tcp
```

4.3 Monitoring logs

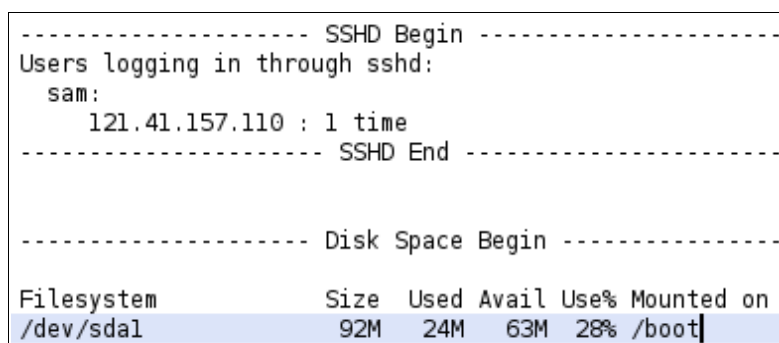
A server generates a large quantity of logs, which can not be well processed by a human being.

Log watcher are software which will parse system logs and alert the administrator according to preset rules ; we will see how to install and configure 3 of them : logwatch, logcheck (and OSSEC).

5.2.1 Logwatch

First we will install, if relevant regarding the situation, logwatch. Logwatch will make a report at given intervals of time, by default it will report a status on :

- Cron
- iptables
- pam
- postfix
- ssh
- Disk space



```
----- SSHD Begin -----
Users logging in through sshd:
  sam:
    121.41.157.110 : 1 time
----- SSHD End -----

----- Disk Space Begin -----

Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1        92M   24M   63M   28% /boot
```

Illustration 16: An extract of a Logcheck report

Logwatch installation (**optionnal**) :

```
# apt-get install logwatch
```

5.2.2 Logcheck

Logcheck parses logs and raises alarms if an anomaly is discovered, it is a very useful tool.

To install :

```
apt-get install logcheck
```

Then edit /etc/logcheck/logcheck.conf if you want to change some configuration options, default settings are ok. (if you correctly set the mail alias for root)

Change SENDMAILTO="you@yourdomain.com"

Logcheck will raise a lot of alarms, and most of all a lot of regular alarm. To avoid this, you may add a personal file in /etc/logcheck/ignore.d.server/

In this file you can specify exceptions based on regular expressions :

Here is a **quick and dirty hack** I used on the test machine :

```
^.*ntpd\[0-9]+\):.*$
^.*pam_unix.*cron:session.*$
^.*UFW BLOCK INPUT.*$
^.*usr/sbin/fcheck && if ! /usr/sbin/fcheck -asxrf /etc/fcheck.*$
^.*postfix/smtp.*to=<me@gmail.com>.*$
^.*USR/SBIN/CRON.*$
^.*-- MARK.*$
^.*usr/local/rtm/bin/rtm 20 > /dev/null 2.*$
^.*IPv6 addrconf: prefix with wrong length 5.*$
^.*Reading included configuration file: /etc/xinetd.d/.*$
^.*postfix/smtp.*$
^.*dev/vmmon.*$
^.*dev/vmci.*$
^.*perl: warning:.*$
^.*ks postfix/pickup.*$
^.*ks postfix/cleanup.*$
^.*ks postfix/qmgr.*$
^.*ks postfix/local.*$
```

Advanced HIDS :

For advanced HIDS, you may write your own scripts which will be much more difficult to find and stop before they send the alarm.

Some tricks :

- send its log to a remote server for remote checking
- Hide it, as a backup script, as a [pdflush process], etc¹²

¹² How to change a UNIX process and child process name by modifying argv[0] <http://www.uofr.net/~greg/processname.html>

```
int argv0size = strlen(argv[0]); //Take note of how many chars have
been allocated
```

```
...
    strncpy(argv[0], "main-thread-name", argv0size); //replace argv[0]
[0..argv0size]
...
```

4.4 Watchdogs

For specific purposes, you can set watchdogs, special alarm raisers. Some examples :

- IP watchdog : watches after a server and regularly pings it, will raise an alarm if he can not ping it
- defacement watchdog : a watchdog can regularly monitor a webpage looking for a defacement

```
fork(); //make child process
...
strncpy(argv[0],"child-thread-name",argv0size); //replace argv[0]
[0..argv0size]
```


5. KEEPING UP TO DATE AND INFORMED

KEEPING A SYSTEM UP TO DATE IS OF TREMENDOUS IMPORTANCE, AN OUTDATED SYSTEM IS PRONE TO BE ATTACKED USING KNOWN VULNERABILITIES WHICH ENLARGES THE ATTACK PERIMETER.

UPDATING IS PART OF THE MANAGEMENT PROCESS, BUT STANDS ON TECHNICAL MEANS. IN THIS CHAPTER WE WILL SEE SOME BASIS ON HOW TO KEEP UP TO DATE AND HOW TO KEEP INFORMED ON NEW ATTACKS.

5.1 Keeping Up To Date

2.2.1 Updating with apt

Installing, removing and keeping the system up to date is done via the **apt- or aptitude commands**. Here is a short overview of some useful commands :

First you need to get the last version of the packages list :

```
$ sudo apt-get update
```

To upgrade your system :

```
$ sudo apt-get upgrade
```

To upgrade your system and install last versions of the dependencies :

```
$ sudo apt-get dist-upgrade
```

To install a new packet, here vlc :

```
$sudo apt-get install vlc
```

To remove a packet :

```
$ sudo apt-get remove vlc
```

To remove it and remove the configuration files :

```
$ sudo apt-get remove --purge vlc
```

To search for a package :

```
$sudo apt-cache search vlc
```

2.2.2 Automatic notifications

To have **automatic notification** of the need to update a system, a cron job can be set or the reader can use cron-apt depending on the number of server you have to manage.

The **cron-apt** solution is perfect for a **few servers** (up to 10), for **more** than 10 servers an automated or a **managed solution** should be used.

Which may be :

- management solution : decide to update regularly all the servers at the same period of the month/week. With this solution all servers should be on the same state.
- Technical solution : use a dedicated solution or make your own (cronjobs + mails)



**A MEDIUM HARDENED SYSTEM REGULARLY UPDATED
IS MUCH BETTER THAN AN EXTREME HARDENED ONE
UPDATED EACH 3 YEARS.**

cron-apt will regularly update packages list and download new packages without installing them, so logging in to the system and performing an apt-get upgrade is required.

If logcheck is installed you will receive a mail with :

```

System Events
==--==
Feb  9 04:44:39      cron-apt: CRON-APT RUN [/etc/cron-apt/config]: Mon Feb  9 04:00:01
CET 2009
Feb  9 04:44:39      cron-apt: CRON-APT SLEEP: 2677, Mon Feb  9 04:44:38 CET 2009
Feb  9 04:44:39      cron-apt: CRON-APT ACTION: 3-download
Feb  9 04:44:39      cron-apt: CRON-APT LINE: /usr/bin/apt-get dist-upgrade -d -y -o
APT::Get::Show-Upgraded=true
Feb  9 04:44:39      cron-apt: Reading package lists...
Feb  9 04:44:39      cron-apt: Building dependency tree...
Feb  9 04:44:39      cron-apt: Reading state information...
Feb  9 04:44:39      cron-apt: The following NEW packages will be installed:
Feb  9 04:44:39      cron-apt:  libisc35 openssh-blacklist
Feb  9 04:44:39      cron-apt: The following packages will be upgraded:
Feb  9 04:44:39      cron-apt:  bind9-host dnsutils libbind9-30 libdns35 libisccc30
libisccfg30
Feb  9 04:44:39      cron-apt:  openssh-client openssh-server
Feb  9 04:44:39      cron-apt: 8 upgraded, 2 newly installed, 0 to remove and 0 not
upgraded.
Feb  9 04:44:39      cron-apt: Need to get 0B/4149kB of archives.
Feb  9 04:44:39      cron-apt: After this operation, 4751kB of additional disk space will be
used.
Feb  9 04:44:39      cron-apt: Download complete and in download only mode

```

Illustration 17: Cron-apt

```

To install cron-apt :
apt-get install cron-apt
Default setting are should suffice for a standard use

```

If you don't want to use cron-apt, you can use a very simple script like :

```

#!/bin/bash
#
# Cron Script - run from /etc/crontab or /etc/cron.daily
#
# Checks if update are available by simulating an apt-get upgrade

if [ -n "`apt-get --simulate upgrade | grep Inst`" ]
then
    echo "Hello, I need updates" | mail -s "Updates availbale : `uname -n`"
yourname@yourdomain.com
fi

```

and add to your /etc/crontab the following line to check for update every day at 18h30

```
30 18 * * * /pathToScript
```

5.2 Informed

The reader may consider to subscribe to some mailing lists on which new vulnerabilities are announced :

For Debian <http://lists.debian.org/debian-security-announce/>
For Ubuntu <https://lists.ubuntu.com/mailman/listinfo/ubuntu-security-announce>

Bugtraq, Full disclosure are recommended too for general announce. (high volume of exchanged messages)

Other Mailing lists shall be of some interest for the reader :

DailyDave

<http://sikurezza.org/> lists (In Italian)

<http://www.ossir.org/listes/index.shtml> lists (In French)

Some websites and Blogs are listed in appendix F.

CHAPTER 6 : MITIGATION AND CONFINEMENT

UP TO NOW WE FOCUSED ON CONTROLLING ACCESS AND REDUCING THE ATTACK SURFACE, MAKING IT HARDER FOR AN ATTACKER TO ENTER OUR INFORMATION SYSTEM. AFTERWARDS, WE SET UP A MONITORING SYSTEM WHICH AIMS AT ALERTING THE ADMINISTRATORS WHEN IT DETECTS A BREAK IN.

IN THIS CHAPTER WE WILL FOCUS ON MITIGATION AS A PART OF A GLOBAL DEFENSE IN DEPTH STRATEGY.

THE MAIN IDEA IS, CONSIDERING THE ATTACKER MANAGED TO BREAK IN, TO MITIGATE HIS ACTION AND TO LIMIT THE DAMAGES HE CAN DO.

6.1 Intro

Let us consider the simple example of a compromised corporate mail server. Thanks to a misconfiguration, a software flaw or a human error, an attacker managed to have full admin rights on a corporate mail server. Of course if the intrusion detection system is correctly configured, the administrators will quickly be alerted and will handle the situation. But what if it takes 3 hours to make a quick fix ? It means that the intruder had access to all the mails from the CEO strategic plans to some technical details on a new product.

Now if this company had encrypted all of the mail traffic, even in the case of a successful break in, the information leakage would be strongly mitigated.

There are a lot of mitigation techniques for a lot of different purposes. Here we will focus on service isolation (mostly daemons), both at the confidentiality and availability levels, which represents an important issue.

Of course the simplest way to do so is to use a dedicated physical machine for each service on a dedicated network. For high security environments this is likely the best solution, but for most of the situations this is simply too expensive.

Furthermore, using strict isolation on all services can improve security even on single purpose servers, since we can still isolate the daemon from the operating system itself. We will focus on two types of solutions : Access Control like (ex SELinux) and virtualization like.

Note on chroot :

Chroot is just a syscall which changes the root directory of a process: at the beginning it was written for testing purposes, not security purposes.

Chroot is not a security tool and should not be considered as one. In some cases, however, when the software is really well written with security in mind (ex openssh) it can be used for security purposes. For more information on the matter, see the appendix on chroot.

Note on systrace :

Systrace can be used as a syscall firewall to sandbox untrusted softwares or network daemons.

Systrace will intercept syscalls made by a software and allow or deny them according to a policy. While setting it up under strict requirements is not difficult, it takes a good knowledge to configure it well.

It can still be useful to box network daemons, for example I used it to restrict Apache allowing it to only to access in read mode a few directories.

An how to on using systrace can be found at the address in the note.¹

6.2 Access control

In this section we will focus on Mandatory Access Control and Role Based Access Control, and describe some solutions.

While this kind of solutions are mature and used since years in the military and government context, they are not largely deployed on the civilian market, apart from some selected, high security companies.

Even if those solutions have improved a lot in the last years, configuring them is still difficult and requires experienced administrators.

Of course a lot of policies are already available for standard softwares, for example the SELinux targeted policies of SELinux for apache. But when you want to use specific software or use MLS, the task becomes much harder and requires some considerable experience.

Maintenance is also harder to perform as it requires at least some training and good documentation, a point which becomes of critical importance in the case of turnover in the sysadmin teams.

Here we will consider a few solutions, their advantages and disadvantages¹³.

6.2.1 SELinux

SELinux is a set of modification proposed by the NSA to provide Mandatory Access Control on the Linux platform.

¹³ See <https://wiki.ubuntu.com/AppArmor> on MAC problems

SELinux adds to the linux kernel a Mandatory Access Control which provides information separation and threats tempering. Its most interesting features is containment : with well defined policies, it can contain a lot of attacks.

A policy is a set of rules describing what a given user or process can actually do on the system. These rules supercede any permission or properties at the file system level, being even able to somehow restrict the superuser (i.e. root). For instance, if the MySQL user is not explicetly allowed to do so, it won't be able to write on the /tmp directory, which is usually world writable.

As a result, even if an attacker takes control of a daemon, his actions will be restricted to what the policy allows him to do.

If used in strict mode, even if the attacker manages to attain root privileges, he will be restricted to a specific role. If the security policies are well configured, he can not do any damage.

If you want to test such a case, Russel Coker provides a SELinux Debian based play machine where you can connect as root¹⁴

SELinux responds to problems such as "I need to run our legacy application as root, but only access to /var/customer_data with read rights and no other part of the system such as /var/mail".

The big problem is that, if it is not configured well enough, this kind of control can provide a false sense of security where there is none.

For example during the 2009 French Symposium on security (SSTIC09), a speaker presented a solution for a secure computer for home users (project SEC&SI). As for an example he shown us that even if an user had the classical unix rights on "test.sh" which allowed him to execute it, he could not. He tested ./test.sh and nothing happened. At the end of the talk one of the attendee asked to try "sh test.sh", which worked.

Of course this was an example for a talk, but this kind of issue can also happen in real life, on production system.

Also, don't forget that while SELinux is able to block some root exploits¹⁵ it will not stop all of them. The play machine was owned shortly after the publication of the vmsplce exploit¹⁶, but as a very good example of defense in depth it was contained by the underlying xen.

14 <http://www.coker.com.au/selinux/play.html>

15 <http://doc.coker.com.au/computers/se-linux-saves/>

16 <http://etbe.coker.com.au/2008/04/03/trust-and-play-machine/>

SELinux supports 4 different kind of policies¹⁷ :

- Strict : Where everything is denied by default and where you need to specify rules for each action. An excellent choice at a strict security consideration, but difficult to configure and manage. Due to its long and difficult configuration, most people choose not to use it, and we tend to agree.
- Targeted : In this case, the idea is to lock down only a few specific domains, and leave the userspace unconfined. These policies, as long as you stick to a standard use, are easy to set up, especially for network daemons such as apache.
- MLS : Which allows the use of security levels. For high security purpose, hard to configure and maintain.
- Minimum : like targeted, but optimized for low memory systems

Last but not least SELinux is present by default on all Ubuntu server kernels, not requiring any recompilation and thus being suitable for mass deployment.

Note that while SELinux is very well supported under Red Hat-like operating systems, and at a lower level on Debian, it is not exactly mature on Ubuntu server.

Here we shall take the example of confining apache on a Debian system :¹⁸

Part1 : SELinux Install - Remove apparmor :

```
apt-get remove apparmor
```

```
# apt-get install selinux
```

edit your /boot/grub/menu.lst and add selinux=1 at the end of the kernel line

Reboot your system (needed for relabeling, that is having the kernel add the labels SELinux leverages to implement its security restrictions)

If you plan to write you own policies , edit /etc/selinux/config and set

```
SELINUX=enforcing
```

```
SELINUXTYPE=refpolicytargeted
```

SELinux' status can then be checked out by running

```
# sestatus v
```

In order to retrieve all the installed SELinux policy modules

```
/usr/share/selinux/refpolicy# semodule l
```

```
authlogin 1.9.1
```

```
cups 1.9.0
```

```
getty 1.5.0
```

```
inetd 1.6.0
```

17 <http://fedoraproject.org/wiki/SELinux/Policies>

18 For more details see <https://help.ubuntu.com/community/SELinux>
<https://wiki.ubuntu.com/HardySELinux>


```
init 1.9.0
libraries 2.0.0
locallogin 1.6.0
logging 1.9.0
lpd 1.8.0
modutils 1.6.0
mount 1.9.0
mta 1.9.0
selinuxutil 1.8.1
ssh 1.9.0
stunnel 1.5.0
sysnetwork 1.5.0
unconfined 2.1.0
userdomain 2.5.0
xserver 1.7.0
```

After installing `refpolicytargeted` (`apt-get install selinuxpolicydefault`), we will try to load the Apache policy :

```
# semodule i /usr/share/selinux/refpolicytargeted/apache.pp)for Debian
# semodule i /usr/share/selinux/default/apache.pp for Ubuntu
```

Relabel the files related to the new policy :

```
# restorecon R v /etc /usr/sbin /var/run /var/log
```

Restart Apache and check with `ps axZ` that apache is confined

Note that here SELinux is running in targeted mode, not strict mode and is not providing containment against an attack which leads to a root shell.

6.2.2 GRSecurity, AppArmor, TOMOYO, SMACK and co

Beside SELinux, there are other solutions which provide Access Control, each having its peculiar strengths and weaknesses.

We chose to focus on SELinux because of its presence by default in the Ubuntu server kernel and its large use. But there are other very good solutions such as GRSecurity, which however requires a recompilation of the kernel.

It must be noted that AppArmor is, at least for now, the default MAC on Ubuntu systems. While the inner workings of AppArmor and SELinux are quite different, from a user perspective they are not so different, with sets of policies defining what each user/process can actually do on the system.

- **GRSecurity** : A very complete solution which on top of providing a rather easy to configure RBAC system enhances the global security of the system. For more information see chapter 7. A

- AppArmor : An alternative solution to SELinux, which aims at being easier to configure and maintain. It is the default MAC present on Ubuntu server, and its policies can be found under its self-named directory inside /etc
- TOMOYO¹⁹ : A MAC implementation by NTT, it is pathname based (where SELinux is label based) and present by default on linux kernel (from 2.6.30²⁰). Note the presence of an interesting learning mode. This is, indeed, a promising project, but not yet ready for production.
- SMACK²¹ : A MAC system which aims at simplicity, integrated by default in the Linux kernel.
- RSBAC²² : An other AC control mechanism.

AppArmor is no longer really supported by Novell nor included by default in the kernel, which raises interrogations on his continuity. Mandriva switched from AppArmor to Tomoyo (in Mandriva Linux 2010).

Summing up, Red Hat and Fedora leverage SELinux, Suse Linux and Ubuntu use AppArmor and Mandriva Tomoyo.

6.3 Virtualization

As presented in part 6.2 Access Control mechanism has its advantages and drawbacks.

While virtualization is a broad technology with multiple impacts at many level of IT, it can be somehow used to restrict and control access and execution of software. In some situations, virtualization is a very effective and convenient way to achieve privilege separation and isolation.

In comparison to AC mechanism it has some advantages :

- It is easier to configure and maintain the confined parts. Out of the initial configuration, maintenance of the confined parts is identical to a classical system which a large majority of your sysadmins are able to do (in comparison to a system running SELinux or GRSecurity which requires experienced admins). Formation and team switching are much easier to perform.
- Most MAC relies on a well done configuration and on a trustable kernel. A good configuration is difficult to achieve and requires experienced admins. Some times the kernel can not be trusted because of a procedure error or in the case of some exploits (see part 6.2.1 on SELinux). Virtualization techniques which allows each VM to run its own kernel (so we exclude Operating System level Virtualization) permits to avoid such problems. Of absolute isolation is not easily achievable, some attacks may permit to

¹⁹ <http://tomoyo.sourceforge.jp/>

²⁰ http://kernelnewbies.org/Linux_2_6_30

²¹ <http://schaufler-ca.com/>

²² <http://www.rsbac.org/>

execute code on other VM or on the host, such as CloudBurst for VMware²³. But it adds one more layer of security : the attacker needs at first to control the guest and then to pierce the virtualization system.

- The classical advantages of virtualization : server consolidation, easier management, optimized hardware usage, etc. For more informations on the advantages of virtualization see this pointers²⁴²⁵²⁶.

There are mainly 2 kinds of technologies .

- Full virtualization, which emulates completely or partially hardware. It allows to run any kind of guest OS (Linux, Windows, BSD, Plan 9). It provides very good isolation but requires much more resources to run (VT instructions and multi core OS are advised). VMware, VirtualBox or QEMU are some examples of this kind of technology.
- Operating System Level Virtualization is a lightweight approach, similar to a fat BSD jail or a chroot on steroids : one kernel and n instances of the same OS. It provides the best performances but provides less isolation and require the same OS to run in the containers.
OpenVZ or Linux V-Server are some examples
Note that OpenVZ and V-Server are easy to install : a simple apt-get is sufficient
V-Server is compatible with GRSecurity

Paravirtualization is a particular kind of hardware virtualization, where the guest OS is modified to perform some of the tasks of the VMM, thus providing better performances but requiring a specially modified Guest OS. XEN, UML or KVM are some examples of Hypervisors (Virtualization Softwares) supporting paravirtualization natively.

Example

A typical example of isolation by virtualization, considering a LAMP/Asterix install would be :

- a VM acting as a firewall/QOS running an OpenBSD (or a PfSense)
- a VM for Apache/PHP
- a VM for MySQL
- a VM for asterix
- The 3 applicative VM are not directly connected to the network but placed on specific “host only” virtual networks. All the communications shall pass via the firewall VM which can filter the flux and manage the QOS.

²³<http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2009-124>

²⁴ <http://software.intel.com/en-us/articles/the-advantages-of-using-virtualization-technology-in-the-enterprise/>

²⁵ <http://www.networkworld.com/community/node/34082>

²⁶ <http://www.btquarterly.com/?mc=pros-cons-virtualization&page=virt-viewresearch>

CHAPTER 7 ADVANCED HARDENING

THE LAST 2 CHAPTERS OF THIS GUIDE ARE A LITTLE MORE THAN A COLLECTION OF NOTES. WHILE ARE NOT AIMING AT PROVIDING A COMPREHENSIVE STEP-BY-STEP GUIDE TO ADVANCED HARDENING, WE HOPE TO PROVIDE SOME USEFUL INSIGHTS TO THE READER NONE THE LESS.

7.1 SSH

In this section we will consider that the attacker may have a 0 day exploit for SSH, and see some workarounds to mitigate the impact of its attack. Beside mitigating a 0-day attack, these workarounds may apply to SSH access to appliances which are rarely updated or to other kind of authentication services.

7.1.1 Invisible

To avoid being attacked via the SSH vector, the simplest solution is to make it unreachable.

Obviously the first question is « do I really need an OpenSSH daemon running ? », if it is just by convenience or because it is running default, get it down.

The next question is « why do I need and OpenSSH running ? », is it possible to perform the same task without it. I remember a sysadmin who was getting his servers statistics (disk occupation, average load, some logs from a software) with a script getting them via OpenSSH. He could have got them by mail, without having another daemon running.

« From where do I need to access it ? » and « Which kind of use ? ». If you connect to the server only from a limited number of IP, you should allow connection only from those IP, which will reduce the attack IP space.

If you need an access from everywhere you will need to hide it, use a security airlock or a dedicated connection.

7.1.2 Dedicated connection

A dedicated connection over a RTC/GSM/Radio link is a robust but costly solution. It is also possible to filter the incoming communications. In general, this kind of solution is rather used as a backup line rather than for pure security purposes. The main advantage is that the network is not the internet but an operators network which is (hopefully to

some degree) better controlled and easier to restrict (ex bruteforcing, MITM).

7.1.3 Portknocking

Port knocking works like the secret knock on the closed door, identifying you as a friend. The basic idea is to close all the ports of a given service, so if an attacker performs a scan he will see a closed box. When you want to connect to your box, you first « knock » on port 3000 then 4000 and 6000. The daemon understands that you are a friend and opens the desired port only for your IP and for a determined amount of time. This solution can be implemented for example with portknockd.

However these solutions present some drawbacks : it is replayable, does not authenticate the user and does not work with NAT.

An other idea is single packet authorization : the client sends a unique encrypted packet which is passively monitored by the server. This solution is much more discrete (it won't make your IDS yield) and non replayable.

An implementation is fwknop²⁷.

7.1.4 Airlock

If the service needs to be exposed and the previous techniques can't be used, you can still set up an airlock : a dedicated or virtual machine acting as a proxy. It can be a thin, lightweight machine or a virtual machine, acting only for one machine or for an entire network.

The machine needs to be particularly well hardened and monitored. A good solution may be to use SSH to connect to it and, an other way to connect to the server (ex SSH to the proxy then connect to a VMware admin console over the tunnel which gives you access to the VM).

7.1.5 Anti bruteforcing

Softwares such as fail2ban or OSSEC can blacklist IPs from which are coming bruteforce attacks.

Those kind of software are effective against bruteforce attacks comming from a limited number of IPs but will not work against a distributed bruteforce attack.

Securing a system in such a way is always a tradeoff between securing it against one type of attack and opening new DOS vectors.

A nice how to for fail2ban may be found here²⁸

There is also the iptables solution²⁹ :

27 <http://cipherdyne.org/fwknop/>

28 http://www.howtoforge.com/fail2ban_debian_etch

29 http://kevin.vanzonneveld.net/techblog/article/block_brute_force_attacks_with_iptables/

```
sudo iptables -A INPUT -i eth0 -p tcp --dport 22 -m state --state NEW
-m recent --set --name SSH
sudo iptables -A INPUT -i eth0 -p tcp --dport 22 -m state --state NEW
-m recent --update --seconds 60 --hitcount 8 --rttl --name SSH -j DROP
```

Pam Tally can also be used for this purpose³⁰

7.1.6 Using keys

To avoid password bruteforcing an easy and efficient solution is to use Public Key authentication³¹. An how to can be found here³². Of course do remember to disable the password based authentication.

As this is in theory, the best solutions there are usually two problems :

- strong PRNG are rarely used, which are mandatory for high security environment. A lot of companies would not have so many problems after the Debian OpenSSL debacle if they had used an hardware PRNG.
- Key distribution is a huge problem. Usually users are not likely to send their passwords via mail (of course some of them do it) but it won't bother them to send a simple file. In other cases they will leave their private key unprotected and world readable. And to often they will leave both key and password authentication, in case they loose their key, with weak passwords.

I warmly advise you to use Public Key authentication, but only if you are sure to be able to correctly manage the keys.

7.2 Anti scanning

Softwares such as portsentry³³ can block certain kind of port scanning^{34,35}.

7.3 Kernel Hardening

Grsecurity/PaX works at 2 levels :

- an hardened kernel (a lot of patch which will close a lot of possible attack vectors)
- An ACL system, less powerful than SELinux but easier to use.

Grsecurity/PaX contains attacks at application level, attacks which leads to a root shell, and will prevent some attacks which leads to arbitrary code execution in kernel space. For example the vmsplice() exploit doesn't work on a box running a grsecurity/PaX

30 http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/sag-pam_tally.html

31 <http://www.openbsd.org/cgi-bin/man.cgi?query=ssh-keygen>

32 <http://sial.org/howto/openssh/publickey-auth/>

33 <http://sourceforge.net/projects/sentrytools/>

34 <http://linux.cudeso.be/linuxdoc/portsentry.php>

35 <http://aide.sivit.fr/index.php?2006/01/20/96-portsentry> (in french, but easy to understand)

kernel with the UDEREF option activated

From the website of the project :

«grsecurity is an innovative approach to security utilizing a multi-layered detection, prevention, and containment model. It is licensed under the GPL.

It offers among many other features:

- * An intelligent and robust Role-Based Access Control (RBAC) system that can generate least privilege policies for your entire system with no configuration
- * Change root (chroot) hardening
- * /tmp race prevention
- * Extensive auditing
- * Prevention of arbitrary code execution, regardless of the technique used (stack smashing, heap corruption, etc)
- * Prevention of arbitrary code execution in the kernel
- * Randomization of the stack, library, and heap bases
- * Kernel stack base randomization
- * Protection against exploitable null-pointer dereference bugs in the kernel
- * Reduction of the risk of sensitive information being leaked by arbitrary-read kernel bugs
- * A restriction that allows a user to only view his/her processes
- * Security alerts and audits that contain the IP address of the person causing the alert»³⁶

Be careful at some options, especially if you want to run it in a virtual machine.

Emulate Trampolines → NO

from the help in menuconfig : “enabling this feature *may* open up a loophole in the protection provided by non-executable pages that an attacker could abuse.”

Restrict mprotect

→ YES if you are running a physical server

→ NO if it is a virtual one

Prevent invalid userland pointer dereference

If you are running on a physical machine → YES

If it is a virtual Machine it will slow it down a lot → NO

Note : UDEREF makes sure that (data) segments for userland and the kernel are properly limited, either upwards (userland) or downwards (kernel)³⁷

UDEREF and KERNEXEC on would have prevented for example the vmsplice() local root exploit³⁸

³⁶ <http://www.grsecurity.net/>

³⁷ <http://grsecurity.net/~spender/uderef.txt>

³⁸ MISC 39, p8, “Linux/vmsplice : la faille 3 en 1”, Matthieu Loriol, in French

7.4 Delusion and obfuscation

Up to now we focused on securing and monitoring our server, but there's still something we can do to void attacks: delude the attacker.

From an attacker's perspective, the first step to attack a system is to study it : which ports are open, which services are running, what OS, which version, and so on.

With such a knowledge, an attacker will be able to perform aimed attacks which have the maximum of chances to succeed.

The general aim of this section is to explain how to complicate this information gathering step, hiding or tampering information.

The main advantages of such an approach are :

- it will stop a lot of dumb attackers/automated attacks (ex : bots) which rely on banner grabbing
- It may slow down an attacker, and buy you a little time to react
- It will likely make the attack noisier

7.4.1 Hiding Banner informations

A lot of softwares such as SSH, Apache or Postfix give a lot of information to an attacker. For example if I telnet a freshly installed Ubuntu server on port 22 I get :

```
"SSH-2.0-OpenSSH_4.7p1 Debian-8ubuntu1.2"
```

We know that the machine is running OpenSSH v4.7 and that it is an Ubuntu: free information for an attacker to get. Hiding at least part of the information (the version is needed for some RFC, but not the "ubuntu" for instance) to make life a little harder for the attackers.

If we take a random website running apache we may have something like :

```
"Server: Apache/2.0.55 (Debian) PHP/5.1.2-1+b1 mod_ssl/2.0.55 OpenSSL/0.9.8b"
```

For some applications (like Postfix) modifying the banner is an easy task, a matter of trivial reconfiguration; for other applications, like OpenSSH, you'll need to recompile the software from source after applying a proper patch.

Some examples :

Apache :

To avoid "Server: Apache/2.0.55 (Debian) PHP/5.1.2-1+b1 mod_ssl/2.0.55 OpenSSL/0.9.8b", in your httpd.conf add/(or uncomment)

```
ServerTokens ProductOnly
```

```
ServerSignature Off
```


(and `expose_php` = Off in your `php.ini` if you use php)

Regarding Apache, an elegant and clean alternative is leveraging `mod_security` with

Server masking is optional

`SecServerSignature` "Microsoft-IIS/0.0"

see here : <http://techgurulive.com/2008/08/15/secure-your-apache2-with-mod-security/>

Postfix :

in `main.cf` modify `smtpd_banner` = `$myhostname ESMTP $mail_name` to `smtpd_banner` = `$myhostname ESMTP`

As for SSH, its version can be hidden by patching the source code³⁹ itself: in `servconf.c` replace

```
snprintf(buf, sizeof buf, "SSH-%d.%d-%.100s\n", major, minor, SSH_VERSION);
```

with

```
snprintf(buf, sizeof buf, "SSH-%d.%d-HIDDEN\n", major, minor);
```

However, it should be noted that if the operating system version or distribution is given out by some other daemon, the burden of patching and keeping the package updated is easily not worth it.

7.4.2 Deluding Scans

Deluding scans is a difficult matter, meddling with low-level TCP details: having a firewall or being behind a NAT can change TCP behaviour completely.

Generally speaking the best solution is to rely on an applicative proxy in front of the real service, for example an OpenBSD based proxy (`pf` allows to set fingerprints⁴⁰).

Should you want to try decoy methods in a production environment, which is not really a best practice, these three tools are among the most well known software:

- PortSentry can be used to block some kind of scans, other tools try to hide or fake the OS type and other to slow down the scans.
- IPMorphis a tool which aims at this kind of delusion, in fact it works against Nmap, Xprobe2, Ring2, SinFP and p0f
- Chaos Tables (now in x-tables) aims at deluding nmap scans⁴¹.

39 See http://www.kramse.dk/projects/unix/opensshhideversion_en.html

40 See *set fingerprints* in <http://www.openbsd.org/cgi-bin/man.cgi?query=pf.conf&sektion=5&arch=i386&apr>

41 <http://jengel.medozas.de/documents/Chaostables.pdf>
<http://jengel.medozas.de/projects/chaostables/>
<http://xtables-addons.sourceforge.net/>
<http://nfws.inl.fr/en/?p=104>

CHAPTER 8 : RANDOM LITTLE THINGS

8.1 Bash history

An attacker can learn a lot of informations from the `.bash_history` file : addresses of other servers, installed software or even passwords.

You can add to your `.bashrc` and `.bash_profile` or `/etc/profile` an order to delete history at login :

```
rm ~/.bash_history  
or  
shred ~/.bash_history
```

The paranoid way consists in directly linking your `bash_history` on `/dev/null`

```
ln -s /dev/null ~/.bash_history
```

Remember that this may impact usability of your system as you will not be able to access the history of the commands you entered.

8.2 Blowfish /etc/shadow

By default Ubuntu and Debian uses a md5 based function to store hashes of passwords in `/etc/shadow`. If an attacker is able to read `/etc/shadow`, he will try to force the passwords.

The time it will require depends mostly on two parameters : the quality of the password and the way it is stored. Both of them contribute to the quantity of calculus to perform, and thus the time an attacker will need to crack the passwords.

The OpenBSD team implemented⁴² a password hashing method based on blowfish, which due to the way blowfish works, requires more calculus and so more time to crack the passwords.

You can use blowfish based password storage by installing `libpam-unix2`. An how to may be found here⁴³.

42 Cf the USENIX paper <http://www.openbsd.org/papers/bcrypt-paper.ps>

43 <http://digitalconsumption.com/forum/615-Blowfish-shadow-files-on-Debian>

8.3 Absolute Path

It is advised to use important commands such as su or sudo by calling them with their absolute path : ex /bin/su.

If you don't know an absolute path use “which COMMAND” to get it.

This is a defence against a very trivial attack : if an attacker modifies your PATH variable to include “.” and puts a custom made su or sudo command he can steal your password or execute arbitrary code.

Check that \$PATH does not contain “.” or an empty string. Check it by echo \$PATH

8.4 Web Apps

Web applications are too often one of the weakest part of a system. If you can avoid their use, do it. If not you can hide them by restricting their access to some IP or only over a SSH tunnel or with portknocking, or even a trivial HTTP Basic Authentication. There's a huge difference between exposing a web application to the internet and protecting it even with the easisest of the passwords.

8.5 About Bubbles

Bubbles are more a management issue than a technical one : it consists of creating « security bubbles » for a few very important computers. Those few are placed in a separated dedicated network which is accessible only over a VPN and maintained by an « alpha team » (trained admins which only maintains a very small number of computers) and regularly audited.

8.6 About dedicated management networks

As a general rule, a separated, isolated network is a very good idea. Most implementations, however, are just flat network connecting every server on the network, regardless of firewalls and IDS. That's a very bad idea from a security perspective!

8.7 Secure deletion

If your server contains sensitive information, we need to prevent an attacker from recovering them if they are deleted. This is especially the case if the server is moved from one place to an other, if it's harddrives are sent to an other location or if it is stolen.

For this we can use shred which will overwrite the file a lot of time before canceling it, making very difficult its recovery.

To securely delete a file named toto2 :

```
# shred --remove toto2
```

Scenario : you need to move a file server from your italian office to your french data center. The server contains corporate critical informations (client informations, projects details, strategic plans). Even if you backuped your data and deleted it on the hard drive, if the transporting services is lost it or if it is stolen, an attacker can use forensics tools to get the deleted data. As filesystems encryption is not always an option on servers, you need to securely delete the information, for example using shred.

8.8 Gcc SSP

GCC SSP should be by default from edgy⁴⁴

8.9 More links on IDS

Tiger : <http://www.nongnu.org/tiger/>

Diffmon : <http://linux.about.com/cs/linux101/g/diffmon.htm>

swatch : for log watching

Scan detection : portsentry, scanlogd

OSIRIS : <http://osiris.shmoo.com/>

Module Hunter :

http://exitthematrix.dod.net/matrixmirror/misc/kernel_auditor/module_hunter.c

8.10 Deception Networks

Some companies uses large deception networks, a subnet of honeypots to slow down (ex with labrea) and detect automated attacks

8.11 Bastille

Bastille Unix⁴⁵ is an interactive script which helps at securing an unix system.

How to here⁴⁶

8.12 PSAD

“**psad** is a collection of three lightweight system daemons (two main daemons and one helper daemon) that run on Linux machines and analyze [iptables](#) log messages to detect port scans and other suspicious traffic. A typical deployment is to run **psad** on the iptables firewall where it has the fastest access to log data.”⁴⁷

⁴⁴ <https://wiki.ubuntu.com/GccSsp>

⁴⁵ <http://www.bastille-unix.org/>

⁴⁶ <https://help.ubuntu.com/community/BastilleLinux>

⁴⁷ <http://cipherdyne.org/psad/>

APPENDIX C : SOME SECURITY LINKS

A non exhaustive list of links around security which may be of some interest for the reader.

Advisories :

CERTA ANNOUNCES (In French) : <http://www.certa.ssi.gouv.fr/>

Secunia Advisories : <http://secunia.com/>

Packet Storm Security Headlines : <http://packetstormsecurity.org/>
<http://milw0rm.com/>

The Open Source Vulnerability Database <http://osvdb.org/>

Computer security experts Blogs :

(not an exhaustive list)

Robert Graham's Blog : <http://erratasec.blogspot.com/>

Stefano Zanero's Blog : <http://raistlin.soup.io/>

Bruce Schneier Blog : <http://www.schneier.com/blog/>

Richard Bejtlich's Blog : <http://taosecurity.blogspot.com/>

Claudio Criscione's Blog : <http://oversighting.com/>

Anton Chuvakin's Blog : <http://chuvakin.blogspot.com/>

Security Research, Computer Laboratory, University of Cambridge
<http://www.lightbluetouchpaper.org/>

Doxpara : <http://www.doxpara.com/>

Other source of information :

The register : <http://www.theregister.co.uk/security/>

GNUCitizen : <http://www.gnucitizen.org/>

Security Focus : <http://www.securityfocus.com/>

Heise security : <http://www.h-online.com/security/news/>

REFERENCES

References

[Cromwell] Bob Cromwell <http://www.cromwell-intl.com>
[W risk mana] http://en.wikipedia.org/wiki/Risk_management
[Mehari] <https://www.clusif.asso.fr/en/production/mehari/>
[Virt] http://en.wikipedia.org/wiki/Platform_virtualization
[VMware] www.vmware.com
[Ubuntu Server Install] <https://help.ubuntu.com/8.04/installation-guide/>
[Debian Install] <http://www.debian.org/releases/stable/installmanual>
[DefPAssList] Default password list <http://www.phenoelit-us.org/dpl/dpl.html>
[Ross Anderson SHB 2008] <http://www.cl.cam.ac.uk/TechReports/UCAM-CL-TR-500.pdf>
[NF wi] <http://en.wikipedia.org/wiki/Netfilter>
[Shorewall bastion] <http://www.shorewall.net/standalone.htm>
[Shorewall start] <http://www.shorewall.net/GettingStarted.html>
[Netfilter How to] <http://www.netfilter.org/documentation/HOWTO//networking-concepts-HOWTO.html>

For more convenience when reading the pdf version, all other references are made as footnotes.

Books in French

Manager la sécurité du SI - *Matthieu Bennasar, Alain Champenois, Patrick Arnould, Thierry Rivat* – DUNOD 2007

Comprendre et gérer les risques - *Franck Moreau & al.* - Editions d'Organisation - 2002

Books in English

Building secure servers with Linux – *Michael D. Bauer* – O'Reilly – 2003

Hardening Linux – *James Turnbull* – Apress – 2005

Hack Proofing Linux- *James Stranger, Patrick Lane, Edgard Danielyan* – Syngress – 2001

Linux security cookbook – *Barett & Al.* – O'Reilly – 2003

Practical Unix & Internet Security – *Garfinkel & Al.* – O'Reilly – 2003

Linux Bible – *Christopher Negus* – Wiley – 2007

Linux Firewalls: Attack Detection and Response with iptables, psad and fwsnort – *Michael Rash* – No Starch Press – 2007

Other Hardening guides

UNIX and Linux Security Checklist, Australian Computer Emergency Response Team (AusCERT) <http://www.auscert.org.au/5816>

Cromwell's Hardening Ressources, <http://www.cromwell-intl.com/security/linux-hardening.html>

Securing and Optimizing Linux, <http://www.faqs.org/docs/securing/index.html>

Gentoo Linux Security, <http://www.gentoo.org/security/en/>

NSA Hardening Tips Pamphlet for the Red Hat Enterprise Linux 5, <http://tuxtraining.com/wp-content/uploads/2008/04/rhel5-pamphlet-i731.pdf>

NSA Guide to the Secure Configuration of Red Hat Enterprise Linux 5, <http://tuxtraining.com/wp-content/uploads/2008/04/rhel5-guide-i731.pdf>