FIRST STEPS TO BUILDING A SECURE LINUX SERVER

Emil BURTESCU
University of Pitesti, Romania
emil.burtescu@upit.ro

Keywords: account, distribution, kernel, Linux, management, security, password.

Abstract: *Installing a Linux distribution is currently an important alternative to take into consideration not only for the server systems but for the desktop systems. Implementing Linux based servers for organizations must take into consideration the security options. Implementing security must be done by translating the security policy of the company in the options of configuring the Linux distribution. The subject proposed – Secured Linux server – (in 2 parts) intends to be an absolutely necessary guide for the companies which want to implement an informatics system based on a server which is efficient from the point of view of technique, stability, security and performance at a lower price.*

1. INTRODUCTION

Opening up your system as a server on a public network creates a whole new set of challenges when it comes to security. Linux is the best choice for an operating system when it comes to servers. Implementing a secured Linux server involves completing the following six stages: *Choosing a Linux distribution, Building a secure kernel, User account security, File and directory permissions, Syslog security and Filesystem encryption.*

In the following we will refer only to the first three stages, following that in Part II to treat the other steps also.

2. CHOOSING A LINUX DISTRIBUTION

In the following we will refer only to the Linux enterprise distributions.

The first stage, and the most important one in creating a secured architecture with Linux servers, is represented by choosing a distribution which meets the demands. Very many specialists consider this stage as being crucial.

The first question that arises is: 32 or 64 bit? Many will consider that the use of a 32-bit distribution offers the advantage of maturity. The ones who consider this are mostly users of Linux desktop. The real performance benefits of 64-bit computing are not found in day-to-day applications. Addressing an enormous quantity of memory and working with immense databases represents the raw material of a server. But why shouldn't I exploit the advantages used by the 64-bit processor which the server is equipped with? So the choice is clear: I will choose a 64-bit Linux distribution. So, we will leave behind the i386 distribution (sometimes also calls x86_32) and we will choose x86_64.

Choosing a Linux distribution is often a difficult operation. This situation is generated by several factors, including:

- Server specification;
- The existence on the market of more Linux distributions;
- The volume and quality of the necessary documentation for installation and administration;
- Vendor support (update and consultancy);
- Distribution alignment to the security needs of the corporation.

Choosing a secured Linux distribution is made according to the following important considerations [2][5]:

- Is there any mechanism for reporting security vulnerabilities discovered in the vendor's distribution?

- Is the user alerted about the vulnerabilities discovered in his distributions by periodically security alerts sent by the vendor?

- What is the occurrence of the problems in that certain distribution? What is the rate at which these problems are solved?

- How often are general updates for the distributions provided by the vendor?

- Does the site of the specific vendor provide a specific section concerning treating vulnerabilities?

- Is there an intuitive, easy-to-use tool for installing and updating software packages provided by the vendor?

- Are open source efforts to improve the level of Linux security or to create vendor-neutral security scripts and tools supported by the vendor?

- Since when does the vendor exist on the market?

- Have the previous versions received proper security and maintenance?

- What is the market share of Linux distribution?

The most popular Linux Enterprise distributions on the market are: Red Hat Enterprise Linux (RHEL), Suse Linux Enterprise Server (SLES), Centos Enterprise Linux, Oracle Enterprise Linux (OEL) and Ubuntu Server Edition. The prices of these distributions vary from several hundred dollars for lite versions and go up to several thousands of dollars for complex versions, according to the number of years for assured support. There is also the option of annual subscription.

Given the considerations listed above, a first choice would be RHEL. Going on a Red Hat we can choose CentOS (the Red Hat Enterprise Linux clone) which has the advantage that is free and they don't really provide technical support staff that you can call for assistance while the Red Hat gives you 90-days of support with a new purchase (with more support available via contracts for more dollars). I have personally chosen this distribution due to the zero costs.

3. BUILDING A SECURE KERNEL

The second stage is represented by building a secure kernel. A great mistake is that of installing a Linux server with standard installation options. It would be perfect for us to build a custom kernel. For this we have to download the source, modify its configuration and build a custom kernel.

Building a custom kernel (compiling kernel) offers some advantages such as:

- Allows the configuration of the security specifications according to the demands.

- Allows the installation only of the specified drivers.

- The server administrator will know what he compiled and what the security holes that were left uncovered are.

Depending on the level of the security hole left uncovered it will be proceeded to an immediate or subsequent cover. The solution of many undercover security holes come in the form of a patch to be applied to the Linux kernel, that implies a new recompilation of the kernel.

Step 1. Obtain a kernel source.

To install a new kernel, we need to download the software packages for that kernel to our system or we can use packages installed on system. In Red Hat/CentOS we need either the kernel-headers and kernel-source RPM packages.

Step 2. Locate the kernel-headers and kernel-source. For that we need to be logged as root. From any Terminal window or shell, we can simply type:

[…]$ su

With the result

Password: ******

[root]#

and then use the following rpm query commands to display the kernel version:

[root]# rpm –q kernel-headers

Kernel-header-2.6.18-194.3.1.e15

[root]# rpm –q kernel-source

Kernel-source-2.6.18-194.3.1.e15

To compile the kernel from the beginning with our options we have to locate where the kernel-header and kernel-source are. In most of the cases these two packages are found in usr/src/linux directory.

On the Linux directory, cleans out the configuration files and any object files an older version might have:

[root]# cd /usr/local/linux (or in directory where is located e.g. cd/usr/src/linux)

[root]# make mrproper (erase any .config file)

Step 3. Specify the option that we want to build into the kernel.

We use the command:

[root]# make config

This command will display a series of options that will be added or not to the new kernel by selecting the options Y (yes) N (no).

According to [5], the main options for the kernel are displayed as it follows (Configure option, Description and Recommendation):

CONFIG_PACKET. The Packet protocol is used by applications which communicate directly with network devices without an intermediate network protocol implemented in the kernel, e.g. tcpdump. Recommendation: Y.

CONFIG_NETLINK. Netlink is used to transfer information between kernel and userspace processes. It consists of a standard sockets-based interface for userspace processes and an internal kernel API for kernel modules. Recommendation: Y.

CONFIG_FIREWALL. Adds the kernel support necessary to make this system a firewall. Enable this only if the Linux system will be your firewall. The full firewall installation requires additional software outside the kernel. Recommendation: Y.

CONFIG_INET. These are the protocols used on the Internet and on most local Ethernets. It is highly recommended to set Y, since some programs (e.g. the X windowsystem) use TCP/IP even if your machine is not connected to any other computer. Recommendation: Y.

CONFIG_IP_FIREWALL. IP filter is a software package that can be used to provide network address translation (NAT) or firewall services. Recommendation: Y.

CONFIG_SYSN_COOKIES. Normal TCP /IP networking is open to an attack known as "SYN flooding". If option is set to Y the system is protect against sysnflooding denial of service attack. Recommendation: Y.

CONFIG_NET_IPIP. IP inside IP encapsulation. Can be useful if you want to make your or other machine appear on a different network than it physically is, or to use mobile-IP facilities (allowing laptops to seamlessly move between networks without changing their IP addresses). Recommendation: Y.

CONFIG_IP_ROUTER. Some Linux network drivers use a technique called copy and checksum to optimize host performance. For a machine which is forwarding most packets to another host this is however a loss. Recommendation: Y.

CONFIG_IP_FORWARD. Determines whether or not the system forwards IP datagrams. This feature must be enabled if the Linux system is an IP router. On Linux host systems, this feature is disabled, which is the default. Select "no" unless this box is a router. Recommendation: Y.

CONFIG_IP_MULTICAST. This is code for addressing several networked computers at once. We need multicasting if we intend to participate in the MBONE, a high bandwidth network on top of the Internet which carries audio and video broadcasts. Recommendation: Y.

Step 4. Build a list of dependencies.

This step is a must. This insures that all of the dependencies, such as the included files are in the right place. This removes all of the object files and some other things that an old version leaves behind.

[root]# make dep

[root]# make clean

It will save a lot of disk space and can help build a clean distribution.

Step 5. Compile a new kernel.

Don't confuse a zImage kernel with a disk image. A disk image contains a kernel and a filesystem, the zImage kernel is just the kernel itself in a compressed format. Additionaly we have a bzImage. Basically, a bzImage kernel is loaded into high memory and zImage (a regular kernel) is loaded into low memory.

If the loadable module support was chosen during step 3 (make config), then the following instructions must be executed:

[root]# make modules

[root]# make modules_install

These commands will build the various components which we chose to build as modules and will create /lib/modules/'uname -r' and copy the modules there.

Step 6. Saving the old Linux image.

Make a backup copy of an old Linux image:

[root]# cp /zImage /zImage.OLD

Save zImage:

[root]# mv /usr/scr/linux/arch/ixxx/boot/zImage /zImage

Update the bootloader:

[root]# /boot/grub

That command causes the bootloader, grub, to check its config file (/boot/grub/grub.conf) and update the stuff that sits on the master boot record (mbr) of your hard disk, so that the next time you boot up, it (grub) will show the latest configuration.

Step 7. Restart the computer.

After restarting the computer we will have a kernel built from our own security demands. It remains to be seen if it works as we want.

Before pre-install a new kernel step is necessary to make an emergency boot floppy. Linux has a small utility named mkbootdisk to simply do this. Finally, after restarting the computer and making sure the new built kernel works well, it's time to make a new rescue image with the new kernel in case of future emergencies.

4. USER ACCOUNT SECURITY

The third stage is represented by the delicate problem of account security. The user's login using account and password represents the most used method of validating the access to the resources of a system, and because of this a special attention must be given to this aspect when security measures are implemented. To achieve the desired security level it is necessary to be taken into account the following important indications [5]:

Disabling the inactive accounts

Attacking inactive accounts offers the attacker the advantages given by the fact that these accounts are present for a longer period in the system and due to the fact that they are inactive it is possible for them not to be monitored by the intrusion detection mechanisms. Taking these into account it is necessary that these accounts to be deleted immediately as they expired and are not used anymore. This operation is done automatically by the operating system by specifying the conditions by which an account can be disabled.

Disable of Root Access NFS Mounts

If the option of sharing the Network File System (NFS) is used, the Linux distribution maps the user IDs 0- root to a non-privileged user. Thus, the access to the NFS server can be done by any root from the client work stations.

Using the root account must be done only when it is absolutely necessary

In quite many cases, when debugging or checking something on the client workstations the administrator has the tendency to login as a root even if it is not necessary. This situation increases the possibility for the root password to be discovered.

Using a specific prompter when a one works as a root

Using a specific prompter for the root and also some colors specific to this situation will help avoiding the execution of a user command from a root window.

Use a minimal $PATH for your root account

Using a minimal set of variables and of the content of variables that are found in $PATH will reduce the attacks that use Trojan horses to find out the root password. In the $PATH only variables and folders written by the root should be found.

Using specific accounts

The accounts will be created and will be given privileges according to the level of security to which the account is assigned taking into consideration the security policy of the company also.

Creating groups of users

This option allows the grouping of users that have a certain task in the users group and creating rights for that certain group. This will reflect later on the users of the group. In this way managing the rights will be done easier - I give rights to the groups and in the groups I insert users.

Restrict root login to the system console

The etc/securetty defines which terminals root can log in only from console.

This option forces the authorized root users to use su or sudo to obtain root privileges, and so allowing the counting of logins.

Edit the securetty file

[root]# more /etc/securetty

and comment out the following lines:
vc/1, vc/2, … vc/11 and corresponding
tty1, tty2, … tty11.

Each line represents a connection that root was allowed to connect to. If we want to restrict root to logging in from the system console then:

[root]# vi /etc/securetty
and disallowed with an # in front of (e.g. #vc/1)

We should also consider adding TMOUT to /etc/profile. This environment variable will log people to the value set in seconds.

The restricting root from directly accessing the server if only from the console is intended to keep the most of user from authentication directly (without a non-privileged user logging in first). This limits your attack vectors for brute forcing the root password directly. It is a small step, but an important one for the security of your server.

5. PASSWORD MANAGEMENT

For an efficient password management it is necessary for us to take into account the following basic rules:

Choose correctly the time after which a password will expire

Even if it seems a simple operation at first sight, if you will ask different system administrators who do this operation the answers will be different. If the time after which a password expires is too short, then the user will be in the situation to remember as many passwords in a certain period of time. If on the contrary, the time after the password expires is too long, then this situation will offer the attacker a longer period of time for trying to break it. In specialty literature it is considered that a password must not expire sooner than 3 months and not later than 4 months.

Choose correctly the length and the content of the password

In order to accomplish its purpose, a password must be made taking into consideration 2 elements: length and content. Here is where the problem gets delicate. A „short" password can be easily guessed. In specialty literature it is considered that a password under 6 characters is a „short" password. A password that is long and contains only characters or digits is very easily broken. It is preferred that the password contains a combination of letters, digits and other symbols and so a length of six characters will be easily remembered and hard to break. Ideally your password should contain at least one character from each of the following categories: upper case letters (ABC), lower case letters (abc), digits (123), punctuation and other „complicated" symbols (!$%).

Auditing of system passwords

Periodically it is necessary that the „solidity" of passwords to be checked, this being a part of the auditing of security system. For this we use password-guessing tools to find a weak password in your /etc/password file.

Another direction on which one has to choose is that in which he doesn't allow through system configuration that the user to have his own password.

Interdiction of empty password fields

The existence of empty password accounts must be denied. This is sometimes practiced because that account has a low security level and it is considered that important data are not accessed. This is completely wrong because the existence of that account represents a security breach and can be exploited by an experimented intruder.

Deleting the accounts of services that are not used

All accounts of services that work on a Linux Server but they are delivered with that certain distribution must be deleted. It is the case of the services that are not found on a Linux Server such as: Point-to-Point Protocol (PPP), Unix-to-Unix Copy (UUC), Network News Transfer Protocol (NNTP), Gopher and other.

The easiest way to break a password by an experimented user is represented by an employee's use of specific tools. The employee has sometimes the advantage that he has knowledge about the length and the way the password was formed, similar to his own password. An efficient way to manage passwords and to reduce to risk is represented by the support offered by tools such as shadow password and shadow group files

If the shadow password is activated then the password files will be split in two: /etc/password file, which contains the placeholder in the actual password field entries, and /etc/shadow file, which contains the encrypted password. The etc/shadow file can be read only by the root, this way being reduced the possibility for the password to be discovered.

If the shadow groups' option is activated then the group members are not listed anymore in /etc/group file but in etc/gshadow file which is accessible only for the root.

A very efficient utility program used for minimizing root password exposure is represented by sudo.

Password management solutions is another option to reduce the likelihood that passwords will be compromised because they are written down, typed in at a keyboard (and the typing monitored by people or malware), or created weak to make them easier to remember [6].

These are the first „raw" stages that are absolutely necessary to have a secured Linux Server. In the second part we will talk about the stages referring to File and directory permissions, Syslog security and Filesystem encryption.

6. CONCLUSIONS

Besides the well-known advantage of functioning stability, building a Linux server has low installation and maintenance costs. Yet, the installation is sometimes difficult due to a certain amount of factors. The domain experience of the one who does the installation and the related documentation represent an absolutely necessary starting point. Based on these two elements the Linux server will have its foundation assured: a stable and performance Linux distribution and also a secure kernel. The most delicate part is represented by the security and especially the account security. It is delicate because it has to comply with the security policy of the company, which has to be defined and formalized. In many cases, in the small and medium-sized companies this is the place where first problems that need to be solved quickly appear. The optimal management of the access rights, an optimal password management, commissioning and decommissioning of the users that come or leave the company will represent the difference between a company which develops its activities according to the established objectives or will waste its resources for covering the security flaws occurred because of the company. To ensure its security, depending on its size, the company must have well-trained personnel which should maintain at a proposed level the security services or maybe a certain amount of these services to be externalized. The security activities are focused on servers due to the fact that here most of the data of the company is stored.

7. REFERENCES

[1]. T. Boronczyk, C. Negus, „*CentOS Bible*", Wiley Publishing, Inc., 2009.
[2]. E. Burtescu, „*Servere de baze de date*", Ed. SITECH, Craiova, 2012.
[3]. R. I. Petersen, „*Red Hat Enterprise Linux & Fedora Core 4: The Complete Reference*", Mc Graw Hill/Osborne, 2005.
[4]. M. Turner, „*Administrarea Red Hat Linux. Cunoştinţe esenţiale*", Ed. ALL, 2004.
[5]. ***, „*Security Complete, Second Edition*", SYBEX Inc., 2002.
[6]. http://csrc.nist.gov/publications/drafts/800-118/ draft-sp800-118.pdf
[7]. http://kernel.xc.net/html/linux-2.6.19/ x86_64/
[8]. http://www.faqs.org/docs/securing/secopt-kernel.html
[9]. http://www.linux.com/archive/feed/56984
[10]. http://www.tuxradar.com/content/how-choose-best-linux-distro
[11]. http://linuxfromscratch.org/~bdubbs/secure-linux.pdf