Securing Linux Servers

Best Practice Document

Produced by the ASNET-AM Server Security Working Group
Authors: A. Petrosyan (ASNET-AM), ...
October 2015

Table of Contents

Executive Summary

This document describes the best practice recommendations for system administrators regarding installation and initial configuration of Linux servers. The goal of the document is to assist administrators in preparation of production-ready servers to provide Linux server-based network services. Most of the solutions described are time-tested must-have widely used basics practices and solutions, freely available within the Linux distributions.

Introduction

Linux is popular choice for an operating system in a server environment. The granularity and flexibility of settings, high performance, reliability and security are some of its comparative advantages over other operating systems. The vast majority of services that academic institutions provide to their users are hosted on servers running the Linux operating system. Due to infrastructure limitations, one server often hosts several services, which adds to the challenge of protecting the Linux server. System administrators are expected to protect the server from potentially malicious activities that could jeopardise or compromise the provision of services. However, the protection of a Linux server is not a one-time effort, but a lasting process that continues as long as the server is in use. This document contains guidelines for an administrator to follow when setting up protection for the Linux server, with the aim of increasing security and detecting problems quickly.

1 Installing the Linux Server

The Linux operating system is freely available in a form of different collections, known as distributions. Large distributions have grown historically and composed distribution families. Most widely used distribution families currently are:

- RedHat family (Fedora/RHEL/CentOS/...)
- Debian family (Debian/Ubuntu/Mint/...)

There are many other Linux distributions (like Slackware, Gentoo, OpenSUSE, Arch Linux, etc) that may be used as well. System administrators may choose any distribution, based on their preference, experience, available documentation or suitability for particular network services.

Linux distributions generally offer the following types of installation:

- Standard or default server installation.
- Specific or custom server installation.
- Minimal server installation.

For any of the above types of installation, a specific set of steps to be implemented is required, in order to protect the server from potential misuse. The basic idea behind the protection of a Linux server is to have the system administrator control the work of the entire server and only enable the packages that are necessary for the planned services.

The **Standard** server installation gets you a Linux operating system with a large number of the most frequently used packages (applications). This is the most suitable type of installation if the server is required to host several services. However, installing a large number of packages also implies a number of applications that are not used, but are open to misuse in compromising the security of the server. If the administrator opts for this type of installation, the packages that are not required for the operation of the server should be manually disabled or removed after installation.

The **Specific** server installation enables the administrator to install the Linux operating system containing the packages necessary to use the server for a narrowly defined set of implementations, e.g. a web server, database server, etc. Such an installation is suitable for servers hosting one basic service. After the specific server installation, the administrator should check the list of packages installed and manually remove those that will not be used.

The **Minimal** installation of a Linux distribution includes the installation of the basic packages necessary for the work of the operating system. Administrators should have a clear picture of the services they plan to host on the server and install the necessary packages separately after the minimal installation of the operating system. This type of installation is best for maintaining security, but it requires an advanced knowledge of the Linux operating system and the services that are intended to run on the server.

1.1 Disabling/Removing Unnecessary Network Services

Before adding any new network server, its role within the network and a list of all the services expected for it to provide, should be clearly defined. All other services are to be disabled or removed.

Besides disabling unneeded services, it is also possible to take the more drastic approach of removing the packages that control such services.

Removal of services is safer than just disabling them, because it excludes the possibility of re-enabling the service by chance, either manually or after next reboot. Removal is recommended when there is absolutely no need for certain services, or if there is an alternative option that ensures a higher level of protection. System administrators should deeply understand the functionality of all active processes on each Linux server they administer in order to be able to predict possible security-related or any other impact, that the server could have to a services provided or to the whole network stability. This requires understanding of how any active Linux process operates, what default configuration parameters does it use for the service provided.

It is also important to compare the advantages offered by each individual service against the problems they might cause as each additional service may hatch a new security problem. Such an approach requires the administrator to invest more time into installing and configuring the Linux server, but it dramatically cuts the maintenance time in the long run as the potential security issues have been dealt with proactively.

There is no universal list of unneeded services for disabling. Linux distributions differ in the way which of the services they install by default. In each case the administrator should investigate the server after installation to find out and disable/remove unneeded start up processes to keep the system as clean and tuned as possible.

The following services mostly have to be disabled or removed for server environments:

Network Service	*Reason for disabling/removal*
lpd, cups	Unless printing is required.
fingerd	Insecure.
inetd*, *xinetd	Outdated and insecure. Consider switching to more secure services which provide the needed functionality.
tftpd	Insecure.
avahid	Unless automatic detection of other devices in the network is required.
bluetooth	Unless Bluetooth is required.
nfsd, mountd, lockd, statd, nfslock, portmap	Unless NFS is required.
rsh	Insecure.
telnetd	Insecure.

anacron	Same as cron, but does not assume machine to be always up and running. Mostly not needed for server, where cron can we used instead.
autofs	Unless automount is required. Mainly is used for X and desktop apps.
kdump	Unless kernel crash dump analyzing is required.
haldaemon	Unless collecting and maintaing information about hardware from several sources is required. Mainly is used for X and desktop apps.
gpm	Unless mouse cut/paste function in virtual text consoles is required.

Any unneeded network service package should be disabled or removed, because by listening some network port (tcp/udp) and accepting remote connections from clients, it can become a potential security flaw.

List of current active network services can be obtained by executing commands like: *netstat -nlptu*, *ss -nlptu*, etc. It's also important to look at which network address does particular service listen (services listening only to 127.0.0.1 do not pose any threat from outside). In addition some service in modern linux configurations could be initially armored by some firewall rules, which should be investigated too to have a complete picture of open ports.

1.2 Kickstart/Preseed Installation for Specific Needs

A majority of Linux distributions come with tools that enable administrators to automate the installation process. The two most common mechanisms are Preseed for Debian family distributions and Kickstart for RedHat family distributions. A Kickstart/Preseed installation allows the administrator to create a configuration file that will contain the answers to all the questions asked during the installation process (including security settings, for example). This enables the administrator to create uniform Linux installations for specific requirements within the network. The configuration file is generated manually (from a text editor) or by using GUI tools such as the Kickstart Configurator. An example of the Kickstart configuration file corresponding to a given installation for RedHat family distributions can be found in the */root/* directory under the name *anaconda-ks.cfg. system-config-kickstart* graphical application can be used to create and modify Kickstart files. To automate the installation process with a Kickstart file, the ks option can be used to specify the name and location of the file during installation boot process (more information can be found in the corresponding distribution manual).

2 Securing Remote Administration

Nowadays Linux servers are mainly administered remotely. Thus a secure remote connection means is to be defined for Linux server administration. Such link-level security can be obtained by network connection encryption wherever possible.

2.1 Securing Remote Terminal Access (SSH)

System administrators most often use the Telnet and SSH protocols for remote access to the Linux server. Telnet has largely become obsolete since it does not encrypt the data that is exchanged with the server, including user credentials. Anyone on the network between the administrator's computer and the server it communicates with can intercept the packets and come into possession of the user credentials, and thereby establish full control over the server. For these reasons, it is generally recommended to avoid using Telnet and to use the SSH protocol instead.

The SSH protocol offers protection in communication with the server by using asymmetric-key cryptography. The entire communication between the SSH client and the SSH server is encrypted and incomprehensible to third parties that might intercept the packets being exchanged. The SSH client authentication can be achieved by the encrypted transfer of user credentials, or by manually generated asymmetric keys, in which case it is not necessary to use a password. If the authentication is conducted using manually generated asymmetric keys without password transfer, the authentication key must be stored on the device that will be accessing the Linux server. The use of user credentials is recommended for administrator login to the server in case different computers are used for remote login to the Linux server. For communication and mutual authentication between two Linux servers, it is recommended to use manually generated asymmetric keys because the parties to the communication are permanent and the need to have the password written in the files on the server is thus avoided.

2.2 Securing File Transfer (SFTP/SCP)

If data transfer between the Linux server and another remote machine is required, this too should be done through secure communication channels. The FTP protocol is the most popular file transfer protocol, but it involves the same security risks as the Telnet protocol (user credentials are transferred in a readable format, there is a possibility of "man-in-the-middle" attacks, etc.). Therefore, the following more secure alternatives to the FTP protocols are recommended: FTPS, SFTP or SCP data transfer protocols.

FTPS (File Transfer Protocol Secure) is an extension of the FTP protocol based on the TLS (Transport Layer Security) and SSL (Secure Sockets Layer) security protocols that enable secure data transport. FTPS is largely similar to the HTTPS protocol (Hyper Text Transfer Protocol Secure), and it also requires the use of a digital certificate on the server. Since the other party to the communication must trust the installed digital certificate on the server, this may cause a problem in the implementation of the solution as a whole. If the file transfer capability is used for the needs of the system administrator and for server administration purposes (e.g. the transfer of log files to a remote log server), it is recommended to use the SFTP protocol as it offers more file management options. If files are transferred from the Linux server as part of a service provided to end users, then the FTPS protocol is recommended because it enables server authentication through digital certificates. FTPS uses the TCP/UDP ports 989 and 990 for establishing a connection and transferring files, so it is necessary to have these ports open on the Linux server.

SFTP (Secure File Transfer Protocol or SSH File Transfer Protocol) should not be confused with the term Simple File Transfer Protocol. SFTP was designed as an extension of the SSH protocol and is based on it. The SFTP protocol establishes secure encrypted communication with a remote device in order to enable access to, management and transfer of files. The protocol uses an SSH tunnel to manipulate files on the remote server.

SCP (Secure Copy Protocol) is a protocol that uses the SSH protocol to ensure the authentication and encryption of the transferred data. The mechanism is very similar to the one employed by the SFTP protocol, but the latter includes considerably greater capabilities than the mere transfer of files. SCP and SFTP use the connection on the TCP port 22 for secure file transfer.

Thus by using SSH/SFTP/SCP only administrator can get all-in-one solution just via one TCP port 22.

2.3 Securing Web Access (HTTPS)

A web server is one of the most common implementations on Linux servers. Nowadays, web pages increasingly require user authentication for the exchange of sensitive data, which renders the HTTP protocol unsuitable because it does not perform the encryption of credentials exchanged in the process. If it is necessary to ensure authentication and exchange of sensitive data, administrators are recommended to implement the HTTPS protocol only.

The implementation of the HTTPS protocol on the Linux web server requires a digital server certificate. Administrators may choose to individually create a self-signed digital certificate or obtain a digital certificate from an official certificate authority. The creation of a self-signed digital certificate is simple and can be done quickly on the Linux server using the *OpenSSL* tool. The problem with self-signed digital certificates is that they are not trusted by any of the web browsers used by end users. When accessing a web server with a self-signed digital certificate, the web browser will warn the end user that the web server cannot be trusted. Such a warning is dangerous for the following two reasons: the users may get confused by the warning and decide against accessing the web server, or they can develop the risky habit of ignoring such warnings. Self-signed digital certificates should only be used in cases where administrators are not in the position to obtain a digital certificate from a certificate authority that is trusted by all web browsers. Redirecting all HTTP requests to HTTPS, i.e. rerouting the traffic from port 80 to port 443, is considered best practice in the HTTPS protocol implementation.

2.4 Using Secure Remote File Systems

It is often necessary that servers have additional free space for storing data collected during their operation. Administrators resort to using remote file systems in order to expand the capacity of their servers. For this purpose, Linux servers are connected to a remote server for data storage and "borrow" a part of the file system. The remote file system is accessed through the network and the system administrators have the same experience as though the remote file system was a part of the Linux server. The data transfer between the server and the remote file system must be adequately protected in order to avoid compromising the transferred data. System administrators are recommended to use the SSHFS (Secure Shell File System) that enables mounting of remote file systems over SSH/SFTP. Both manual mounting and automounting of remote file systems via SSHFS can be configured.

2.5 Keeping the System Updated

Being well informed is the key to the successful protection of the Linux server. The system administrator should ensure a reliable source of information about security trends concerning the relevant Linux distribution and the installed software packages. Various vulnerabilities are discovered over time in the software packages, and they can be used by attackers to compromise the server. Following the mailing lists that deal with security issues in Linux distributions and software packages enables the administrator to react promptly by updating the system on time and eliminate vulnerabilities. It is therefore recommended to follow

the security mailing lists for the corresponding Linux distributions (Red Hat, Debian, SuSE, etc.). Based on the obtained information, the administrator can decide when and how to update the system and software packages used on the server.

It is recommended to use a package manager such as *yum* (for Red Hat distributions) or *apt-get* (Debian distributions), which ensure the easier installation and updating of the software packages and relevant dependencies. They also provide for the easier removal of software packages. Package managers automatically check the digital signature of a package and refrain from installing it if the signature is invalid. Administrators should use the official software repositories recommended by the vendors. Software packages can also be retrieved from other repositories, but this requires additional precautionary measures. It is generally recommended to check for a newer version of a package before any update in order to check its stability and identify possible security issues. If it is necessary to install software that cannot be found in an official repository, it is recommended to retrieve the software package directly from the developer's website and check the validity of its digital signature. No unofficial software packages or packages in a testing phase should ever be installed on a production Linux server. Keeping a list of installed packages and their previous versions after every update is considered best practice. In this way, the administrator has an insight into the latest stable versions of the software used on the server. If a newer package threatens the stability of the server, the administrator can find the previous stable version of the software package in the list and reinstall it.

Production systems are generally not recommended to be configured for automatic software packages update. A better practice is to individually verify each package update. The administrator can also configure the package manager to send periodic email notifications listing all the packages that can be updated on the server. After reviewing the list of packages, the administrator can decide whether to update each package separately. The periodic sending of emails in the *Yum* package manager is configured in the */etc/yum/yum-updatesd.conf* configuration file. In addition to the check interval, the following line of code should be defined in the configuration file:

```
# how to send notifications (valid: dbus, email, syslog)
emit_via = email
# who to send email to
email_to = linux-administrator@example.org
```

It is necessary to restart the *yum-updatesd* service once the changes have been made to the configuration file.

Administrators using Debian family distributions are recommended to install and use the *apticron* software package to ensure automatic checks for package updates and email notifications. In the configuration file of the *apticron* software package, it is necessary to enter the email address the notifications will be sent to.

If the administrator is experienced enough to define the packages that are needed on the servers, it is considered good practice to create a local repository within the network. The repository of well-known and stable versions of packages can be created on one of the Linux servers and other servers on the network can retrieve the packages from there. Such a solution requires a detailed check of every package before it is stored in the local repository.

2.6 Limiting the Number of Services per Server

Linux servers can perform several functions and provide a number of services at the same time. This may lead to the more economical utilisation of the server infrastructure and maximum efficiency. Still, system administrators should observe the golden rule of security: "The system is as secure as the most vulnerable service in it." An ideal solution would be if each Linux server provided only one service to end users. Since it is difficult to put this scenario into practice, it is necessary to carefully plan the construction of the entire server architecture within a network. System administrators should define the services that will be provided to end users, classify the planned services in terms of priority, identify the set of available servers and finally distribute the services among the available servers. When distributing the services to the available servers, the most critical services should be run on separate Linux servers or combined with services that are considered the most secure. Less critical services can share one server or be combined with less secure services. The purpose behind this plan is to avoid a situation in which an exploited unsecured service would lead to a compromised server and the takeover of a critical service maintained by the administrator. It is generally recommended to keep web, email and SQL services apart and on separate servers.

3 Tools for Improving Security

These tools are used to prevent unauthorised access to services and malicious attempts to compromise the Linux server. Linux comes with built-in security tools such as *Netfilter* or *SELinux*. Netfilter is a utility that intercepts and processes network packets, which can be found on almost all Linux distributions with 2.4 or later kernel versions. It was introduced as a replacement for the older network security tools - *ipchains* and *ipfwadm*. *Netfilter* comprises a set of administration tools, among which *iptables* is most often used.

3.1 Iptables

The iptables tool includes five tables, each of which defines a different set of operations involving network packets:

- filter - performs packet filtering, used most frequently.
- nat - performs network address translation.
- mangle - processes packets in a specific manner (e.g. changing the TOS and TTL fields).
- raw - processes packets that precede and/or avoid the other defined tables.
- security - performs packet filtering according to the Mandatory Access Control (MAC).

The tables contain predefined chains of rules. Each rule comprises the condition a packet should meet and the action that is executed if the condition has been met. Packets that do not meet any of the conditions within a chain undergo the default action for the chain (the so-called policy). Unlike regular rules, policies are limited to accepting or dropping packets. The exact syntax and sequence of tables and chains can be found in the man pages and the official *Netfilter* documents. Using the *iptables* tool requires root privileges and it is most often used for filtering packets, translating addresses, capturing traffic, directly changing certain fields in packet headers and for traffic load balancing. It should be noted that *ip6tables* is applied to IPv6 traffic.

3.1.1 Security Recommendations for Iptables

The security recommendations provided in this document are focused on the criteria by which unwanted traffic is detected and rejected. It is generally recommended that the network protection be configured as strictly as possible, thereby reducing the possibility of exploiting potential vulnerabilities in the services hosted on the server. The parties that communicate should be described as precisely as possible, primarily in

terms of source and destination IP addresses and TCP/UDP ports. Packets should also be identified by other fields in the header, if their content is known in advance. For instance, ICMP does not have ports, but packages for this kind of traffic can be limited by the type and code of the messages relayed by the protocol.

Iptables enables the possibility of defining rules for manipulating packets depending on the side that initiated the connection. This functionality is provided by the connection tracking module (*conntrack*). In the case of more complex protocols such as FTP, SIP and H.323, it is recommended to use auxiliary *conntrack* modules (connection tracking helpers). Otherwise, an active FTP session, for instance, would require the client to open all incoming TCP ports greater than 1023, regardless of the connection control.

Any *iptables* configuration should be thoroughly tested before it is finally implemented. More complex or less clear rules should be extended to include comments (e.g. ***- m comment -- comment "Administrative IP address range"***). Related rules can be grouped into user-defined chains, as grouping increases readability and facilitates the maintenance of the iptables configuration.

3.1.2 Common Oversights in Configuring Iptables

The order of the rules in a chain is important. Rules containing stricter conditions should be placed closer to the beginning of the chain, otherwise there is a risk that packets never reach them. All changes to an *iptables* configuration should be saved in an explicit manner because the content of the tables is irretrievably lost once the system is restarted. Commands such **as *****/etc/init.d/iptables save*** on Red Hat distributions or the universal ***iptables-save*** and ***iptables-restore*** can be used in this respect. Before erasing a filter table (***iptables -F***) the policies defined for the chains in the table should be checked to avoid blocking desired traffic. More complex configurations require knowledge of the sequence in which packages traverse the tables and chains, wherefore it is necessary to check whether, for instance, per-port filtering is conducted before or after the traffic is redirected.

3.2 Tools used as a defence against ARP Spoofing Attacks

A device that receives an ARP reply will update its ARP table regardless of whether the reply was requested or not. ARP does not possess a mechanism to authenticate the exchanged messages, which has opened up the possibility of so-called spoofing (poisoning) attacks by which the attacker imposes its MAC address as the destination address for other devices in the network. ARP spoofing is usually used as an overture to more complex attacks such as man-in-the-middle, denial-of-service and session hijacking. There is no failsafe protection from this type of attack. The recommendations provided in this document may be of assistance in detecting and limiting such attacks. The safest means of protection comes in the form of static entries in the ARP tables, which are usually defined in the */etc/ethers* file. However, this solution is less practical in networks containing a large number of devices. The problem may be alleviated if the static entries are configured for important devices only, such as default gateway. The *arptables* tool is commonly used for limiting the pairs of MAC and IP addresses the server is to communicate with.

It is not uncommon to have network addresses dynamically assigned to end user devices within a network, which may create a favourable environment for ARP spoofing attacks. *Antidote* is an application that checks whether a previously learned MAC address is still active in the local network after a new ARP reply has been received. If it is, the new MAC address is placed on the list of forbidden addresses. The principal shortcoming of this algorithm is that it is based on the assumption that the current entry in the ARP table is legitimate, which does not prevent a competition between attackers and the legitimate destination when creating new entries. *Arpwatch* can be used to monitor ARP activities. This application maintains a local database of pairs of IP and MAC addresses, and notifies the administrator via syslog and emails if changes occur.

3.3 Tools Used as Defence Against Brute Force Attacks

One of the most common type of brute force attacks is SSH brute force attacks. Initial effective way of protection against SSH brute force attacks, can be changing the SSH access port from default 22 to some non-standard higher number port (e.g. 2202). This step is recommended to be done from the very beginning of putting any Linux server with public IP addresses in production, since it promptly eliminates the most standard brute force attack attempts. Of course relocation of the SSH port is not a panacea, since the scanning of other ports and attempts to access them will eventually enable attackers to find the "new" SSH port. Thus remote access should be protected with some additional methods described below.

It is recommended to limit access to the servers by SSH port to only trusted networks. A trusted network is typically some internal subnet, managed by the system administrator.

Access limitation can be done in multiple layers:

- Router/Firewall Rules
- Local *iptables* Rules
- Local SSH Configuration

Simultaneous use of multiple layers adds to the efficiency of protection, since the routers/firewalls and local Linux server are generally being administered separately.

If the administrator wants to enable remote access from outside, the best solution is to implement a VPN (Virtual Private Network). The administrator can connect to the VPN service in their network from anywhere in the world, obtain a local IP address and thus access the local server. The VPN service can be implemented on some separate Linux server using the free *OpenVPN* solution or with some separate specific device, like *MikroTik RouterBOARD*.

One additional step to be protected from SSH brute force attacks is the use of *iptables* tool to rate-limit the number of incoming connections within a specified time frame.

Besides SSH attacks there are situations when brute force attacks are not as obvious and when the aim of the attack is to take advantage of vulnerability in the software installed on the Linux server. In such cases, the protection of the server would require the following:

- Detection of the brute force attack.
- Blocking the source of the attack.

Fail2ban is an interesting tool ensuring flexibility in setting up protection mechanisms. *Fail2ban* monitors the information entered in the selected log files and activates the blocking in the *iptables* or *TCP wrapper* tools (the latter is described in Section 3.4) when it encounters specific lines in the log files that are indicative of an attack. In addition to detecting attacks and activating the blockades, *Fail2ban* can send emails to the system administrator about all the actions that are being taken and immediately notify the administrator about the attack. An additional advantage of this tool is that it contains predefined patterns for protecting the web, email, asterisk and other applications.

The implementation of the *Fail2ban* tool requires certain preparations. The system administrator should arrange the logging of information concerning the software they wish to protect and locate the log file in which the information will be entered. It is then necessary to analyse the data in the log file and select log lines that are indicative of an attack. Selecting these lines in the log file is not simple as there are a number of different attack vectors, but the administrator will usually rely on experience from previous attacks and thus identify the lines that are indicative of attempts to compromise the work of an application. The next step is to

configure the *Fail2ban* tool to search for such lines and count how many times they repeat in a specified period, and to configure the blocking for a defined time frame of the IP address behind the attacks. Parameters such as the number of repetitions of log lines, the period in which they repeat and the time frame in which the IP address is blocked are configurable. It should be noted that log lines that suggest an attempt to attack the server must contain information about the IP address of the attacker in order to be able to activate the IP address blocking. *Fail2ban* is usually used for IP address blocking after several failed web server login attempts.

Fail2ban should be used with some caution, bearing in mind the possibility of the so-called injection attacks, in which the attacker would create a false IP packet and misrepresent themselves as having the IP address of the victim. By generating a large number of such packets, the attacker can cause *Fail2ban* to configure the blocking of the victim's IP address. For instance, the victim can be the system administrator or another regular user of the server. It is for these reasons that administrators are recommended to first try and prevent, where possible, the arrival of false IP packets in the local network (i.e. to prevent spoofing attacks). Then, administrators should identify all IP addresses that are important for the work of the service being protected by the *Fail2ban* tool. Finally, they should use the functionality of the *Fail2ban* tool enabling them to define the set of IP addresses that will never be blocked. This is achieved by defining the *ignoreip* option in */etc/fail2ban/jail.conf* configuration file of the tool.

DenyHosts is a similar tool, but its use is limited to mitigating SSH attacks. Administrators are recommended to use *Fail2ban* as it enables simultaneous protection of several services on the server.

3.4 TCP Wrapper

TCP Wrapper is a tool for controlling access to certain daemons and services on the Linux server. This tool uses the *libwrap.so* library to implement the access rules. It operates in the application layer of the TCP/IP networking model and ensures additional protection of services to that provided by tools such as *iptables*. In order to implement the *TCP Wrapper* on a service, it is necessary for the relevant daemon running the service to support the *librwrap.so* library. This is checked by using the *ldd* command, and the example below shows the checking of the sshd daemon process:

```
#ldd /usr/sbin/sshd | grep libwrap*
```

In this example, *sshd* is located at */usr/sbin/sshd*. The *ldd* command checks whether *libwrap.so* is one of the libraries necessary for running the daemon process.

```
libwrap.so.0 => /lib64/libwrap.so.0 (0x00002b3f6617c000)
```

If the result of the ldd command is as shown in the above example, it can be concluded that the *sshd* daemon supports the *libwrap.so* library and that the *TCP Wrapper* protection can be implemented on the SSH service. The next step would be to configure the *TCP Wrapper*, which is done by entering changes in */etc/hosts.allow* and */etc/hosts.deny* files. The machines that are allowed to access the specified service are defined in the *hosts.allow* file, while machines that are explicitly prohibited from accessing the service are defined in the *hosts.deny* file. It should be noted that *TCP Wrapper* first checks the *etc/hosts.allow* file and then the */etc/hosts.deny* file. If a machine has been defined in both of the files, the */etc/hosts.allow* file takes precedence and the machine is granted access.

System administrators are recommended to use the *TCP Wrapper* tool, but never as a replacement for *iptables*. It should only be used as additional protection for individual services on the Linux server.

3.5 SELinux

SELinux (Security-Enhanced Linux) is a Linux kernel security module providing additional protection mechanisms for server operation. While the standard Linux kernel implements object access restrictions based on the identity of a user or group (the so-called Discretionary Access Control), *SELinux* has introduced the so-called Mandatory Access Control that limits the privileges of individual users or processes to the minimum necessary for their proper functioning.

SELinux has also introduced new entities for all users and processes on the server: role, user name and domain. Role and user name are not correlated to the standard user logins and groups on the Linux operating system. The entire mechanism of allowing certain actions is defined by the domain. Actions and privileges on the server are arranged into specific domains and a user becomes entitled to use them by being allowed to use the relevant domain. *SELinux* policies are defined for all users and processes and they determine which domains a user is allowed to enter.

The final purpose of configuring the policies for individual users and processes is to limit the scope of their actions. This is very useful as, for instance, it prevents an attacker who has compromised the web server from taking control of the rest of services on the server. SELinux is commonly used for limiting the scope of daemons and services that run web servers, email servers or database servers.

SELinux requires some time to grasp and become skilful enough to operate it. Limiting the processes that run on a server can cause the malfunctioning of certain applications on the server. For these reasons, administrators are recommended to familiarise themselves with the operation of *SELinux* in advance and to first configure it on servers providing only one service. After becoming well versed in managing this tool, they can use it on other servers. *SELinux* can be a very powerful security tool in the hands of a skilful system administrator.

3.6 AppArmor

AppArmor is a Linux kernel security module that can be found on Debian, SUSE and Mandriva distributions. Although it is conceptually very similar to the *SELinux* module, the focus of *AppArmor* is on restricting the capabilities of computer software, rather than of individual users. *AppArmor* functions on the basis of establishing profiles that can limit the activities of certain software or just report situations when the software takes actions outside its allowed scope. System administrators are recommended to familiarise themselves with the *AppArmor* security module if they find it necessary to restrict the capabilities of certain programmes.

4 User Management

No doubt, the most serious damage can be done by compromising user accounts. System administrators must be familiar with all local user accounts on the server. Therefore, it is necessary to have a strategy for creating and managing user accounts that will meet the following criteria:

- Each user on the Linux server must be uniquely identified by the user account.
- Each user account that will be used for login purposes must have a sufficiently secure password for authentication.
- The server must record log information about previously authenticated sessions of users.
- The activities of each user must be limited to the set of necessary privileges.

- Any unnecessary account should be disabled/locked
- System administrator is required not to use root access remotely, but login with own login and then obtain appropriate privileges with *sudo* (described below)

The administrator can use some of the built-in tools of the Linux operating system to automatically apply certain rules that increase the security of user accounts.

4.1 Creating User Password Policy

Almost all the Linux distributions that are in use today save passwords in a disguised format (shadow passwords). Using shadow passwords ensures that a user who is logged in cannot see the passwords of other users by reading the */etc/passwd* file. The administrator should check whether there is an */etc/shadow* file, which indicates that the Linux server relies on shadow passwords. If the Linux server is not using shadow passwords, the administrator is strongly recommended to implement such a solution.

Good passwords are essential for having secure user accounts. The system administrator should establish clear rules and restrictions that will make the users create strong passwords.

4.1.1 User Password Structure

User passwords should abide by clearly established rules that would make them more resistant to cracking. If there are a larger number of Linux server users, it is considered a good practice to prepare recommendations for creating passwords in the form of a document. The document would then be distributed to all the users who have accounts on the server. A password should meet the following criteria in order to be deemed sufficiently secure:

- it must not be a word or several words from any language put together.
- it must not be shorter than 12 characters.
- it must not be a name or surname.
- it must not be similar to the username or form a part of the user name.
- it must include at least one upper case letter.
- it must include at least one number.
- it must include at least one special character.

A good password could be made from the initial letters of a personal favourite phrase, paraphrased to become a combination of lower case and upper case letters, numbers and special characters. Such passwords are currently considered the most difficult to crack. System administrators can also apply tools that evaluate the strength of passwords to ensure that all the passwords on the Linux server are sufficiently secure. It is recommended to use the *John the Ripper* programme, which will try and crack the password for each account and mark those that have been successfully cracked. *John the Ripper* and similar tools generate a relatively high CPU load, so they should only be run during off-peak hours on the server.

4.1.2 Limitations in Creating Passwords

The PAM module (Pluggable Authentication Module) enables the administrator to establish rules that restrict users in reusing their previous passwords or parts thereof. PAM can be found on virtually all Linux distributions, and the limitations regarding old passwords can be set in the */etc/pam.d/system-auth* file in Red Hat family distributions or */etc/pam.d/common-password* file in Debian family distributions. By adding the following line in these files (or editing the existing line), the system will memorise the previous 10 passwords for each user:

password sufficient pam_unix.so use_authtok *md5* shadow remember=10

It is important to note that some Linux distributions may use the PAM 2 module, in which case it is necessary to enter *pam_unix2.so* instead of *pam_unix.so* in the above command. Also depending on the hashing algorithm used an existing entry with *sha512* instead of *md5* can be found. The Linux operating system will keep all the old passwords of the users in the */etc/security/opasswd* file. If there is no such file, the administrator will need to create it and set the privileges so that only the root account can read and write in it.

The system administrator can impose additional limitations, such as a minimum number of characters that need to be different from the previous password. Such a limitation is imposed by using the *pam_cracklib.so* module in the file */etc/pam.d/system-auth* or */etc/pam.d/common-password*, depending on the distribution.

password required pam_cracklib.so difok=4 minlen=12 dcredit=-1 ucredit=-1 ocredit=-1 lcredit=0

Adding this line in the files described above implements the rule that the password must contain at least 12 characters (minlen=12), which must include at least one digit (dcredit=-1), at least one upper case letter (ucredit=-1) and at least one special character (ocredit=-1), while at least four characters must be different between the old and the new password (difok=4). The *pam_cracklib.so* module uses a credit system that assigns a number of points to each character, depending on its type. The credit score must correspond to the minimum length of the password. If the credit system contains negative values, this means that the user must use a certain type of character the required number of times. Besides the above restrictions, the *pam_cracklib.so* module performs additional checks, such as whether the new password only reverses the order of characters of the old password, or whether the new password only contains changed letter cases in comparison to the old one.

4.1.3 Enforcing User Password Ageing

An additional safety mechanism is reflected in setting the time limits in which user passwords will be valid. Periodic changes of passwords for all users are configured in the */etc/login.defs* file. Using the ***chage*** command, the administrator can set parameters for individual users.

Information about the time attributes of a specific user's password can be obtained using the following command:

#chage –l *username*

The above command will display the time the password was created, the password expiry and the number of days of warning before the password expires. Displaying a warning a few days before the password expires is a very useful function, and it is recommended to set the warning period two to three weeks before the password finally expires.

The time limits for individual accounts can be set using the following command:

#chage –M 90 –m 7 –W 20 *username*

The above command sets the duration of the password to 90 days, while the minimum period of validity of the password is seven days, and the expiration warning will start to appear 20 days before the expiry of the password. The warning will be displayed when the user authenticates on the Linux server. The minimum validity period of the passwords is useful because after the password expires, a user could change it for a few minutes and then return to the old one. This would render the entire concept of password ageing practically meaningless. However, the mechanism that prohibits the user from resorting to the old passwiord would also solve this problem.

Time limitations for a specific user can be removed using the following command:

#chage *username*

The effectiveness of password ageing is questionable because strong passwords are difficult to create and memorise. Frequent changes of passwords may cause users to start writing them down, which is rather risky. For this reason it is widely believed that it is better to have one strong password than to frequently change weak passwords.

4.1.4 Locking User Accounts

The Linux operating system offers the possibility of locking a user account if an incorrect password is entered a predefined number of times. You can use the PAM module *pam_tally.so* to define the maximum number of attempts to enter the password and set the period for which the account will be locked afterwards. A locked account cannot be used to access the Linux server. This mechanism is implemented by making changes in the *etc/pam.d/system-auth* file in Red Hat distributions or */etc/pam.d/common-password* in Debian distributions. It is necessary to add the following lines:

auth required pam_tally.so no_magic_root

account required pam_tally.so deny=5 no_magic_root lock_time=3600

The *no_magic_root* option defines that it is not possible to lock the root account. The above example defines that failed attempts will be logged and sets a limit of up to 5 failed attempts, after which the account will be locked for the period of 3600 seconds. The information on failed login attempts can be obtained using the ***faillog*** command.

#faillog –u *username*

#faillog -a

The *-u* option gives information on a specified user, while the *-a* option displays information on all the users of the Linux operating system. The counters of failed loggins can be reset by using the *-r* option.

#faillog -r

If automatic user account locking is used, it is considered good practice to reset the counters occasionally. The entire process can be automated by entering the above command for resetting the counters in *crontab*. It is recommended to reset the counters daily or perhaps once a week.

In addition to automatic account locking, it is possible to lock accounts manually using the ***faillog*** command.

faillog –l 3600 -u *username*

The *-l* option locks an account for a time period specified in seconds, while the *-u* option determines which account will be locked.

Another simple method of manual locking a user account is using the following command

passwd –l *username*

To unlock manually locked user account the following command can be used

passwd –u *username*

4.2 Sudo Access

The root user account is very powerful and represents the so-called superuser that by default has unrestricted access to all the system processes and files in the Linux operating system. Red Hat family distributions require that the root account be created when installing the operating system, while Debian family distributions create the root account without assigning a password to it, so it is not possible to log into the Linux operating system using this account (an exception to this is Ubuntu, where root account is initially disabled by default and all administrative actions are being performed via sudo). Regardless of the distribution, it is generally recommended to never use the root account for accessing the server. If several users used the root account for accessing and maintaining the server, it would be impossible to know who was responsible for certain settings on the server. It is considered better practice if each user has their own user account for accessing the system and managing the server. With the ***sudo*** command, regular user accounts can be granted temporary authorisation to execute commands for which they otherwise do not have privileges.

Sudo (Super User Do) offers a mechanism for providing trusted users with administrative access to a system without sharing the password of the root user. When users given access via this mechanism precede an administrative command with *sudo* they are prompted to enter *their own* password. Once authenticated, and assuming the command is permitted, the administrative command is executed as if run by the root user.

The *sudo* mechanism is configured in the */etc/sudoers* file.

/etc/sudoers file can be configured to request the root password instead of the account password of the user. This setting can be useful as it introduces an additional level of protection if the relevant user account is compromised. If *sudo* requests the root password, it is not enough to just know the credentials of the regular user, but one also needs to know the root account password. Configuring the ***sudo*** command to include the root password is done by adding the following line in the */etc/sudoers* file:

Defaults rootpw

Depending on the configuration of the */etc/sudoers* file, different users can have different privileges. An overview of all the root commands granted to a specific user account can be obtained using the following command:

sudo -ll

The *sudo* tool is very useful when it is necessary to limit the influence of certain users. It might be good practice to distribute responsibilities among the Linux system users and to configure the *sudo* tool accordingly in order to ensure that no user oversteps his or her authority.

4.3 Monitoring Server Access

Information on user logins in the Linux server can be very useful in potential investigations and for establishing responsibility for incidents. The Linux operating system possesses built-in mechanisms that monitor user login.

The ***w***, ***last*** and ***lastb*** commands provide an overview of all the logged in users, regardless of whether a user logged in via Telnet, SSH or console. Bearing in mind that the SSH protocol is recommended for remote access, the system administrator can also use the information contained in the */var/log/secure* log file that monitors all the SSH login attempts. The ***lastlog*** command can also come in handy, as it displays the time of last login for each user.

4.4 Monitoring Executed Commands

The Linux operating system comes with a built-in mechanism that provides an overview of all the executed commands. The list of executed commands can be obtained using the ***history*** command.

The commands are listed in the order of execution, but there is no information on the exact time a certain command was run. In order to get the information on the time, it is necessary to add the following line in the */etc/profile* file:

export HISTTIMEFORMAT='[%F] [%T] '

The ***history*** command lists the commands executed by the system administrator, but sometimes it is necessary to establish what commands were run by other users. The hidden file called *.bash_history*, which is located in the */home/* directory of each user, provides insight into the commands executed by other users in the Linux operating system.

Advanced users can cover their tracks by changing the content of the *.bash_history* file and manipulating the data on the executed commands. If the Linux server is used by advanced users, a good tool for monitoring user activities is *psacct* (Process Accounting). The *psacct* tool keeps track of how long users have been logged into the system and the commands they have executed. The tool is available on all commonly used Linux distributions and system administrators are recommended to use it if they want to have reliable information on the activities of all users. After the tool has been installed, the service is started automatically on Debian distributions, while Red Hat distributions require it to be manually started after installation.

The tool provides some other interesting data, such as the RAM memory load and CPU utilisation per user and per command. Relying on this tool, the system administrator can gain a good insight into user activities, which is why it is recommended when one Linux server is administered by several users.

4.5 LDAP Authentication

User account maintenance on a single Linux server is not too great a task for the system administrator. However, if it is necessary to maintain user accounts on several Linux servers, there is a risk of creating a chaotic system where user accounts will not be consistent on different Linux servers. Each additional Linux server brings additional administrative work in terms of managing individual user accounts. A new problem can arise in the form of removing a user account, as it would have to be done on several servers. A centralised authentication system greatly reduces the problem of handling user accounts on several Linux servers. All accounts are kept in one place and they are synchronised on all Linux servers. It is recommended to use the LDAP (Lightweight Directory Access Protocol) directory for the centralised management of all user accounts authorised to connect to a Linux server.

Linux treats the user accounts in the remote LDAP directory the same way as if they were in the */etc/shadow* or */etc/passwd* files. A special PAM module - *pam_ldap.so* - is needed to set up LDAP authentication on the Linux operating system. The SSH access should be configured to support PAM modules, and configuring the PAM authentication and LDAP service on the Linux operating system sets the parameters of communication between the Linux servers and the remote LDAP server. This results in a system in which users access the Linux server through the Secure Shell Protocol (SSH) and the Linux operating system checks the user credentials and privileges in the centralised LDAP directory on the remote server. Administrators are recommended to use the LDAPS protocol (LDAP Secure) for the exchange of user account information between the Linux server and the remote LDAP server in this type of architecture, in order to ensure the secure transfer of user credentials through the network.

5 System Monitoring

Overseeing the work of a server and paying attention to the vital system information is the absolute priority in ensuring a stable and secure operation of the server. Information on the server performance is the key to stability, which is why the administrator has the important task of gathering the following information:

- Information on the status of applications on the server.
- Information on the network activity of the server.
- Information on the use of system resources.
- Information on the status of the operating system.

The information gathered can point to errors in configuring the Linux server, and it can also lead to the detection of security threats faced by the server such as:

- The unusual operation of an application, indicating an ongoing attack.
- Network activity of the server can indicate an attack and help discover its source.
- An abrupt surge in the use of system resources can also indicate an attack.
- The unstable performance of the operating system can indicate that the server has been compromised.

By continually monitoring the operation of the Linux server, the administrator becomes more experienced and intuitive about the routine behaviour of the server. Any information that indicates irregularities in the work of the server should be looked into as it may reveal vulnerabilities in the system or an attack on the Linux server.

5.1 Syslog

Logs contain invaluable information that can reveal early signs of the system being compromised. Additionally, in the case of security incidents, logs provide important forensic data that indicate vulnerabilities in the system and helps to reconstruct the course of events that led to a server security breach. *Syslog* is an unavoidable tool for monitoring the operation of servers, and one of its advantages is that it is included in all the Linux distributions. *Syslog* collects and filters the information about the kernel, local processes and applications. Log information filtering is flexible and enables the administrator to decide what information to log, to what extent and in which location.

The *syslog* tool on the Linux operating system is usually configured in the /etc/syslog.conf file. The lines in the configuration file consist of two elements: a selector and an action. The selector is structured in the "facility.priority" format and it defines the data that will be recorded and its extent. The facility defines the category of information to be collected, and administrators are recommended to collect information from the following categories (facilities):

- Auth – contains messages about the authentication of users and about events related to the security of the server.
- Authpriv – contains messages about server access control.
- Daemon – contains messages about the system and daemon processes.
- Kern – contains messages about kernel-related activities.

Depending on the requirements and purpose of the server, other facilities may be included, but the ones listed above provide information that can be significant in terms of security. In addition to the facilities, it is necessary to define the priority of messages that will be logged. *Syslog* differentiates between 8 levels of priority of messages: emergencies (0), alerts (1), critical (2), errors (3), warnings (4), notifications (5), informational (6) and debugging (7), where a lower number indicates a higher priority level. It should be noted that logging a large number of messages impacts the work of the server because the processing and recording of log messages utilises some of the CPU resources. Administrators are advised to select at least level 3 priority (errors) as this level logs messages concerning errors in processes and applications. Error messages are indicators of the instability of the system, and the source of the instability may be a security incident. If it is suspected that there is a security risk for the system, the administrator can temporarily enable the debugging level (7) to log all messages and get a complete picture of the performance of a given process or application.

A typical line in the */etc/syslog.conf* configuration would be:

auth.err /var/log/auth.log

The above configuration line defines the logging of messages regarding authentication (facility), of level 3 priority (errors) and all higher levels of priority. The log messages are recorded in the */var/log/auth.log* file (action). *Syslog* can also be configured to display messages on the terminal of all or certain users logged into the system, but this setting is only recommended for the highest priority levels, i.e. in the case of critical errors that interrupt processes or applications. The advantage of displaying log messages on the terminal is that the information cannot be altered, which is the case when they are entered in the log file. Anyone who has access to the log file can change or erase certain information in the file and thus cover their tracks. A shortcoming of displaying log messages on the terminal is that they are not presented clearly and make the work on the server more difficult as they can interrupt the activities of the system administrator.

Some applications do not rely on *syslog* for collecting log messages and they need to be configured separately in the configuration file of the application. A typical example is the *Apache* application for web servers, whose logging capabilities are configured in the */etc/httpd/httpd.conf* file or the */etc/apache2/apache2.conf* file for the Apache2 version of the package. Administrators are strongly advised to log messages on the performance of applications, whether through the syslog tool or directly.

5.2 Log Rotation

If not controlled log files may grow without bound until you run out of disk space. The solution is to use log rotation: a scheme whereby existing log files are periodically renamed and ultimately deleted. Most Linux distributions come with a program called *logrotate*, which should be run daily by *cron* (*/etc/cron.daily/logrotate*). *logrotate* can be configured with */etc/logrotate.conf* to perform rotation on any or all log files. Although main config file for *logrotate* is */etc/logrotate.conf*, it also includes all files from */etc/logrotate.d/* directory. This way *logrotate* rotates files not only for *syslog*, but for many other services.

Typical *logrotate* config defines the rotation period (how often it should be rotated) for each log file, as well as the number of previous copies to be kept. So there is a risk of loosing previous log information if by default logs are configured to be rotated too quickly (for example daily) and only few number of previous copies are kept (for example seven). Thus it's very important for administrator to review all *logrotate* config files and

ensure, that both rotation period for each log file, as well as the number of previous copies to be kept fits current needs.

5.3 Centralised Syslog Monitoring

Generally speaking, information on the performance of servers is recorded locally on each individual server. If the administrator manages several servers, it can prove difficult to monitor the work of different systems at the same time. A good solution in such cases can be to send all log messages to a central location – typically a Linux server dedicated to storing this type of information.

Syslog protocol as, defined in RFC 5424 (*https://tools.ietf.org/html/rfc5424*), is widely used by network devices to send logs to a centralized logging server.

Syslog supports the sending of log messages to a remote syslog server, and the settings are entered in the */etc/syslog.conf* configuration file.

*.err @syslog-server.ac.rs

The above configuration line defines that all *syslog* messages of level 3 priority (errors) and higher will be sent to the remote server *syslog-server.ac.rs.* The remote server must also have its own *syslog* tool configured to receive logs from the remote machines. Administrators are strongly advised to limit access to the central *syslog* server, and the most efficient way to do so is by using the *iptables* tool. The central *syslog* server usually receives *syslog* messages on the UDP destination port 514. Access to the server on this port should only be allowed to machines that use the services of the central *syslog* server.

The simplest log architecture is to have single syslog server per data center, assuming the log server is powerful enough to handle all the logs from the systems in the data center. Suchs design is not a scalable and is also not fault tolerant. It can be made more scalable and fault tolerant by adding some load balancer and increasing the number of syslog servers or by periodically sending copies of log messages from main log server to the secondary. Any solution depends on the local needs.

5.4 Other Loging Alternatives

Syslog-ng (Syslog Next Generation) is an upgraded version of the basic *syslog* tool. In addition to filtering and classifying log data, it offers parsing and formatting of *syslog* messages. Parsing and formatting enable system administrators to adapt log messages to their needs as they can remove unnecessary data or rewrite certain parts of the messages. The advantages of *syslog-ng* also include the possibility of storing log messages in SQL databases, which enables their integration with applications that analyse log messages. Using an SQL database to record log information provides a better overview of the relevant data and enables the administrator to generate statistics of the performance of individual processes, applications or entire devices.

Syslog-ng has introduced improvements in terms of security during the transfer of information to the central *syslog* server. The reliable transfer of information is conducted via the TCP protocol, as opposed to the unreliable UDP protocol that is used by the basic *syslog* tool. Additionally, the new tool provides the possibility of the encrypted transfer of *syslog* packets by using the TLS protocol, which prevents an attacker from intercepting and reading log messages sent by various devices. Due to the above advantages, system

administrators are recommended to use the *syslog-ng* tool in architectures that rely on centralised monitoring.

An alternative to the *syslog-ng* tool could be *rsyslog,* which can be found on most of the commonly used Linux distributions. This tool is an upgraded version of the basic *syslog* tool, and it includes the reliable TCP transfer of data to a remote server, the storing of messages in a local buffer if the remote server is temporarily unavailable, integration with databases and additional options for the precise filtering of log messages.

Other interesting logging solutions are listed below:

• systemd-journald

• Splunk

• Fluentd

• Logstash

• Graylog2

• Apache Flume

• Logback

• Log4j

Administrators are recommended to at least become familiar with the above solutions to be able to choose the best solutions for their local needs.

5.5 Log Analysis

Log analysis is a crucial component of network infrastructure, consisting mostly of search and reporting for specific messages. There are some practical tools for log analysis.

Logwatch is an application that helps with simple log management by daily analyzing and reporting a short digest from activities taking place on your machine. Reports created by Logwatch are categorised by services (i.e. applications) running on your system, which can be configured to consist of the ones you like or all of them together by modifying its relatively simple configuration file. Furthermore, Logwatch allows the creation of custom analysis scripts for specific needs.

Swatch is a tool for actively monitoring log files. Swatch is controlled by a single file, default $HOME/.swatchrc. This file contains text patterns in the form of regular expressions you wish swatch to watch for. Each regular expression is followed by the action(s) you wish swatch to take whenever it encounters that text.

Adiscon LogAnalyzer is a web interface to syslog and other network event data. It provides easy browsing, analysis of realtime network events and reporting services.

Tenshi is a log monitoring program, designed to watch one or more log files for lines matching user defined regular expressions and report on the matches. The regular expressions are assigned to queues which have an alert interval and a list of mail recipients.

5.6 SNMP

SNMP (Simple Network Management Protocol) is a protocol used for the transfer of data on various performance parameters of remote machines. It is often used for monitoring the volume of network traffic, CPU load, RAM memory use, or storage space on remote Linux servers. The data obtained through the SNMP protocol should be constantly monitored as they are often the first indicators of changes in the performance of a server. A sudden surge in the utilisation of system resources is a primary indicator of the instability of the system, which prompts the administrator to inspect the suspicious server and check the logs collected on it. It is often the case that a security incident is first detected based on data collected via the SNMP protocol.

Today, the most frequently used versions of the SNMP protocol are v2, v2c and v3. Version 2 introduced a new functionality compared to version 1, as it uses the Inform message instead of Trap. Both messages notify the remote client or application about an event on the server that requires special attention. Unlike the Trap message, Inform demands a confirmation of the received message, which ensures that the message is reliably delivered to the remote client. Version v2c is very similar to the basic version, but it made the security aspect of the protocol less complex and relied on authentication through the Community attribute. It is because of the simpler security model that v2c is much more widely used than the basic version 2.

Version 3 is the latest version of the SNMP protocol that has brought improvements to the security system by introducing identifiers for each device (entity) on the SNMP network. Each device possesses a unique EngineID and uses its own key to authenticate messages. The authentication is implemented in order to exchange SNMP messages between the intended entities only. Also, version 3 has introduced the encryption of SNMP messages, ensuring the confidentiality of the transferred messages. The reliable transfer of messages, authentication, integrity check and the encryption of SNMP messages make version 3 of the SNMP protocol recommendable to system administrators.

6 Protection from Most Common Vulnerabilities

Although Linux servers can be used for various purposes, they are typically used as web servers, email servers, DNS servers or database servers. The applications that support these services can be susceptible to security problems that attackers can use to compromise the operation of the server. Over years of usage, these services have occasionally shown various security flaws, which have been resolved on an *ad hoc* basis by new patches and new versions of software. However, the architecture of the protocols supporting these services could be constructed in such a manner that the inadequate installation or configuration of services on the Linux server can leave the system vulnerable and susceptible to attacks from the Internet. This Section will address some of the most common attacks on Linux servers and their prevention.

6.1 Hardening User Accounts

Attempts to break into user accounts and get remote access to the server are probably one of the most commonly attempted attacks on Linux servers. Attackers usually scan for open ports on the server and if they detect open ports 22 (SSH) and 23 (telnet), they will try to connect to the server using standard user credentials. They typically first try with the user name "root" or names (John, Michael, George), using various typical values as passwords (pass123, john123456, etc.). This type of attack is called the "dictionary attack", where different words in English and other languages are tried out as passwords, including common combinations of names and numbers. Surprisingly, the number of compromised accounts is not small despite many years of warning users of the rules for creating usernames and passwords.

Defence against these attacks has been discussed above (in Section 3.3). The bottom line is that remote access to the server should not be allowed using the "root" account. Only secure remote access (SSH) should be enabled. User passwords need to be sufficiently strong and they should not be a combination of words and names commonly used in the spoken language.

6.2 DDoS Attack Mitigation

Distributed Denial of Service (DDoS) attacks today use DNS reflection and amplification to achieve attack data bit rates up to 300 gigabits per second (Gbps) and even more. Underlying many of these attacks is packet-level source address forgery or spoofing, a well-known vulnerability in which an attacker generates and transmits User Diagram Protocol (UDP) packets purporting to be from the victim's IP address. Attackers often use query-response protocols, such as DNS to reflect or amplify responses to achieve attack data transfer rates exceeding the victim's network capacity either in bits per second, packets per second, or both.

DNS is especially suitable for such attacks because the response is typically larger, and in some cases, much larger than the query.

6.2.1 DNS Amplification Attack

A Domain Name Server (DNS) Amplification attack is a popular form of Distributed Denial of Service (DDoS), in which attackers use publically accessible open DNS servers to flood a target system with DNS response traffic. The primary technique consists of an attacker sending a DNS name lookup request to an open DNS server with the source address spoofed to be the target's address. When the DNS server sends the DNS record response, it is sent instead to the target. Attackers will typically submit a request for as much zone information as possible to maximize the amplification effect. In most attacks of this type, the spoofed queries sent by the attacker are of the type "ANY," which returns all known information about a DNS zone in a single request. Because the size of the response is considerably larger than the request, the attacker is able to

increase the amount of traffic directed at the victim. By leveraging a botnet to produce a large number of spoofed DNS queries, an attacker can create an immense amount of traffic with little effort.

Additionally, because the responses are legitimate data coming from valid servers, it is extremely difficult to prevent these types of attacks. While the attacks are difficult to stop, network operators can apply several possible mitigation strategies. Best practice configuration of protection methods for the most widely used Name Server Software - "Berkeley Internet Name Domain" (BIND9) package is presented below.

First of all the network infrastructure must give high quality (smallest delay and reliability) of DNS traffic, to provide customers with comfortable browsing and downloading. Next DNS system must be divided into two subsystems:

1. Authoritative non-recursive DNS servers, serving own domains zones to the outside world.
2. Local recursive DNS resolvers for customers.

Restricted access to the outside public DNS servers must also be provided (accounting that their response time is much longer than that of local DNS resolvers).

Authoritative non-recursive DNS servers preferably should be located in a demilitarized zone (DMZ).

Local recursive DNS resolvers preferably should be located inside LAN of each organization not to cross the border routers with customers' requests.

Each DNS server should have SNMP support enabled, to monitor DNS traffic activity. Average normal activity must be calculated and used for local firewall setting.

6.2.1.1 Authoritative DNS Servers Protection

Authoritative DNS servers are deployed to provide name resolution for hosted domains. As stated above DNS resolution for private client systems should be provided by a separate server and the authoritative name server should act only as a DNS source of zones information to external clients. Thus, these systems do not need to support recursive resolution of other domains on behalf of a client, and should be configured with recursion disabled.

It is strongly recommended for BIND9 global options to contain the following lines:

```
allow-query-cache { none; };
recursion no;
```

A very important feature available in recent versions of BIND9 allows an administrator to limit the maximum number of responses per second being sent to one client from the name server. This functionality named Response Rate Limiting (RRL) is intended to be used on authoritative domain name servers only when it affects the performance on recursive resolvers. To provide the most effective protection, it is strongly recommended for BIND9 global options to contain the following lines to implement RRL:

```
rate-limit {
   responses-per-second 5;
   window 5;
};
```

6.2.1.2 Local DNS Resolvers Protection

Since customers' PCs, infected with botnet agents create DNS requests to local DNS servers and further to victims or directly to victims, the router's firewalls must be activated to filter the outgoing external DNS traffic. Limiting a Query Per Second (QPS) locally helps to distribute the load and not to overload border routers in case of massive attack. As the practice shows, local DNS requests filtering for customers has a good effect. In normal situation each customer's PC generates not more than 100 QPS. Whereas each botnet agent infected customer PC can create up to 20 000 QPS.

Thus it is strongly recommended for BIND9 global options to contain the following lines:

```
acl INT_IP_RANGE { [INTERNAL-IP-RANGE] };
allow-query { INT_IP_RANGE; };
allow-query-cache { INT_IP_RANGE; };
allow-recursion { INT_IP_RANGE; };
rate-limit {
   responses-per-second 100;
   window 5;
};
```

6.2.2 NTP Reflection Attack

NTP (Network Time Protocol) is a protocol that enables the time synchronisation of devices on a network. Typically, all devices are synchronised to a central node in the network. *NTPD* is an NTP daemon process in the Linux operating system that is responsible for the NTP protocol implementation and can be found in all Linux distributions. The exact time on the server is very important since all log information is recorded with a time reference so administrators can have a clear picture of the time an event occurred on the server. System administrators are required to use the NTP service on the Linux servers so they can follow the timeline of events on the server.

An NTP reflection attack is very similar to a DNS amplification attack since a small query of the attacker generates a much larger response by the server. As in the DNS amplification attack, the attacker uses the vulnerable Linux server as an intermediary in the attack by using the ***monolist*** command in the NTP service. The attacker fakes an IP packet pretending to be the victim and sends the ***monolist*** command to the vulnerable Linux server. The server responds to the victim's IP address by providing a list of the last 600 devices that had an NTP connection with the vulnerable Linux server. As a result, the victim receives unwanted traffic that may jeopardise its functioning.

This problem can be resolved by obtaining a newer version of the NTPD package – 4.2.7 and later. The administrator should check the version of the NTPD package on the Linux server and if an earlier, vulnerable version is used, the package should immediately be updated to a more secure version. If for any reason this is not possible, an alternative solution is to turn off the ***monolist*** command in the NTP service. The following options should be added to the *restrict default* lines in the NTPD configuration file:

```
restrict default kod nomodify notrap nopeer noquery
restrict-6 default kod nomodify notrap nopeer noquery
```

6.3 Heartbleed Vulnerability

Heartbleed is a recently discovered vulnerability that occurs in the *OpenSSL* tool. The *OpenSSL* tool is very popular open-source software that can be found on two thirds of the web servers on the Internet, and is used to implement the SSL/TLS protocol. *OpenSSL* is typically used on servers to establish HTTPS connections, to enable the operation of the VPN server, or to facilitate secure exchange of data between RADIUS servers. The Heartbleed vulnerability enables an attacker to access the 64KB RAM memory through the established SSL/TLS connection with the vulnerable server, and thus possibly steal the last logged-in user credentials or recognise and read the private server key. By obtaining the private server key, the attacker can decrypt the traffic that other users exchange with the server and in this way get to confidential information, e.g. user credentials or financial transactions. This vulnerability is historically probably the most serious since the Internet has become widely used and the consequences of the use of a vulnerable OpenSSL tool can be disastrous.

The problem can be resolved by using a version of the OpenSSL tool that is not susceptible to this problem. The system administrator should make sure that versions of the OpenSSL tool from 1.0.1 to 1.0.1f are not used on the server. If any of the vulnerable tools are being used, it should be immediately updated to a newer, safer version. The system administrator should then change the private key and the digital certificate used on the server, and then change all the user passwords that were previously exchanged with the server through the SSL/TLS connection.

6.4 Bash ShellShock Vulnerability

Shellshock (also known as Bashdoor) name refers to a series of vulnerabilities found in the widely used GNU's Bash shell, that gives attackers access to run remote commands on a vulnerable system.

It is important to have the production Linux server protected from this type of vulnerability, even if no shell access is provided to anyone except the administrators. The reason is that shell environment could be exploited in some non-direct ways (such as via some web server scripting interface).

In order to get the most up to date version of bash available, depending on the package manager (yum, apt-get, etc) something similar to the following should be done.

For RedHat Faimly Distributions:

yum update bash -y

For Debian Faimly Distributions:

apt-get update; apt-get install --only-upgrade bash

6.5 Disabling Unneeded IPv6 Protocol

Internet Protocol version 6 should in the future replace IPv4 that is now predominantly used on the internet. IPv6 offers a number of advantages, but unfortunately it has not yet assumed a significant role in Internet traffic. It seems that IPv6 has been somewhat neglected when it comes to the security testing of various applications that are commonly used on the Linux operating system. An additional problem is reflected in the fact that IPv6 support is automatically activated on almost all Linux distributions, which administrators often forget and thus potentially leave the Linux server fully open to attacks through the IPv6 protocol. The commonest mistake in configuring the Linux server is neglecting the IPv6 protocol and setting up security

mechanisms for only IPv4 traffic. Administrators usually configure the rules for filtering the IPv4 traffic, while the IPv6 traffic can remain unfiltered and provide attackers with a window for compromising the server.

If the services hosted on the Linux server are not intended for IPv6 traffic, administrators are advised to implement a strict network policy by using the *ip6tables* tool or by disabling IPv6 support at the operating system level. Disabling IPv6 support is done by adding the following lines in the */etc/sysctl.conf* file:

net.ipv6.conf.all.disable_ipv6 = 1

net.ipv6.conf.default.disable_ipv6 = 1

net.ipv6.conf.lo.disable_ipv6 = 1

The above lines do not disable IPv6 in the kernel, they only prevent the assigning and using of IPv6 addresses on network interfaces. It is not recommended to disable the IPv6 module in the kernel as it may cause problems with certain applications that attempt to use IPv6.

If administrators opt for disabling the IPv6 support on network interfaces, they should comment out all IPv6 addresses in the */etc/hosts* file. After the changes have been made, it is necessary to restart the Linux operating system.

7 Linux Server Performance Diagnostics

While maintaining the server, the system administrator often needs to check the current state of the operating system – either to check a network part, a system part or log files. These checks are not part of the typical server performance monitoring, but rather diagnostics of the current state of the server. Although Linux incorporates numerous tools for this purpose, there are additional tools that provide better organised information. The system diagnostic tools are usually used in inspecting the performance of certain applications or services, or in examining security incidents. System administrators are advised to try out different tools and to choose the ones that suit them best.

7.1 Checking System Resources Utilisation

A check of system resources utilisation is performed when it is suspected that the Linux server is unjustifiably overloaded. The source of the increased utilisation of the system resources can be a certain process, while the reason behind it may be a security incident on the server. The administrator should find out which process is responsible for the increased load on the server and then, through further examination of the process, identify the reason for such behaviour. Various Linux tools that examine the utilisation of the system resources can be useful in diagnosing the problem.

7.1.1 System Resources Utilisation

The Linux server can sometimes experience difficulties in operation and perform anticipated functions slower than usual. The administrator's first task is to check the utilisation of the system resources in order to establish whether there are any discrepancies from the usual values. Typically, the administrator would check the RAM and CPU usage, and then the performance of the data storage disks. A large number of tools are available to help the system administrator quickly identify the part of the server that is over-utilised.

The ***free*** command shows the current RAM usage and it can also present usage of the swap memory.

The ***vmstat*** command shows information on RAM usage, blocks of entries and deletions, and the CPU activities. The command can also include an interval in seconds, after which the latest results on system resources utilisation will be presented.

The ***uptime*** command provides information on the time elapsed since the last boot of the Linux operating system and information on the number of users currently logged into the Linux operating system. Finally, this command provides the value of the server load in the previous period. Load values vary from server to server, but the generally acceptable value is up to 3 for servers with one CPU and up to 10 for servers with multiple CPUs.

The ***iostat*** command provides an overview of the CPU usage and the statistics on disk and partition activities on the Linux server. A tool that provides similar data is *dstat*.

The ***sar*** command presents information on the CPU power consumption at periodic intervals. The ***mpstat*** command is very similar to this tool except that it presents the information broken down by individual CPU units.

7.1.2 The Activity of Individual Processes

After detecting an excessive utilisation of the system resources, the next logical step is to identify the processes responsible for the Linux server load. There are a number of useful tools and commands for this purpose, and this section will describe the most commonly used ones.

The ***top*** command lists the CPU usage information by individual processes in the Linux operating system. The processes are listed from the most to the least demanding. There are numerous tools similar to this command, such as *htop* or *atop*, which offer a more appealing and better organised overview of the information, and in addition can provide more information on individual processes. The *atop* tool is particularly interesting since it can store information on resource utilisation in log files so the system administrator can compare the current consumption to the consumption of several days ago. The *htop* tool has the same form as *top* except that it analyses input/output activity on the disk by active processes.

The *nmon* tool presents overall information on resource utilisation, as well information on utilisation by processes. This is a very attractive tool that enables the administrator to use keyboard commands and move through various statistical data in order to get information on the usage of the CPU, memory and storage space. This tool also identifies the most resource intensive processes in the system.

The *glances* tool is similar to nmon and it represents a complete solution that provides information on the overall utilisation of system resources, as well as information on utilisation by processes. This is an attractive and well-organised solution that includes several tools such as *top*, *iostat*, *vmstat* and other similar tools.

The ***ps*** command lists all the active processes in the Linux operating system. This command provides all the important information on active processes, it is highly configurable, and it also offers the possibility of showing the most demanding processes by memory usage or CPU time.

The ***pmap*** command presents memory usage by the chosen process and it is often used in combination with the ***ps*** command in order to get precise information on memory consumption or by process.

The ***lsof*** command presents a list of open files and the processes that have opened them. It can also show a list of open network sockets by processes.

7.2 Network Status Check

The majority of security problems come through the network so it is necessary to have an insight into the current network activities of the Linux server and the interaction of the server with other devices, as well as to identify open ports and services. The administrator should use the tools that will provide information on the network traffic exchange between the server and remote devices.

The ***netstat*** command presents information on active connections, the server listening ports, the routing table and the network traffic statistics. This is a powerful command that provides the system administrator with all the important information on the network traffic exchange between the server and other devices in the network. In addition to the ***netstat*** command, it is also recommended to use the ***ss*** (Socket Statistics) command that provides statistical data on connections in a similar format.

The *iftop* tool presents information on the quantity of the traffic exchange by individual active connections that the Linux server establishes with remote devices in the network. What the *top* tool presents in terms of the CPU consumption by active processes, the *iftop* tool presents in terms of the quantity of traffic exchange by active connections. *Nload* is a similar tool that provides a basic graphic overview of the incoming and outgoing traffic, making it easier to detect anomalies in the network exchange of information.

The *iptraf* tool is a comprehensive tool that monitors the network activity on the server. It collects statistical data on the traffic exchange, individual connections, network activity on individual interfaces, and information by ports or packet sizes. It also provides information on the traffic exchange with individual MAC addresses, which can be useful.

The *NetHogs* tool lists information on the network traffic of each individual process in the Linux operating system and it can be helpful when examining the behaviour of individual processes.

The *arpwatch* tool monitors the ARP protocol on the network where the Linux server is located and records changes in the pairs of IP and MAC addresses. It can be very for detecting ARP spoofing attacks on the network.

The *apachetop* tool is intended for monitoring HTTP requests received by the *Apache* web server. This tool presents the statistics on the requested URL addresses and the flow, thus providing a good general picture of the activities of the web server.

GoAccess is similar to the *apachetop* tool, although it provides more information on the traffic exchange with the web server. The system administrator can see which remote operating systems or Internet browsers are accessing the web server, as well as the most active IP addresses and the most requested URL addresses or files on the web server.

The ***tcpdump*** command is very useful for monitoring certain network traffic on the server. This command is intended for inspecting individual packets arriving at the server and enables the most precise examination of the Linux server network traffic. The *Wireshark* tool, which has the same purpose, can serve as an alternative to this command.

Nmap is an irreplaceable tool in testing network services and open ports. It is sufficient to have the *nmap* tool installed on just one of the Linux servers as it can scan all the other devices in the network. Administrators are advised to use this tool in order to test whether the server ports access limitations are working properly.

7.3 Checking Typical Log Information

Log files provide an overview of the operation of individual processes and applications on the Linux server. They can be set as summary or detailed log files, depending on the level of detail the administrator wants to

record. System administrators are encouraged to check the log files when the server is running without any problems in order to learn how processes and applications operate and which log messages characterise the stable operation of the server. When a problem occurs, it will be too late to gain a general understanding of the work of individual processes. Log files in the Linux distributions are typically found in the */var/log/* directory, and some of the basic files are the following:

- */var/log/messages* – General information on the operation of the Linux system.
- */var/log/boot.log* – Information on the processes active when the system is booting.
- */var/log/auth.log* or */var/log/secure* – Information on user authentication.
- */var/log/yum.log* or */var/log/dpkg.log* – Information on package installation.
- */var/log/mail.log* – Information on the operation of the mail server.

The system administrator needs to set the recording of log information in the syslog tool as defined in Section 5. Certain applications will individually create log files and will typically place them in the */var/log/* directory. Administrators are advised to periodically review the log information to see if there are any new developments in the operation of the server.

8 Backup Procedures

Creating backup copies is a very important aspect of server security maintenance. It is always good to have the possibility of retrieving lost files. The reasons behind file loss vary from a compromised service on the server to unintentional mistakes made by users. The goal of implementing backup procedures is to avoid reinitialisation of the entire system, to recover the server in crisis situations and to reduce any down time.

8.1 The Strategy of Creating Backup Copies

The system administrator needs to develop a backup strategy that will provide answers to the following questions:

- Which files will be copied and saved?
- Where will the backup copies be kept?
- What backup technique should be applied?
- How long does the backup cycle take?
- How long are the backup copies to be kept?
- How much space should be reserved for storing backup copies?

The set of files that needs to be saved varies depending on the purpose of the server. The file locations vary from server to server, but generally speaking it is essential to backup the files of individual users, files related to critical applications, system files and the entire content used in the everyday operation of the server. The user data is usually stored in the */home/* directory, the application files are stored in the */usr/* directory, while the configuration files of daemons responsible for the stable operation of the Linux operating system are stored at */etc/*. Each server may have different priorities when it comes to the choice of data to be backed up. For example, in the case of a web server, it is important to backup configuration files in the */etc/apache2/* or */etc/httpd/* directories, including the content offered to end users that is typically stored at */var/www/*. If the

Linux server is used to maintain an SQL database, it is necessary to backup both the data contained in the database and the configuration files of the SQL server. Nevertheless, from the security point of view it is important to backup files in the */etc/* directory since this directory also stores the configurations of the user accounts and network filters.

The backed-up data can be kept on the server itself and later transferred to another medium, such as optical storage. The problem with this way of storing backup copies lies in the inefficient resource management, the necessity of physical space where the discs will be stored and the delayed procedure of retrieving the backup data. A better way of storing backup copies is to transfer data via the network to one of the available storage servers. Data transfer via the network can be performed through the FTP protocol, with *rsync* tools or by adding a remote file system. The advantage lies in the fact that each of these methods offers the option of secure data transfer: SFTP, *rsync* via SSH or SSHFS for adding a remote file system. The system administrator should always bear in mind that one of the most important requirements of the data transfer is the confidentiality of data and the prevention of unauthorised access to the server on which the backup copies are stored.

When choosing a technique for creating backup copies, one should take into account the quantity of data that needs to be copied, as well as the space available for storage. The system administrator may choose a full backup or an incremental backup. The full backup involves copying complete files each time the backup is performed. This technique is convenient for simpler solutions that require the storage of a smaller quantity of data. The incremental backup involves only copying files that have been changed since the last backup. This solution requires an initial full backup and afterwards the incremental backup of files that have been changed. When a piece of information needs to be retrieved, the full backup represents the basis from which all subsequent incremental backups are applied until the last saved version of a file is found. This solution is convenient for backups of large quantities of data since it does not involve copying complete data but only the data that has been changed since the last backup, which reduces the use of the storage space to the minimum. Additionally, an incremental backup significantly reduces the time of the backup procedure, and if the files are kept on a remote server, it additionally reduces the load on the network link.

The backup frequency represents the period between two backup procedures and is adjusted depending on the frequency of change of data on the server. If the content is changed dynamically, the best practice is to apply daily backups or set an even shorter time period between backups. The shorter the period between the backups, the lower is the risk of data loss.

The backup cycle is the period between two full backups. The cycle begins with the first full backup; it includes all subsequent incremental backups and ends with the first subsequent full backup.

The time to keep the backup copies cannot be easily determined since it largely depends on patterns of usage and available resources. Backup copies should be kept for at least the minimum period required for a response in the case of unforeseen circumstances. If the system administrators have sufficient resources at their disposal, backup copies can be saved for a longer period of time (several backup cycles).

The space required for storage is directly related to the quantity of data backed up and the defined storage time. Backup copies can take up a lot of space on the server so administrators are advised to make a calculation based on the available space before implementing the backup solution.

8.2 Backup Tools

Depending on the chosen strategy, the system administrator should select appropriate tools for backup implementation. This section will present some of the most commonly used backup tools.

The ***tar*** command serves to archive data on the Linux operating system. It can be used for backups performed by archiving and compressing the desired files on the server. After archiving, the .tar file should

be transferred to an optical medium or remote server, depending on the selected storage space. If the transfer of the .tar file is performed via the network, administrators are recommended to use SFTP, SCP or SSH connections since they ensure the confidentiality and security of the transferred data. The ***tar*** command usually serves to implement the full backup solutions and is recommended in the cases when a small quantity of data needs to be backed up. Larger quantities of data would use more system and network resources.

The *Bacula* tool is very useful for incremental backup procedures since it enables differential backup. Differential backup involves copying all the files that have been changed since the last full backup. This means that the differential backup incorporates all the incremental backups performed to that point. An example of a backup cycle that relies on the *Bacula* tool is to have a full backup performed every first day of the month, to have incremental backups performed on a daily basis, and to have a differential backup performed on a weekly basis. The backup cycle starts with the full backup and includes all subsequent incremental and differential backups until the next full backup. The introduction of differential backup reduces the time required to retrieve data since the retrieval process uses the full backup copy and updates it with the latest differential copies, after which the latest incremental updates are applied. The disadvantage of differential backup is in that it requires more space for storing copies since it involves a larger number of files in comparison to incremental backup. The *Bacula* tool can be configured to use the SSH connection in order to ensure the transfer of the copied data via network. Administrators are advised to use SSH keys in order to secure the connection between the Linux server on which the backup is performed and the server on which the backup copies are stored.

Duplicity is another very good tool for incremental backup. This tool is based on creating encrypted .tar files that can be securely transferred to a remote server for data storage. *Duplicity* uses GPG (Gnu Privacy Guard) secure keys in order to have the .tar files digitally signed and encrypted and thus prevents the copied data being compromised. This tool supports SSH and SCP connections so administrators are advised to use them if the data is stored on remote servers.

Rsnapshot is another interesting tool for incremental backups. It is based on the *rsync* tool for file synchronisation. *Rsnapshot* puts a limit on the number of backup cycles, thus preventing an uncontrolled increase of the space required for data storage. The tool also enables the secure transfer of copies through the SSH protocol.

References

[1] Kickstart installation: https://github.com/rhinstaller/pykickstart/blob/master/docs/kickstart-docs.rst

[2] Preseed installation: http://www.debian.org/releases/stable/i386/apb.html.en

[3] SSHFS: https://help.ubuntu.com/community/SSHFS

[4] iptables: http://www.netfilter.org/projects/iptables/

[5] Antidote: http://antidote.sourceforge.net/

[6] Arptables: http://ebtables.sourceforge.net/misc/arptables-man.html

[7] Arpwatch: http://www.linuxcommand.org/man_pages/arpwatch8.html

[8] Fail2Ban: http://www.fail2ban.org/wiki/index.php/Main_Page

[9] DenyHosts: http://denyhosts.sourceforge.net/

[10] TCP Wrapper: https://www.centos.org/docs/4/html/rhel-rg-en-4/s1-tcpwrappers-access.html

[11] SE Linux: http://selinuxproject.org/page/Main_Page

[12] AppArmor: http://wiki.apparmor.net/index.php/Main_Page

[13] Sudo: http://www.sudo.ws/

[14] Syslog: http://man7.org/linux/man-pages/man3/syslog.3.html

[15] Syslog-ng: http://www.syslog-ng.org/

[16] Rsyslog: http://www.rsyslog.com/

[17] OpenVPN: http://openvpn.net/

[18] Glances: https://github.com/nicolargo/glances

[19] Iptraf: http://iptraf.seul.org/

[20] Nload: http://www.roland-riegel.de/nload/

[21] NetHogs: http://nethogs.sourceforge.net/

[22] Apachetop: http://www.fr3nd.net/projects/apache-top/

[23] GoAccess: http://goaccess.prosoftcorp.com/

[24] Wireshark: http://www.wireshark.org/

[25] Nmap: http://nmap.org/

[26] Bacula: http://blog.bacula.org/

[27] Duplicity: http://duplicity.nongnu.org/

[28] Rsnapshot: http://www.rsnapshot.org/

[29] Systemd-journald: https://wiki.archlinux.org/index.php/systemd#Journal

[30] Splunk: http://www.splunk.com/

[31] Fluent: http://www.fluentd.org/

[32] Logstash: http://logstash.net

[33] Graylog2: http://www.graylog2.org

[34] Apache Flume: http://flume.apache.org

[35] Logwatch: http://sourceforge.net/projects/logwatch/files/

[36] Logback" http://logback.qos.ch/

[37] Log4j: https://logging.apache.org/log4j/2.x/

[38] Adiscon LogAnalyzer: http://loganalyzer.adiscon.com/

[39] Tenshi: https://inversepath.com/tenshi.html

Glossary

ARP Address Resolution Protocol

DNS Domain Name System

NTP Network Time Protocol

CPU Central Processing Unit

IP Internet Protocol

GUI Graphical User Interface

HTTP Hypertext Transfer Protocol

MAC Media Access Control

NFS Network File System

RAM Random Access Memory

SQL Structured Query Language

SSH Secure Shell

TCP Transmission Control Protocol

TFTP Trivial File Transfer Protocol

UDP User Datagram Protocol