Degree Project

Creating a Secure Server Architecture and Policy for Linux-based Systems
Applied to Softwerk AB

Author: Marios Kourtesis
Supervisor: Dr. Rüdiger Lincke
Examiner: Dr. Mathias Hedenborg
Semester: HT2014
Subject: Computer-Science
Course code: 2DV00E
Abstract

Creating and maintaining servers for hosting services in a secure and reliable way is an important but complex and time-consuming task. Misconfiguration and lack of server maintenance can potentially make the system vulnerable. Hackers can exploit these vulnerabilities in order to penetrate into the system internals and cause damage. Having a standard architecture/configuration supporting the needed services saves time and resources while it reduces security risks. A server architecture protected by a security policy can secure the integrity and quality of the overall services. This research demonstrates building a secure server architecture protected by a security policy. To achieve this a security policy and a checklist was designed and combined with a host based IDPS, a NMS and a WAF.

Keywords: IDS, secure environments, Linux, network management systems, server security, web application firewall, server security.
Table of Contents

Acronyms.................................................................................................................................1

1 Introduction..............................................................................................................................2
  1.1 Previous research.................................................................................................................2
  1.2 Problem and Motivation......................................................................................................2
  1.3 Target group.......................................................................................................................3
  1.4 Goals and Criteria..............................................................................................................3
  1.5 Outline...............................................................................................................................4

2 Background.............................................................................................................................5
  2.1 Operating Systems..............................................................................................................5
  2.2 Virtualization Technologies...............................................................................................5
  2.3 Secure Environment...........................................................................................................5
  2.4 Firewalls............................................................................................................................5
  2.5 Intrusion Detection and Prevention Systems.....................................................................5
  2.6 Security Threat....................................................................................................................5

3 Work-plan and Method...........................................................................................................6
  3.1 Work-plan...........................................................................................................................6
  3.2 Research Expectations and Requirements.........................................................................6

4 Softwerk's Infrastructure.........................................................................................................7
  4.1 Overview...............................................................................................................................7
  4.2 Softwerk's Servers..............................................................................................................8
  4.3 Security Threats..................................................................................................................8
  4.4 Attackers.............................................................................................................................9
  4.5 Cyber Threats.....................................................................................................................10

5 Needs and Requirements.......................................................................................................12
  5.1 VPS Infrastructure requirements......................................................................................12
  5.2 Policy and Guidelines.......................................................................................................13

6 Research and Implementation...............................................................................................14
  6.1 VPS Operating System.......................................................................................................14
    6.1.1 OS Needs, Criteria.........................................................................................................14
    6.1.2 OS Current Situation....................................................................................................14
    6.1.3 OS Desired Situation.................................................................................................15
    6.1.4 OS Research...............................................................................................................15
    6.1.5 Architecture 32 vs 64 bit............................................................................................17
    6.1.6 Package Backporting.................................................................................................17
    6.1.7 OS popularity for servers.........................................................................................18
    6.1.8 OS Conclusion and Solution......................................................................................18
  6.2 Virtualization Solutions.......................................................................................................19
    6.3.1 Anatomy of a remote attack.......................................................................................20
    6.3.2 Detecting the Remote Attacking Surface..................................................................20
    6.3.3 Securing the Remote Attacking Surface...................................................................21
    6.3.4 SSH Hardening..........................................................................................................21
    6.3.5 Port Knocking............................................................................................................22
  6.4 Securing the VPS against Web-Based Attacks..................................................................22
    6.4.1 Anatomy of a Web Based Attack..............................................................................22
    6.4.2 Web Server Hardening...............................................................................................23
    6.4.3 Web Application Firewalls.......................................................................................24
    6.4.4 Protecting Web Applications with ModSecurity and OWASP CRS.......................24
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.4.5 Criticism to WAFs</td>
<td>25</td>
</tr>
<tr>
<td>6.5 Boosting Local Surface Security</td>
<td>25</td>
</tr>
<tr>
<td>6.6 User management</td>
<td>25</td>
</tr>
<tr>
<td>6.7 Network Management Systems</td>
<td>25</td>
</tr>
<tr>
<td>6.8 Host based intrusion detection systems</td>
<td>27</td>
</tr>
<tr>
<td>6.8.1 Integrating Local with Remote Security</td>
<td>28</td>
</tr>
<tr>
<td>6.9 Putting All Together</td>
<td>29</td>
</tr>
<tr>
<td>7 Policy and Guidelines</td>
<td>30</td>
</tr>
<tr>
<td>7.1 Overview</td>
<td>30</td>
</tr>
<tr>
<td>7.2 Purpose</td>
<td>30</td>
</tr>
<tr>
<td>7.3 Scope</td>
<td>30</td>
</tr>
<tr>
<td>7.4 Policy and Guidelines</td>
<td>30</td>
</tr>
<tr>
<td>7.5 Policy Compliance</td>
<td>31</td>
</tr>
<tr>
<td>7.5.1 Compliance Measurement</td>
<td>31</td>
</tr>
<tr>
<td>7.5.2 Exceptions</td>
<td>31</td>
</tr>
<tr>
<td>7.5.3 Non-Compliance</td>
<td>31</td>
</tr>
<tr>
<td>8 Case Study</td>
<td>32</td>
</tr>
<tr>
<td>9 Summary and Conclusions</td>
<td>35</td>
</tr>
<tr>
<td>9.1 Summary</td>
<td>35</td>
</tr>
<tr>
<td>9.2 Conclusions</td>
<td>35</td>
</tr>
<tr>
<td>10 Future Work</td>
<td>37</td>
</tr>
<tr>
<td>11 Bibliography</td>
<td>38</td>
</tr>
<tr>
<td>12 Appendix</td>
<td>41</td>
</tr>
<tr>
<td>12.1 Security checklist Template</td>
<td>41</td>
</tr>
</tbody>
</table>
**Acronyms**

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>WAF</td>
<td>Web Application Firewall</td>
</tr>
<tr>
<td>IDS</td>
<td>Intrusion Detection System</td>
</tr>
<tr>
<td>IPS</td>
<td>Intrusion Prevention System</td>
</tr>
<tr>
<td>IDPS</td>
<td>Intrusion Detection &amp; Prevention System</td>
</tr>
<tr>
<td>VPS</td>
<td>Virtual Private Server</td>
</tr>
<tr>
<td>SME</td>
<td>Small &amp; Medium Enterprise</td>
</tr>
<tr>
<td>VPN</td>
<td>Virtual Private Network</td>
</tr>
<tr>
<td>I/O</td>
<td>Input/Output</td>
</tr>
<tr>
<td>NMS</td>
<td>Network Management Systems</td>
</tr>
<tr>
<td>CIA</td>
<td>Confidentiality-Integrity-Availability</td>
</tr>
</tbody>
</table>
1 Introduction

The thesis has been written in cooperation with Softwerk AB, a small IT company (belonging to the category of small and medium size enterprises, short SME) in Växjö Sweden. Currently Softwerk has about ten employees working from office, home, and customer locations in and outside of Sweden. Most software systems and services Softwerk provides to their customers are server based, and often every customer gets an own virtual private server (VPS) installation. This allows to separate concerns, balance server load, increase security and distinguish different service levels and contractual periods, but it creates also a few problems [1].

1.1 Previous research

There are many studies and researches on server security. While many of them develop new security mechanisms and other provide guidance on how to secure a systems using certain technologies. The problem of server security is hard to solve while different requirements and system configurations among with human mistakes and unknown factors increase the complexity. We can see that none of the existing solutions can guarantee 100% security as security breaches in high security systems has happened many times in the past years.

There is research that focuses on securing Linux using Firewalls (iptables) and SELinux (Security-Enhanced Linux, which is a kernel module for enforcing access control policies), and monitoring the systems using simple tools such as ping [2].

Other researches focuses on in intrusion detection and prevention methodologies that are being used by modern security software and more efficient alarming [3][4].

Further research focuses on securing a virtualized environment from network based attacks and unauthorized access using Linux firewall [5].

In this project we suggest which existing solutions to use, package them together to an architecture and apply it to Softwerk servers.

1.2 Problem and Motivation

Almost every day new security vulnerabilities become public. In particular IT news and frequently provided security updates bear witness of this. Individuals, groups and organizations try to abuse this fact for their own purpose. Often automated attacks are used for identifying vulnerable systems reachable over the Internet. Therefore, it is not a surprise that even Softwerk servers have been under attack [6][7].

With a growing number of customers the number of server installations increases, currently about 25, as well as the diversity of the used virtualization technologies (currently three), OS versions (currently five different), and other services has massively increased over time. This does not include Softwerk’s dedicated server hosting services essential for back-office and development activities.

As a result of diversity and continuously discovered security vulnerabilities, it becomes increasingly difficult to:

(a) keep servers, services and data safe and secure,
(b) while still allowing the right employees and customers to access the servers and services from different places when needed,
(c) while supporting different technologies required by the hosted services, and
(d) while monitoring operation, detecting and solving disturbances.

In particular, when considering the limited resources available to a company of Softwerk’s size.

Therefore, the problem addressed by this thesis is the design and implementation of a
secure server architecture and policy for Linux-based servers.

The research question addressed in this thesis is:

*How can Softwerk deploy and administrate existing security tools to improve the security of their servers?*

The solution should allow creating and maintaining servers for new and existing customers and services in a secure and reliable way. Having a standard architecture/configuration supporting most of the needed services should reduce complexity, save time and resources, and reduce security risks even with limited staff and experience for setting up secure systems.

**1.3 Target group**

Since the software, configuration and services that the servers are providing can in general vary, the provided solution for the above problem is initially focusing on Softwerk's requirements. Yet, hoping that once the problem is solved for Softwerk, the solution and the methodology (architecture and policy) can be generalized to a security approach that can be customized to the needs of other SME's having a similar situation as Softwerk. Other SME's with similar requirements can also benefit from the practices that in this thesis project have been applied to Softwerk.

**1.4 Goals and Criteria**

To address the above problem, what is needed is to identify a suitable, i.e., meeting Softwerk's needs, and secure server architecture for hosting the services provided by Softwerk. Ensuring that the architecture is kept intact by defining a policy for maintaining it, since without maintenance Softwerk will get the existing problems sooner or later again.

The result should be a securely configured set of servers, requiring low costs for maintenance, being difficult to attack, easy to keep secure, up-to-date, and to monitor for operational disturbances, e.g., failures of services, running out of resources (memory and disk space), possible intrusion attempts.

To achieve this, the following goals should be reached:

1. Identify and document Softwerk's current operational requirements to make clear which services and servers exist and how (by whom from where) they are used. Further more, threats should be identified and described. This goal is reached when every company service is documented, including graphs which contain all the necessary information that are needed to move on as well as an overview of current threats is provided.

2. Based on the operational requirements and threats identified, a generic architecture is created for a secure server infrastructure fulfilling the operational requirements and being resistant to threats. The architecture should minimize diversity to maximize maintainability, i.e., minimize time and costs needed for maintaining and operating the required servers and services. For each of the architectures areas different alternative solutions should be compared choosing the most suitable solution according to Softwerk's needs. High diversity due to coexisting variations otherwise makes administration and operation difficult and reduces the security of the system (too many things to keep track of).

3. A policy and guidelines is needed, for administrating and maintaining such an architecture and the servers implemented according to it. This goal is reached if a document (incl. possible scripts) has been created describing how to administrate and maintain the identified parts of the architecture.
4. Migrate as many existing servers as possible to this architecture using the policy and guidelines. This goal is reached if at least one server have been migrated using the architecture in compliance with the policies. The server monitoring will be solved by using a NMS (network management system), the system security will be tuned up by using an IDPS (intrusion prevention and detection system) and a WAF (Web Application Firewall), and the systems will be protected from administration mistakes with a security policy and a checklist.
1.5 Outline

Section 2 provides a brief overview of a number of technologies playing a prominent role in the thesis. Section 3 describes the methodology and the work plan of the research. Section 4 describes the Softwerk network and security threats that can potentially arise. Section 5 lists the research needs and the requirements. Section 6 contains the research and implementation. Section 7 contains the security policy and directives. Section 8 presents a security checklist that has been applied to one of the Softwerk's servers. Section 9 contains the conclusion of the research. Finally, Section 10 discusses future work.
2 Background

This section provides a brief overview of a number of technologies that plays a prominent role in the thesis.

2.1 Operating Systems

Operating systems are a collection of software that are responsible for managing the hardware. There are many different type of OS (operating systems) for that are targeting a variety of devices. In this research the focus is on Linux based OS for servers. Linux is released in distributions, which are combinations of a Linux Kernel with a collection of softwares. Other Linux based OS such as BSD does not have any separation between the software and the kernel as Linux does [8].

2.2 Virtualization Technologies

Virtualization technologies are referring to systems that allows to run multiple OS inside an OS. For example, a server can have a host OS that emulates the execution of multiple guest OS. This gives the ability of environment separation, resource sharing, different OS execution, easy management, snap-shop management, portability and many more. Unfortunately virtualization technologies have a performance overhead which is expected but it is a fair exchange with its advantages [9].

2.3 Secure Environment

Secure environment is referring to a system that implements protections against any kind of disaster or attack. Secure environments are applying data protection methods, encrypted communication and many more security features in order to keep a system secure against any kind of security risk [10].

2.4 Firewalls

Firewall is a network security utility that is inspecting the network traffic. The security administrators can configure rules that decide which network traffic patterns should be blocked or allowed. There are different kinds of firewalls which have different purposes. For this research the interest is in network and web application firewall. WAFs are application firewall of the HTTP protocol. They can inspect and analyze the HTTP traffic [11].

2.5 Intrusion Detection and Prevention Systems

IDPS (intrusion detection and prevention systems) are analyzing and inspecting the protected system for anomalies and attacking patents. Their purpose is to detect, report and protect the system against potential attacks. They have the ability to diagnose attacks and automatically respond/block it before the administrator take action [12].

2.6 Security Threat

In the most abstract level, a security threat is a way or methodology that an attacker can follow to threaten the confidentiality, integrity or availability of a system. In other words it is every potential danger that can harm a system [13].
3 Work-plan and Method

This section summarizes the work-plan, requirements and implementations carried out in this thesis project.

3.1 Work-plan

By interviewing the Softwerk staff we concluded that in order to find a solution the following steps are needed:

a) Analyze the requirements by reviewing current installations.

b) Propose a reference/standard architecture and policy for Softwerk fulfilling these requirements.

c) Investigate different standards and reference policies, for example hosting services on Linux based servers.

d) Generalize this to find common solutions that fits to our companies resources.

e) Classify the existing services according to classification scheme for new services allowing to add them in a structured way to an existing network, reorganize the existing one using this methodology.

f) Finally, after validation the solution will be documented, optimized if possible and present it to the company.

3.2 Research Expectations and Requirements

In this thesis a literature review has been conducted where different architectures and policies are evaluated based on the requirements at Softwerk. These requirements are:

- Supporting PHP, Java, Tomcat and MySQL that is required by most applications/configurations.
- Automated security updates and system updates
- Monitoring of services, resources, failures, required updates, notifications to administrator, etc.
- Hardened installations (make it difficult for attackers to get in, make it easy for administrators to identify a breach, to stop it, and to analyze what happened)
- Same conventions/configuration on all servers
- Apply secure passwords and access patterns, certificates and etc.
- Create a VPS policy and guidelines for the servers maintenance, administration and security.

Additionally, we have taken the following aspects into account when evaluating architectures and policies:

- Create and configure a new server.
- Update new server.
- Setup firewalls.
- Setup intrusion detection.
- Setup Antivirus.
• Users and logins.
• Monitoring of services and log files.

As a short overview, Softwerk requirements might include:

4 Softwerk's Infrastructure

This section provides an overview of Softwerk infrastructure, servers and services as well as security threats.

4.1 Overview

Softwerk has their own offices, intranet, servers, printers and much general purpose networking equipment. Some of the servers are hosted in the intranet but the majority of them are hosted by third party services. Figure 4.1 shows an overview of the current infrastructure, yet not all details are shown. Other SME architecture's can be similar.

The employees are not limited to working only from the company's offices, it happens quite often that they work from home and other places. Currently, many core services are exposed in the Internet and accessible after authentication (password for web-based service or key-based for SSH), some, e.g., printers are available after connecting to the intranet over VPN (a technology that extends the intranet over the Internet) where they can access all of the company's resources, even if they work at home. Additionally, remote employees can use Internet telecommunication services such as Skype and Google Hangouts to join the internal meetings.
Figure 4.1 shows the intranet and extranet of Softwerk. As shown in the figure Remote Workers are connected using VPN to the intranet from various unknown networks. From there they can access all the intranet resources. The intranet consists of the computers, mobile phones, printers, servers and networking devices (such as switches, firewalls and routers) and it is connected to the Internet with the connection marked as red. The extranet consists a number of VPS (virtual private servers) hosted at Glesys, a VPS hosting service in Sweden, and many multi-purpose third party services that are used by the company (such as Google drive).

At Softwerk, all employees are considered as a single group having the rights and privileges to access all parts of the infrastructure from a security point of view. They should be able to perform their work from the office or from outside the office.

4.2 Softwerk’s Servers

The majority of the servers used by Softwerk in their projects are Linux servers that are running as VPS (using different virtualization technologies) at a third party provider (Glesys). The number of servers Softwerk has to maintain is steadily increasing (currently about 25), as well as the associated maintenance effort and costs. There are many reasons for companies applying the principle of different service to each server [14].

The main reasons are listed here:

- It is really common for customers to ask for access to the server their service(s) are running on. This can potentially have critical security issues for services of other customers that could be hosted on the same server. It is often very important for customers to have privacy and confidentiality. It is important to point out that every software installed on a server may affect the security in a negative way. By installing a remote service, the remote attacking surface of the server is expanded. Additionally by installing a local software, the local attacking surface is expanded as well.
In many cases customers can make a change to a server which can lead to a denial of service to another service which is hosted at the same server. Denial of Service (often referred as DoS) means that the system is unavailable to its intended users. Often DoS is describing attacks that are trying to break down the availability of the targeted system.

Quite often multiple services can have requirements that are conflicting with each other (for example different versions of the same library).

A service can request resources from the system causing starvation problems to another service or services running on the same system. For example, a starvation problem can be that one service uses all the available memory of the system.

Customers often expect Softwerk to operate, support and maintain the installations according to Service Level Agreements in order to provide required services in a secure and reliable way.

Some of the services provided by Softwerk can require root access. A security hole in one of these services can have critical security, privacy and confidentiality issues to other services running on the same system.

By providing a different VPS to each customer or service is eliminating all of these problems. Even if one should have a problem, services on other servers should not be affected. The respect of the privacy and confidentiality of Softwerk's clients is the top priority.

4.3 Security Threats
Discussing security threats and types of attackers will help to understand potential attacking reasons and impact that attacker actions can cause.

A system is considered secure when Confidentiality-Integrity-Availability (CIA) is applied. The black set of information shown in the following figure is considered secure [15].

![Figure 4.2: Confidentiality-Integrity-Availability [16]](image)

Typically, attacks are attempting to break at least one of the following:

1. Confidentiality
Confidentiality defines that only authorized users can access digital information. It can be also referred as secrecy or privacy.

2. Integrity

- Integrity defines that the digital information can be modified or created only by authorized users.

3. Availability

- Availability defines that the digital information be accessible for the authorized users.

In order to secure a system it is important to know potential enemies and threats that can attempt to break the CIA of the system's provided services [16].

4.4 Attackers

The attackers (individuals or groups) can be classified to the following three (3) categories according to their skills and motivation.

Skills can range from novices with limited security relevant knowledge, e.g., they know how to use automated tools and only cause damage to the targeted systems, to professional attackers with deep security relevant knowledge that, e.g., are able to developed new attacking techniques and tools and they do more sophisticated attacks.

Motivations can typically be [17]:

- Financial – make profit by selling information about security holes, sell stolen data, or blackmail victims, etc.
- Self esteem – get recognition among a group.
- Public recognition – get recognition in public.
- Ideological – such as political and religion related reasons, e.g., to make statements, support various groups or individuals, etc.

Typical examples of attackers are presented in the following subsections.

4.4.1 Script Kiddies

This kind of attackers does not have deep knowledge in computer security. Usually they are advance computer users which are using automated tools and exploits for attacking their targets. Often they are causing troubles and damage to computation systems just for their self esteem without having any financial benefit. They are dangerous because they can cause a lot of problems just for their own satisfaction [17].

4.4.2 Criminals

This kind of attackers are attacking enterprises and users in order to steal money. They achieve this by invading into enterprises networks and steal valuable information in order to sell them in black markets or blackmail the enterprise. Quite often they work together in cyber-gangs. They usually have the same skills as the Script Kiddies but they are using their knowledge for financial crimes.

4.4.3 Black Hat hackers

This title describes hackers that are behind sophisticated crime. They are computer experts with deep knowledge in operating systems, networks and security. They are involved in big financial crimes and are controlled by money. They develop their own attacks and exploits without making them public [17].
4.5 Cyber Threats
A company and its servers can be targeted by many factors that can disturb their quality of services and moreover the operation of company itself.

4.5.1 Hactivism
The word Hactivism is an acronym of Hack and Activism. Hactivism is the phenomenon of massive web based attacks to IT infrastructures and resources. The reason of these attacks are usually political, social or religion related. It is a revolution from the Internet. The attackers are using well known attacking techniques and tools. They are using DDoS (distributed denial of service) attacks or the change the content of the targeted web pages with their own message without permissions (Defacement) [17][18].

4.5.2 Botnets
Botnet is called a collection of infected computers that are controlled by an attacker. The users of the infected computers does not know that their computer is part of a Botnet because the malwares are not leaving any visible traces to the end user. It is common that the infected computers participate in massive cyber attacks. Each of the infected computers (Botnet nodes) are scanning the Internet for vulnerable computers to infect and add them to the Botnet network. All of the infected computers are managed by a central management station. The most common attacks that the Botnets are doing is DDoS. It is really difficult to break down a Botnet because they are communicating using IRC channels, HTTPS requests and the TOR network. In the black market you can rent a whole Botnet for a low cost [19].

4.5.3 0-day Exploits
Zero day exploit stands for a security whole to a software which is unknown to the public and the manufacturers. Hackers are selling these exploits through black markets such as “http://1337day.com/”. Other hackers are buying them and launch attacks to the vulnerable systems for their own benefits. Their benefits are usually financial. Zero day exploits are a threat that can target all size of enterprises [20].

4.5.4 Corrupted Employees
Companies former or dissatisfied employees can do a lot of insider attacks. The reasons for an attack like this can vary.

The most common reasons are:

- a) A distrusted employee which is seeking for revenge.
- b) An employee that knows companies sensitive information can defraud the company for his financial benefit.
- c) Competitive companies can bribe employees for leaking company's critical data.
- d) A whistleblower (whistle-blower or whistle blower) is a person who exposes misconduct, alleged dishonest or illegal activity occurring in an organization. The alleged misconduct may be classified in many ways; for example, a violation of a law, rule, regulation and/or a direct threat to public interest, such as fraud, health and safety violations, and corruption. Whistleblowers may make their allegations internally (for example, to other people within the accused organization) or externally (to regulators, law enforcement agencies, to the media or to groups concerned with the issues) [21].

4.5.5 Trusted Third Party Companies & Services
Trusted Third Party services usually store sensitive information about enterprises. The
risk of a potential denial of service or sensitive information disclosure is relatively high. Attackers can cause serious issues by damaging third party services. Additionally, third party services can be bribed to disclosure critical information.

4.5.6 Industrial Espionage
Competitive companies can bribe companies employees or ex-employees for revealing sensitive information which they can use for causing damage to their infrastructure. Additionally they can try to sabotage targeting companies resources by paying hackers to launch attacks to the targeted enterprise. Also business secrets can be used for financial gain [22].

4.5.7 Employees & System Administration Mistakes
Employees without specialization to security is really common to leave backdoors without realizing it. These kind of backdoors can be placed by system misconfigurations or developing critical parts of Information Systems without using secure development principles and security consulting [17][21][22][23].
5 Needs and Requirements

The needs and requirements are defined for Softwerk's network infrastructure, policy and guidelines. An initial analysis of Softwerk's current installations was performed and derived the requirements listed in this section. The analysis was based on investigation of all the existing current server requirements and moreover new requirements that Softwerk might have in the near future.

5.1 VPS Infrastructure requirements

Softwerk has in general the following needs to support customer projects. The needs can vary from client to client but it is in general a combination of them based on the technologies selected for the solution.

Softwerk Needs:

- Support for multiple programming languages, database systems and web server technologies that are required by the applications and information systems. The installation should support at least the technologies listed in Table 5.1 for flexibility and also support the technology stack Softwerk is used to.

<table>
<thead>
<tr>
<th>Programming Languages</th>
<th>Java, Ruby, PHP, Python</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web/Application Servers</td>
<td>Apache, Nginx, Tomcat, Glassfish</td>
</tr>
<tr>
<td>Database Systems</td>
<td>MySQL, MariaDB, MongoDB</td>
</tr>
</tbody>
</table>

Table 5.1: Supported Technologies Requirements

- Automated security and system updates to reduce administration and keep systems safe and up-to-date.
- Support for monitoring and logging the VPS resources listed in Table 5.2 to identify problems as early as possible and help in debugging and maintenance.

<table>
<thead>
<tr>
<th>OS resources</th>
<th>Running Services, Available updates</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS issues</td>
<td>Failures, Errors, crashes</td>
</tr>
<tr>
<td>System resources</td>
<td>CPU, Memory, Hard Disk, Network I/O, HTTP Protocol</td>
</tr>
</tbody>
</table>

Table 5.2: Monitoring Resources Requirements

- A secure VPS architecture, where attack attempts should be detected, prevented, logged and reported in real time. It should provide a secure configuration of the Web and SSH Server.
- All the VPS should have the same configurations, administration guidelines and security policy to reduce administration and setup costs while avoiding security issues due to inappropriate or incomplete configurations.
- Secure access to the VPS allowing only authorized users to access the servers from work, home or other locations. But at the same time preventing unauthorized users from gaining access.
- Guidelines that should provide administration and maintenance documentation.
- A VPS security policy for the VPS maintenance, administration and security.
5.2 Policy and Guidelines

A policy that describe best practices, clarify principles, resolve conflicts, and achieve the goal of creating a secure and maintainable VPS life-cycle. It should explain and describe standards that all employees should follow, while guidelines are meant to outline best practices for following in VPS administration [24][25].

The policy should specify the standards related to Softwerk's VPS secure life cycle, maintenance and administration. Specifically it should cover the topics described in the table 5.3.

<table>
<thead>
<tr>
<th>VPS Administration</th>
<th>Creating, Updating, Configuring, Maintain</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPS Security</td>
<td>Firewalls, IDPS, User Management, Monitor</td>
</tr>
</tbody>
</table>

Table 5.3: Policy Specifications

Note to the Reader: Specific configuration details, program versions, IP addresses, user-names, passwords, have been omitted from the public part of the thesis. There is a complementing Appendix with the actual details.
6 Research and Implementation

Implementing a VPS architecture requires analyzing the requirements one by one and potentially split them into smaller requirements. Each requirement constitutes an independent problem. Merging the problem's solutions leads to the desired solution.

This academic research solves the problem in theory using abstraction level where multiple theoretical solutions and architectures can be applied. The most appropriate solution for Softwerk will be applied to their infrastructure and described in more details.

6.1 VPS Operating System

The Operating System is the core of the architecture and it should fulfill the desired requirements. The more transparent the architecture is, the more flexible it is for future upgrades and reconsiderations (the term transparent is used to describe the architecture tolerance to core modifications). The implementation should be able to be applied to many Linux OS distributions.

6.1.1 OS Needs, Criteria

The company OS needs are the following:

- Stable
- Secure
- Long Lifetime Support
- Upgrade Support
- Big list of packages
- Support for the latest Software/packages
- Easily configurable
- Low Cost

6.1.2 OS Current Situation

The current situation of the Softwerk servers is shown in table 6.1.
<table>
<thead>
<tr>
<th>Server Name</th>
<th>Operating System</th>
</tr>
</thead>
<tbody>
<tr>
<td>apea.softwerk.se</td>
<td>Ubuntu 12.04.4 LTS</td>
</tr>
<tr>
<td>appcms.softwerk.se</td>
<td>Ubuntu 12.04.4 LTS</td>
</tr>
<tr>
<td>crinefield.com</td>
<td>Ubuntu 10.04.4 LTS</td>
</tr>
<tr>
<td>daypay.se</td>
<td>Ubuntu 12.04.4 LTS</td>
</tr>
<tr>
<td>demo.arisa.se</td>
<td>Ubuntu 12.04.4 LTS</td>
</tr>
<tr>
<td>demo.ticknik.se</td>
<td>Ubuntu 12.04.4 LTS</td>
</tr>
<tr>
<td>dev.softwerk.se</td>
<td>Ubuntu 12.04.4 LTS</td>
</tr>
<tr>
<td>flow.softwerk.se</td>
<td>Ubuntu 12.04.4 LTS</td>
</tr>
<tr>
<td>iec.softwerk.se</td>
<td>Ubuntu 12.04.4 LTS</td>
</tr>
<tr>
<td>iqm.arisa.se</td>
<td>Ubuntu 12.04.4 LTS</td>
</tr>
<tr>
<td>quickvotes.com</td>
<td>Ubuntu 10.04.1 LTS</td>
</tr>
<tr>
<td>staging.flow.softwerk.se</td>
<td>Ubuntu 10.04.1 LTS</td>
</tr>
<tr>
<td>test.arisa.se</td>
<td>Ubuntu 12.04.4 LTS</td>
</tr>
</tbody>
</table>

Table 6.1: Sofwerk's servers OS

It is clear that the current Server Operating system that Sofwerk is using is Ubuntu and that the 10.04 version is soon running out of support (5 years for LTS version).

### 6.1.3 OS Desired Situation

The OS administration should be an easy procedure for Softwerk's employees. In many cases, SME do not have dedicated human resources to handle this tasks. Basic system administration should be an easy process that employees with a basic knowledge of operating systems can handle.

### 6.1.4 OS Research

It is important to note that there is a large number of Linux based OS. For this reason, a full research that cover all of the these systems is not possible to carry out within the time frame of this project. This research has been done by checking a list with the popular Linux OS and compare them based on the Sofwerk needs. These systems were by popularity chosen from Distrowatch, which is a website that enumerating and keeping track of the biggest list with Linux based OS [26].

The testing installations had been done using VMware's Vmplayer version 6.0.3. This software allows installing an OS virtually to your a computer. The term virtually means that an operating system is running on the host OS. In this way you can install one or more OS on a single computer and run them without making any permanent change to your host OS [27].

The tests were implemented by doing simple Operating System's administration tasks. The tasks that were performed to the OS are the following:

- OS update
- OS upgrade
- OS package management (Installation, removal of new packages and etc.)
- Daemon management (Service management)
- Scheduled Tasks (Also referred to as Cron tasks)
- User Management
- Firewall configurations

For this test, the following Linux Operating Systems were tested (see also Table 6.2):
- CentOS, a community Enterprise Linux which is a totally compatible rebuild of RHEL [26].
- Ubuntu LTS, a free Debian based OS which is operated by Canonical [26].
- Debian, a free community build OS that follows the GNU (free software) philosophy [26].
- OpenSuse, a community supported OS based on SUSE which is sponsored by Novel [26].
- Fedora, a RHEL based OS that is bringing the latest technologies, sponsored by REDHAT [26].
- Arch, an independent Linux which is targeting simplicity and minimalism.
- Suse, an enterprise Linux for mission-critical computing with integrated support for Xen virtualization [26].
- RHEL, an enterprise Linux that supported multiple architectures with certification trainings and commercial support [26].

The following table presents some key points of the Linux distribution that are included in the main criteria. These criteria are:

- The support period in years after the release date of the operating system. The shorter the period, the more often one has to do a distribution upgrade or completely reinstall the server causing maintenance costs and down-time of services.
- If there is any commercial support available by companies. This could be helpful in case there are any specific problems otherwise not solvable. It could help to reduce costs or staying with a certain configuration.
- How often a new release is published usually affects how recent the kernel and available software packages are. Even though it is often possible to install any software from scratch, installing software using the package managers has several advantages, e.g., easy installation, updates and security updates, as well as compatibility and dependency management.
- If the operating system contains the latest packages available. Similar reasons as previous point.
- If the operating system is distributed for free, that is if there are any license or other costs which might need to be charged to clients. This usually reduces the costs of a solution and here Linux has here a clear advantage over Windows or MAC as Server OS.
<table>
<thead>
<tr>
<th>Distribution</th>
<th>Support</th>
<th>Commercial Support</th>
<th>New Release</th>
<th>Latest Packages</th>
<th>Free</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ubuntu LTS</td>
<td>5 years</td>
<td>Yes</td>
<td>2 years</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Suse Linux Enterprise Server</td>
<td>up to 11-13 years with extended Support</td>
<td>Yes</td>
<td>3-4 years</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>OpenSuse</td>
<td>Lifetime of 2 releases + 2 months overlap. With a release cycle of 8 months this makes it 18 months</td>
<td>Yes</td>
<td>About 8 months</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Debian</td>
<td>3 years</td>
<td>No</td>
<td>About 2 years</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>CentOS</td>
<td>10 years</td>
<td>Yes</td>
<td>Depends by Red Hat</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Fedora</td>
<td>13 months</td>
<td>No</td>
<td>6 months</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Red Hat Enterprise Linux</td>
<td>10 years</td>
<td>Yes</td>
<td>About 18 to 36 months</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Arch</td>
<td>Rolling Release</td>
<td>No</td>
<td>*Rolling Release</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 6.2: Linux Distribution Table

Note: The methodology that was used to produce the table 6.2, can not be considered totally accurate. Many of the OS are open-source and they do not have strict release dates. Some of the results were calculated according to the previous years release dates. Although, this can give a good estimate of how often a new version is released.

*Rolling Release*, is the continuously process of developing a system. In the case of OS there is not any defined release version. The installation process is following this procedure: a user is downloading a base version of the OS and he is updating his system to the latest version. This process is repeated all the time which makes this type of OS a bad choice for production servers.

6.1.5 Architecture 32 vs 64 bit
The architecture of the OS can play a significant role for the performance of the server. There are some gains and some losses in both options. This is out of scope of this project.
In general it is better to use 64 bit OS if the server CPU support it. This will make the system perform faster in most cases but is more memory consuming [28][29].

6.1.6 Package Backporting
Enterprise class Linux such as RedHat and Suse are not using the latest packages for stability and compatibility reason. They have a different approach to solve security updates and bugs. When a new security issue is discovered in a package, they are patching/backporting the old package without upgrading to the latest version [30].
RedHat is defining backporting as:

“We use the term backporting to describe when we take a fix for a security flaw out of the most recent version of an upstream software package and apply that fix to an older version of the package we distribute.” [30]

It is important to understand this because it can easily lead to misunderstanding. Often people believe that their system is vulnerable due to it is not running the latest version of the software as the security advisors and the software vendors are recommending. Additionally many security assessment tools are using the version number as the main criteria to detect vulnerable software, which is raising a lot false positive alarms.

The main reason for backporting is to boost the stability of the system. Updating the software to the latest versions usually introduce new features, bugs and modifications in the functionality that can have crucial effects to production systems.

6.1.7 OS popularity for servers

Figure 6.1 shows the server market share of OS. It is a fact that the majority of OS being used are Linux Based. It is important to note that RedHat based OS (Fedora, CentOS, RedHat) are having the biggest market share of Linux OS in this graph.

![Figure 6.1: Web Operating System Market Share 2012](image)

6.1.8 OS Conclusion and Solution

It is really difficult to measure stability because of:

1. All of the distributions are referring that they are stable.
2. The test requires that the systems runs for a long time.
3. During these test updates and fixes are released.
4. The errors that are happening is really difficult to reproduce.
5. Problems affect specific hardware.

For the above listed reasons, in a theoretical approach stability will be measured according to the package policy of each distribution. It is a fact that broken packages are responsible for many stability issues in OS. Packages which have been tested for years can boost up the stability of a system.

These tests showed that all of the tested OS can handle these tasks the equally efficient. Each distribution have small differences such as the package management utilities, but the functionality stays the same. In other words, all of the tested OS are doing exactly the same tasks by using different commands and by following similar logic.
The OS can be categorized to:

- **Cutting Edge**: where the latest features and packages are included, sacrificing stability.
- **Enterprise Class**: where the features and the packages have been tested for years sacrificing latest software.

Concluding, no matter what Linux based OS is being used the functionality stays mainly the same. When stability and security it is the main priority is highly recommended to use an Enterprise Class Linux such as CentOS. In the other hand when the latest versions of libraries and software are required, the best choice is to use one of the distributions that have marked the “Latest Packages” column to “Yes” in Table 6.2. In either cases a rolling release OS such as Arch should be avoided for production servers.

For Softwerk's network it is highly recommended to use Ubuntu LTS when latest packages is the main criteria, so they will not have any new costs for porting the servers to another Linux distribution. Additionally they will keep the same homogenia among the other servers. It is preferred if the servers they need to secure would be ported to an enterprise class Linux like CentOS.

From this point, the implementation is referring to CentOS as the core OS of the architecture. All of the configurations provided been tested on CentOS. (Note: Other Linux distributions can potentially require slightly different configuration).

### 6.2 Virtualization Solutions

There are multiple virtualization implementations available and each of them has its own advantages and disadvantages. They can play a significant role to the performance and the security of the server.

This research does not include a virtualization comparison. Although the proposed architecture should work regardless of the underlying virtualization technology. The most common advantages of virtualization technologies are:

1. Fast server deployment.
2. Server Cloning.
4. The deployment is in a controlled environment.
5. Hardware resource sharing and management.
6. Better for deploying testing applications, where they can trigger a crash to the virtualized OS while not affecting the main OS.
7. Server portability and easier re-deployment.

For the above reasons it is highly recommended to deploy the servers in a virtual environment.

### 6.3 Securing the VPS

Securing a server can be a complicated procedure. It requires to detect all of the weak points where an attacker can potentially attack and secure them. After everything is secured a penetration testing can be used. The penetration-tester is evaluating that everything is configured properly and that the security measures, configurations and functionalities does not introduce new vulnerabilities. The evaluation is done by attacking to the systems as hackers do.

The attacking surface of an OS can be categorize to remote and local. The remote attacking surface includes the set of ports/services that are accessible from outside the
network. An attacker can attack these points without having local access to the server; for this reason vulnerabilities in remote services can have critical impact. On the other side, the local attacking surface points consists of the set of OS functionalities, files and applications that the user can access. It does not have as severe impact as the remote attacking surface, but it can play a significant role to the overall security of the server. In many cases, an attacker gains access by exploiting a remote vulnerability, with limited user privileges. At this point the attacker needs to attack the local attacking surface. For example, running a kernel exploit to gain root access. For this reason the local attacking layer can have play serious role in the OS security. An attacker who gained access to a target system can potentially have limited access that does not allow to do any potential damage.

6.3.1 Anatomy of a remote attack

The first step for securing a system is to understand how it is possible to attack it. Without knowing how an attacker is working, it is simply not possible to secure the system.

A generic approach of remote attack has the following two main phases:

1. Information Gathering stage: the attacker is trying to find as much as possible information about the targeting system. By identifying and fingerprinting the OS, running services and the versions. Knowing all of the above information the attacker is ready to move to the next step.

2. Exploitation stage: the attacker is attacking by launching public exploits to potentially outdated or vulnerable services. In many cases the attacker is launching private exploits (0day, not know to the public) he has developed on his own or bought from black markets.

6.3.2 Detecting the Remote Attacking Surface

In order to detect the remote weak points of a Linux OS, it is needed to find the services which can accessed remotely.

This can be done easily by using the `netstat` utility.

```
netstat -lnptu
```

Executing the above command, will return all the listening ports (UDP & TCP) and the services which are using the ports. At this point the remote attacking scope for an attacker is known.

**Example**

The following table contains a truncated output of the `netstat` command. It shows that there are two services listening to all IP addresses (0.0.0.0 represents all IP addresses) under TCP protocol.

```
tcp 0.0.0.0:3306 LISTEN
tcp 0.0.0.0:80 LISTEN
```

In order to know if the services are really accessible from the public, all of the Firewalls that are placed to protect the services must be checked. This includes the internal (Linux Iptables) and external Firewalls that potentially are enabled or installed on the network accordingly.

Another way to enumerate the listening services is to use a port scanner. A commonly used is Nmap which is also often considered the best. It supports multiple types of scanning. It can detect if a port is filtered by a firewall or not. Nmap allows the testers to evaluate if the system correctly configured [32]. In many cases Firewalls can detect and
stop a port scan so it may not be totally accurate.

Note: Port scanning is not considered legal. You must ensure that you have permission before proceeding to these actions. Unauthorized scanning can lead to legal issues [33].

6.3.3 Securing the Remote Attacking Surface

Mitigating the remote weak points requires a lot of effort. Usually different services require different hardening approaches.

The following general directives should be applied if possible to tune up the security. Using these configuration will not prevent all the attacks but they can restrict the exposure and the threat.

- Each remote service should be run as each own user with as limited as required permissions.
- The remote accessible services should be as limited as possible. Services that are not needed should be disabled.
- Remote administration services should be accessible only by granted employees.
- Remote administration services that are used by the granted employees should only be accessible from the company's intranet.
- Remote services should not listen to default ports. Note that this is not feasible in all cases. A web server for example should listen to port 80.
- Remote services that are using unencrypted communication must be avoided.
- Remove fingerprints of remote services.

Daemons such as SSH and web servers have wider attacking surface. For this reason they require specific configurations. The configurations for each of these dedicated services is described in separate sections below.

6.3.4 SSH Hardening

Remote administration services such as SSH can have a very critical impact in case of high-jacking. The attackers know that every VPS has an administration service and this knowledges makes them being one step ahead of administrators. A misconfiguration can be critical.

Hardening configurations [34]:

- Disallow root login.
- Disallow login using username and password.
- Disable Empty Passwords.
- Change default port.
- Allow only login using private keys.
- Disable all functionalities that are not needed or being used.
- Make the SSH accessible only by intranet's IP address range.
- Disable weak ciphers in order to be protected by drop down attacks.
- Remove SSH fingerprints.
- Disable legacy version 1 of SSH.
- Configure Idle log-out time.
• Add a banner with a warning message, to inform the user that the system is being monitored and unauthorized will be prosecuted by law.
• Limit SSH Listening interfaces to one.

6.3.5 Port Knocking

By makings use of Port Knocking technique, system/security administrators can make remote administration services invisible to attackers. This method can efficiently defend against port scanning, network and application discovery attacks [35].

The whole idea behind the port knocking is that the firewall is making the protected service unavailable by blocking all of the network traffic to it's port. This will have the outcome that nobody can open a connection to the service. In order for the administrator to be able to connect to the service, he/she needs to do an action that will trigger the firewall to drop the rule which is blocking the protected port. Usually this action is to send a combination of sequential connection packets (SYN TCP packets) to a secret combination of ports. Once the protected port is unblocked, the administrator has a limited time to access the protected service before it gets blocked by firewall again.

Port knocking can be implemented by using pure iptables (the Linux kernel firewall) or by using a daemon designed for this task.

Example:

Considering that the protected service is SSH and is listening to port 22. Then set a secret port combination that only the administrator knows of. An example is the port 1000, 2000 and 3000. The administrator can access the SSH by doing the following:

```
cnc hostname 1000
nc hostname 2000
nc hostname 3000
ssh user@hostname 22
```

In the above example the netcat (nc) utility is being used to access the secret port combination. An attacker can not access the protected service, because he does not know the secret combination of ports.

6.4 Securing the VPS against Web-Based Attacks

Web based attacks is the biggest threat to the enterprises and users. It is really difficult to protect against this because there are many attacking surfaces in different aspects. Hackers can attack the users (normal users and administrators) of a web application and the server/s. Many people are wondering why are there that many attack vectors in web applications. The answer is simple that the web nowadays is doing something totally different from what it was designed to do.

OWASP (Open Web Application Security Project) is an open organization that is specialized in web security. Its task is to make the web safer by proving educational material and tools that people can make use of. The materials are free and licensed under open software. Their community constitute many security engineers, researchers companies and enthusiasts [36].

For this thesis many OWASP materials have been used.

6.4.1 Anatomy of a Web Based Attack

A web attack can have multiple and complex scenarios. Some simplified examples are presented and explained.

Scenario 1
Figure 6.2 shows an attacker who is attacking to a website and is infecting the web server or a part of the website with a malware (it can be a malicious JavaScript). Users that are connecting to the website are downloading the malware by thinking that it is some kind of legitimate file. In many cases they can get infected without downloading any file. An infected web server can potentially attack many outdated web-browsers or plug-ins in order to gain access to the victim's computers.

![Figure 6.2: Scenario 1](image)

**Scenario 2**

Figure 6.3 show an attacker that knows or has found an SQL injection vulnerability to the target server which allows to interact with the server database. The impact of this issue can allow the attacker to insert, delete or modify database entries. In many cases injecting SQL queries that are calling function of the DBMS back-end can provide access to the file system of the server. By exploiting this fact an attacker can upload a backdoor to the server.

![Figure 6.3: Scenario 2](image)

There are many attacking scenarios that an attacker can exploit in order to achieve his goals. These examples introduce how an attack can look like. Real world attacking scenarios can be more advances and sophisticated [37].

**6.4.2 Web Server Hardening**

Web servers misconfiguration can result in many security issues. There are a few principles and directives that can be applied to secure them. Understanding that each web-server and technology has different configuration option and customizations, this re-
search is targeting general guidelines and not vendor specific. The following is a non-exhaustive list of directives that should be applied, if possible [38]:

- Disable directory listing.
- Run the web server with its own restricted access user and group.
- Remove write permission from the web-root if possible.
- Uploaded files should be uploaded inside the database if possible. If not, they should be uploaded to a directory outside of the web-root that does not have execution permissions.
- Hide web-server version and signatures.
- Disable Database and Server errors.
- Update the server in a regular basis.
- Disable unnecessary server modules.
- Limit the Request size.
- Enable logging.
- Add Denial of service protections by setting connection TimeOut and by limiting the maximum allowed concurrent connections.

Web server hardening can play a significant role in the security of the server. A proper configuration can reduce the threat or the exposure of an attack.

6.4.3 Web Application Firewalls

WAFs are firewalls on the HTTP application layer of the OSI model (Open Systems Interconnection). Their functionality is limited in HTTP packet inspection and manipulation and they can handle both requests and responses on the HTTP application layer. By inspecting the input and output HTTP(S) communication channels can secure the data, servers and applications from web attacks. Properly configured WAFs can successfully mitigate web application vulnerabilities and block attacks before they reach the application [39].

6.4.4 Protecting Web Applications with ModSecurity and OWASP CRS

ModSecurity is a free and open source WAF module developed by Trustwave SpiderLabs. It supports multiple web server platforms (WEB Servers) such as Nginx, Apache and Microsoft ISS. Additionally the unsupported platforms can be protected by combining ModSecurity with Apache. To achieve this, first the ModSecurity module should be configured in Apache and then, the unsupported platform should configure the Apache as a Proxy Gateway [40].

ModSecurity by default does not contain any WAF rules, in other words this means that it does not offer any protection by default. Trustwave is offering a WAF rule set as a non-free product. The security administrator needs to write rules in order to protect their systems if the commercial rule set is not a desired choice. Fortunately there is an OWASP project named OWASP ModSecurity CRS (Core-Rule-Set) which is a collection of generic WAF rules than can then defend the web application against many web attacks. In most times these WAF rules need to be adjusted to the web application use cases and policy [41].

ModSecurity with OWASP Mod-security CRS project can offer a good layer of protection.
6.4.5 Criticism to WAFs
WAFs by themselves does not offer an ultimate protection as the WAF companies and providers promise. Sandro Gauci, a security researcher developed a tool named Wafw00f which can detect WAFs which is included in Kali (the most well recognized Linux distribution for penetration testing). Attackers can detect WAFs by using advanced mapping tools and potentially bypass them. In many cases, WAFs can increase the attacking surface. Additionally there are many techniques that can be potentially bypass WAFs [42] [43].

WAFs should not be a replacement of secure development principles but an extra shield of protection.

6.5 Boosting Local Surface Security
The local layer of security includes the parts of an OS that can be accessed only by users who have access to the system. These users can be normal, masqueraded or attackers. For this reason, the system administrators should maintain an access control list of the system and the resources. Their access should be as limited as their assigned tasks requires. By providing more access and privileges to users than they require can potentially have security impacts.

Detecting and preventing potential attacks require to monitor as much information as possible of the OS flow. Each user activity should me logged and analyzed by the security administrator. In the case of detecting or suspecting a malicious activity alarms should be triggered.

6.6 User management
The user management has a significant role in an OS. It is important to have each user restricted to the appropriate system resources. One of the biggest mistake that the system administrators can make is to have critical or insecure processes running as root user. Additionally violating the user management by sharing accounts to many users can have severe impact in case of a misuse or a security breach. The users management should be as strict as possible making it harder for an attacker who managed to gain access in the targeted system to steal confidential information or cause troubles.

6.7 Network Management Systems
Network management systems (NMS) are monitoring the resources and the status of a system. These systems can be configured to raise alarms under certain conditions such as heavy load. They have the ability to diagnose and even predict potential errors. Without them it is impossible for system administrators to keep track of the server status.

There are many ways that NMS work. The most common way is to use the SNMP protocol, although there are many monitoring implementations that does not require the use of SNMP. The administrator installs a client daemon that forwards the collected information to the central monitoring server. The the data received by the distributed servers can then be analyzed and visualized for all of the monitored servers in a centralized system, web application or server.

A list of popular NMS systems are briefly reviewed below. No matter which NMS system will be used by the SME, the secure architecture approach stays the same [44].

Nagios
Nagios is one of the most popular monitoring systems. The core is free and open-source but the full product (XI User Interface and system frameworks) is commercial. The free version of Nagios requires a lot of configuration effort to be configured properly. For this reason it is not recommended for an SME that does not have an Nagios experienced employee, enough time or the resources to spend on it [45].
**Icinga**

Icinga is a Nagios clone that is totally open-source with some modifications in the core and a big community [46].

As Nagios, Icinga requires a lot of configuration time and resources to spend, which does not make it not a good choice for an SME that does not have the required time and expertise.

**Observium**

Observium is a distributed monitoring system which is using RDDtool for visualizing the collected data. The goals of the product are automation and the ease of use [47].

The installation is an easy procedure that makes it a good monitoring choice. The free version does not support alarming which is a really important feature for an NMS. For this reason it should be combined with an alarming system and not as a standalone product.

**OpenNMS**

OpenNMS is great NMS that provides an enterprise monitoring and alarming solution at no cost [48].

The proper configuration of the system requires a lot of work. Fortunately OpenNMS company is selling training services for their product. One limitation of this system is that it does not support SNMPv3 yet.

**Zenoss**

Zenoss is a unified monitoring and event management system with an open source core. There are two versions of the product, a free with limited functionalities and a commercial one. The free version can support up to 1000 devices with no analytics available. It has a big open community for support. Despite all of its limitations it has many useful capabilities that can be useful for a SME [49].

**Zabbix**

Zabbix is an Enterprise class Monitoring solution with problem detection functionalities. It is licensed under GPL and is a free product. Every six months a new version is released. Additionally every 1.5 years a new LTS (Long Term Support) is released with 5 years of support [50].

The installation of Zabbix is an easy procedure that can save a lot of administration time. It can be a great choice for small and big enterprises.

**Ganglia**

Ganglia provides a scalable monitoring system for distributed systems. It supports up to 2000 nodes and it is targeting high performance computing systems. The core system is developed in C with a PHP front-end and is licensed under BSD. The visualization of the data is being done with RDD. This system does not support alarm management but a supporting tool (ganglia-alert) can be used for this [51].

**Cacti**

Cacti is a distributed network graphing tool. It uses advanced templates and RDD for the visualization of data. It is free and open source licensed under GNU. Cacti requires to add a plug-in for alarming [52].

**Conclusion**

There are many network management solutions that can fit the needs of an SME. This is a choice that differs from SME to SME. The goal of this project is not to do an NMS comparison. The architecture can work with any of the NMS no matter the final choice. Although some of them may have more gains than others. Different requirements and
criteria that every SME set should be used as input for the choice of NMS. In case of Softwerk the ideal case is to use Zabbix because a number of their servers are already using it. Additionally it supports alarms and resource monitoring that are the criteria described in Section 5.

6.8 Host based intrusion detection systems

Intrusion detection systems in general can be categorized in Network and Host based. There is no comparison between them, each of those two have a different purpose. Network based IDS are used to identify and prevent network based attacks by analyzing the traffic of the network. On the other hand, host based IDS are identifying and preventing attacks on a single host, not on a network as NIDS.

For securing a single VPS, a Host based IDS is needed. For this project OSSEC was selected because it is free and open source [53].

OSSEC key benefits [54]:

- Compliance Requirements: It assists the system with compliance requirements such as PCI, HIPAA and etc.
- Multi platform support: It supports Linux, Solaris, AIX, HP-UX, BSD, Windows, Mac and Vmware ESX.
- Custom Alarms and Real-time handling: Support configuration for custom alerts integrated with syslog, smtp, sms that can sent to an email to the administrators or communicated to other devices such as cell phones. Additionally active response options for blocking an attack are available.
- Integration with other systems and the current infrastructure: It can integrate with other systems and software that are already in production.
- Centralized management: It provides a simplified way of centralized management for managing the servers policies across multiple hosts.
- Agent and agent-less monitoring: It supports agent based and agent-less system monitoring and networking equipment (such as switches, routers and firewalls).

OSSEC Key Features

- File Integrity check: The most common pattern of an attack is to make a change inside the system. The purpose of the integrity check is to detect these changes and raise an alarm that shows what exactly happened. This alert can be an attack, misuse or even a totally innocent modification in the system (such as the installation of a new package).
- Log Monitoring: Daemons, applications and operating systems are generating logs in order to provide a way of user interaction. OSSEC raises an alert if an anomaly (attack, misuse, errors and etc.) has been detected by collecting, analyzing and correlating the produced log files.
- Rootkit detection: Attackers are hiding their actions and quite often also hides viruses inside the kernel of the operating system. Rootkit detection can make an alarm when an infection attempt has been detected.
- Active response : Active responses will block an attack in real time before the administrator take action. This feature is saving a lot of time and provides a new layer of protection to the system.
- Operating System event monitoring: It monitors OS events such as opening ports, changing permissions and adding new users to the system.
During the research, the conclusion was that all of these features are eliminating the need for an Antivirus software. With OSSEC, a virus will be detected at least at the following stages:

1. During the penetration into the system.
2. When a malicious action will be performed by the virus, and in most cases will be blocked by the active response feature.
3. If the virus edit a file in the system (an alarm will be raised by integrity checking).
4. At the kernel infection stage (if the virus is infecting the kernel).

6.8.1 Integrating Local with Remote Security
OSSEC can be configured to read the log files of the remote services and analyze them. When a malicious activity is detected in the log files, OSSEC can be configured to actively respond to the attack.

In Softwerk's case the OSSEC is integrated with SSH, Apache and ModSecurity.

There are many benefits integrating OSSEC and SSH:

1. Automatic analysis of log files.
2. Automatic Brute-force attack detection, alarming and prevention by blocking the attackers IP address for a few hours.
3. Automatic account locking in case of multiple brute-force attacks

Integrating OSSEC with ModSecurity results in:

1. Automatic firewall configuration for blocking the attacking IP addresses for a couple of minutes.
2. Alarming when an attack is taking place.

Integrating with Apache results to:

1. Automatically access log file analysis and alarming when content that does not exist is requested.
2. Automatically error log file analysis and alarming when errors occurs.
6.9 Putting All Together

Figure 6.4 shows the resulting architecture. The purple box is the virtualization technology where it hosts the OS of the server. The OS contains the NMS, WAF and IDPS. Each of those systems have their own purposes listed in the green boxes. Additionally the WAF is feeding the IDPS with log so it can actively respond to the HTTP attacks.

Figure 6.4: Architecture Model
7 Policy and Guidelines

In this section, the policy and guidelines are presented.

7.1 Overview
The presented policy is for Softwerk. It is based on SANS server security and web application security policy. Other SMEs can build their policy based on Softwerk's [55] [56].

Vulnerable servers and web applications have the largest amount of attacks. There are essential servers and web services/applications which have to be configured properly configured and have security audits before placed in production.

7.2 Purpose
The purpose of this security policy is to standardize the server administration and configuration of Softwerk Virtual Private Servers. Proper application of this policy will maximize security of the servers and minimize potential threats.

7.3 Scope
All Softwerk employees, contractors, external partners and temporary workers must comply with this policy. This policy applies to VPS administration, equipment and web application/services deployment.

7.4 Policy and Guidelines
The necessary actions must be taken when administrating servers and deploying applications on production to protect the CIA (Confidentiality-Integrity-Availability) of the providing services and customers data. Including restrictions of physical access to servers only to authorized employees and technicians.

The necessary actions include:

- Physical and Remote access restriction to the right employees.
- Employees that have remote access to the servers must have:
  - Hard disk encryption
  - Screen lock
  - SSH access to the servers using password protected private keys.
- Enforce Secure Passwords. A secure password must have:
  - At least 10 characters length
  - Excluding words
  - Contain Uppercase, Lowercase, numbers and special characters
  - Should be rotated at least one time per 3 months
- Disallow installation of unauthorized or non-signed repositories and software on servers.
- Enforce health monitoring on the servers.
- Enforce Intrusion Detection and Prevention Systems on the servers that store sensitive information.
• Sensitive information must be stored encrypted to the servers.
• Development servers must not be accessible on public.
• Web applications and services that contain critical information must have security audits (penetration testing or vulnerability assessment) before placed on production. After each update on the application, a new security audit must be enforced.
• Web applications and services that never before had security audits must be protected by a web application firewall.
• Java web application and services must be checked using OWASP dependency check before put on production.
• Vulnerabilities discovered on the production systems with high impact must be fixed immediately. Whenever a vulnerability fix is not available on a system that contain sensitive information, the server should be turned off to mitigate the risk of data critical leakage.
• Vulnerabilities discovered on the production systems with medium or low risk must evaluated before being fixed.
• On each server the deployment or administration, the Softwerk security checklist must be updated and submitted to the Softwerk's security team for evaluation.

7.5 Policy Compliance
Softwerk policy compliance describes the policy measurement, exceptions and non-compliance.

7.5.1 Compliance Measurement
Softwerk security team is responsible for verifying compliance with this policy.

7.5.2 Exceptions
Exceptions can only be approved by Softwerk security team.

7.5.3 Non-Compliance
Violation of this policy may have severe disciplinary actions, up to contract/employment termination.
8 Case Study

The following is an application of a security checklist that we developed and applied in one of Softwerk's servers. Note: Confidential data of the server have been replaced in the checklist.

Server Information

<table>
<thead>
<tr>
<th>Server name</th>
<th>Timelog</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server Client</td>
<td>Softwerk</td>
</tr>
<tr>
<td>Public Accessible</td>
<td>No</td>
</tr>
<tr>
<td>Server Description</td>
<td>This is the time logging server, where the employees are reporting their time.</td>
</tr>
</tbody>
</table>

Server Security

<table>
<thead>
<tr>
<th>Security Feature</th>
<th>Yes/no (If no, Specify the reason)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intrusion Detection/Prevention System (Required)</td>
<td>Yes</td>
</tr>
<tr>
<td>Network Management System (Required)</td>
<td>Yes</td>
</tr>
<tr>
<td>Web Application Firewall (Optional)</td>
<td>No, the web application had recently been penetration tested.</td>
</tr>
</tbody>
</table>

Installed Technologies/Services

<table>
<thead>
<tr>
<th>Software/Service</th>
<th>Technologies</th>
<th>Manually Installed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java (specify Oracle or Open-Java)</td>
<td>OpenJDK</td>
<td></td>
</tr>
<tr>
<td>Programing Languages (Ruby, PHP, Python)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBMS (MySQL, MariaDB, PostegreSQL, MongoDB)</td>
<td>MariaDB</td>
<td></td>
</tr>
<tr>
<td>Web Servers (Apache, Nginx, Tomcat, Glassfish)</td>
<td>Tomcat</td>
<td></td>
</tr>
<tr>
<td>Virtualization Technology (Xen, KVM, Virtual-box, OpenVZ,VMWare)</td>
<td>KVM</td>
<td></td>
</tr>
<tr>
<td>Other</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SSH Hardening (Required)

<table>
<thead>
<tr>
<th>Setting</th>
<th>Yes/no (If no, Specify the reason)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disallow root login.</td>
<td>yes</td>
</tr>
<tr>
<td>Setting</td>
<td>Value</td>
</tr>
<tr>
<td>------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>Disallow password login</td>
<td>yes</td>
</tr>
<tr>
<td>Disable empty passwords</td>
<td>yes</td>
</tr>
<tr>
<td>Change default port (Fill in SSH port)</td>
<td>No, it accessible only from the intranet.</td>
</tr>
<tr>
<td>Allow only login using private keys.</td>
<td>yes</td>
</tr>
<tr>
<td>Disable the functionalities that are not needed or being used.</td>
<td>yes</td>
</tr>
<tr>
<td>SSH accessible only by intranet's IP address range.</td>
<td>yes</td>
</tr>
<tr>
<td>Disable weak ciphers in order to be protected by drop down attacks.</td>
<td>yes</td>
</tr>
<tr>
<td>Remove SSH fingerprints</td>
<td>yes</td>
</tr>
<tr>
<td>Limit the idle log-out time</td>
<td>yes</td>
</tr>
<tr>
<td>Banner with a warning message</td>
<td>yes</td>
</tr>
<tr>
<td>Limit SSH listening interfaces to one</td>
<td>yes</td>
</tr>
<tr>
<td>Users allowed to connect</td>
<td>time_admin</td>
</tr>
</tbody>
</table>

**Web Server Hardening (Required)**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disable directory listing</td>
<td>yes</td>
</tr>
<tr>
<td>Run the web server in its own user and group</td>
<td>yes</td>
</tr>
<tr>
<td>Remove write permission from the web-root (if possible).</td>
<td>yes</td>
</tr>
<tr>
<td>Files are uploaded inside the database. If not, they should uploaded to a directory outside of the web-root without execution permissions.</td>
<td>yes</td>
</tr>
<tr>
<td>Hide web-server version and signatures</td>
<td>yes</td>
</tr>
<tr>
<td>Disable Database and Server errors.</td>
<td>yes</td>
</tr>
<tr>
<td>Update the server in a regular basis</td>
<td>yes</td>
</tr>
<tr>
<td>Disable unnecessary server modules</td>
<td>yes</td>
</tr>
<tr>
<td>Limit the allowed Request size</td>
<td>yes</td>
</tr>
<tr>
<td>Enable logging</td>
<td>yes</td>
</tr>
<tr>
<td>Setting</td>
<td>Yes/no (If no, Specify the reason)</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>Set connection Time-Out</td>
<td>yes</td>
</tr>
<tr>
<td>Limit the maximum number of concurrent</td>
<td>yes</td>
</tr>
<tr>
<td>connections</td>
<td></td>
</tr>
<tr>
<td>Database Hardening (Required)</td>
<td></td>
</tr>
<tr>
<td>Listening only to local IP</td>
<td>yes</td>
</tr>
<tr>
<td>Applications are connected to the DB only</td>
<td>yes</td>
</tr>
<tr>
<td>as their own restricted user</td>
<td></td>
</tr>
</tbody>
</table>
9 Summary and Conclusions

9.1 Summary
This study demonstrated a security policy and a secure server architecture. The research also introduced the threats and potential attacks to the servers. The standardized architecture protects servers and their services against security threats. This is done by combining a network management with an intrusion prevention system. Additionally, a web application firewall can be set to the protected system when needed in order to provide an extra layer of security. With these components, an anomaly or attack to the server is detected and the internal security mechanisms raise an alarm. This alarm is handled by the intrusion detection system which actively responds to the event by blocking the attack. Simultaneously, the system informs the administrator about the incident. With these actions, there is less need for server administration and maintenance while providing more reliable services. To complete the server architecture, a security policy and checklist are provided with the proper directives in order to prevent potential pitfalls for the system administrator.

9.2 Conclusions
The research question addressed by this thesis is: How can Softwerk deploy and administrate existing security tools to improve the security of their servers?

To address the above research question a secure and suitable, i.e., meeting Softwerk's needs, a server architecture was identified (Section 5) for hosting the services provided by Softwerk. The architecture was kept intact and a policy was defined for maintaining it (Section 7), in order to avoid getting existing problems sooner or later again.

To achieve this, the following goals were targeted:

1. Identify and document Softwerk's current operational requirements to make clear which services and servers exist and how (by whom and from where) they are used. Further more, threats should be identified and described. This goal was reached by documenting every company server (Section 4.2), including graphs providing an overview (Section 4.1) as well as an overview of current threats (Section 4.3).

2. Based on the operational requirements and threats identified, create a generic architecture for a secure server infrastructure which fulfills the operational requirements, while being resistant to threats. The architecture is to minimize diversity to maximize maintainability, i.e., minimize time and costs needed for maintaining and operating the required servers and services. For each of the architecture areas, different alternative solutions should be compared choosing the most suitable solution according to Softwerk's needs as described in Section 6. High diversity due to coexisting variations would otherwise make administration and operation difficult, thus reduce the security of the system (too many things to keep track of).

3. Define a policy and guidelines for administrating and maintaining such an architecture with the servers implemented according to it. This goal is reached if a document (incl. possible scripts) has been created describing how to administrate and maintain the identified parts of the architecture. This is done by creating a policy as documented in (Section 7.4).

4. Migrate as many existing servers as possible to this architecture using the policy and guidelines. This goal is reached if at least one (1) of the servers have been migrated using the architecture in compliance with the policies. This is done by
migrating (so far) one server hosting the Timelog service to the proposed architecture while documenting it in a checklist/protocol (Section 8).

By applying the above goals the research question “How can Softwerk deploy and administrate existing security tools to improve the security of their servers?” is answered.
10 Future Work

In this section topics for future work are outlined.

Investigation in Backup Solutions

- Having a centralized backup and recover center for distributed VPS configurations, log files, databased and application data. It should automatically backup the systems and handle the backup rotation.
- The backup mechanism should be secure without being able to cause problems to the centralized backup server.
- The backups in the centralized backup server should be secured from potential threats

Investigation of IPv6 compatibility to this Architecture

- Investigate a solution that works in both Ipv4 and IPv6 version.
- Evaluate if this architecture can work with IPv6

Log Management

- Centralized log management solution
- Log management security
- Log management rotation
- Logging strategy

Investigation in Virtualization Technologies for servers

- Investigation in security
- Investigation in stability
- Investigation in performance
- Investigation in limitations

Security is killing usability

Many security implementation and measures applied usually are forcing the user to do actions that decrease the productivity and the usability of the system. These security mechanisms for research can be multi-factor authentication, captcha and etc.
11 Bibliography


41
12 Appendix

12.1 Security checklist Template

Server Information

<table>
<thead>
<tr>
<th>Server name</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Server Client</td>
<td></td>
</tr>
<tr>
<td>Public Accessible</td>
<td></td>
</tr>
<tr>
<td>Server Description</td>
<td></td>
</tr>
</tbody>
</table>

Server Security

<table>
<thead>
<tr>
<th>Security Feature</th>
<th>Yes/no (If no, Specify the reason)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intrusion Detection/Prevention System (Required)</td>
<td></td>
</tr>
<tr>
<td>Network Management System (Required)</td>
<td></td>
</tr>
<tr>
<td>Web Application Firewall (Optional)</td>
<td></td>
</tr>
</tbody>
</table>

Installed Technologies/Services

<table>
<thead>
<tr>
<th>Software/Service</th>
<th>Technologies</th>
<th>Manually Installed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java (specify Oracle or Open-Java)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Programing Languages (Ruby, PHP, Python)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBMS (MySQL, MariaDB, PostgreSQL, MongoDB)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Web Servers (Apache, Nginx, Tomcat, Glassfish)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Virtualization Technologies (Xen, KVM, Virtual-box, OpenVZ, VMWare)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SSH Hardening (Required)

<table>
<thead>
<tr>
<th>Setting</th>
<th>Yes/no (If no, Specify the reason)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disallow root login.</td>
<td></td>
</tr>
<tr>
<td>Disallow password login</td>
<td></td>
</tr>
<tr>
<td>Setting</td>
<td>Yes/no (If no, Specify the reason)</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>Disable empty passwords</td>
<td></td>
</tr>
<tr>
<td>Change default port (Fill in SSH port)</td>
<td></td>
</tr>
<tr>
<td>Allow only login using private keys.</td>
<td></td>
</tr>
<tr>
<td>Disable the functionalities that are not needed or being used.</td>
<td></td>
</tr>
<tr>
<td>SSH accessible only by intranet's IP address range.</td>
<td></td>
</tr>
<tr>
<td>Disable weak ciphers in order to be protected by drop down attacks.</td>
<td></td>
</tr>
<tr>
<td>Remove SSH fingerprints</td>
<td></td>
</tr>
<tr>
<td>Limit the idle log-out time</td>
<td></td>
</tr>
<tr>
<td>Banner with a warning message</td>
<td></td>
</tr>
<tr>
<td>Limit SSH listening interfaces to one</td>
<td></td>
</tr>
<tr>
<td>Users allowed to connect</td>
<td></td>
</tr>
</tbody>
</table>

**Web Server Hardening (Required)**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Yes/no (If no, Specify the reason)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disable directory listing</td>
<td></td>
</tr>
<tr>
<td>Run the web server as a different it is own user and group</td>
<td></td>
</tr>
<tr>
<td>Remove write permission from the web-root (if possible).</td>
<td></td>
</tr>
<tr>
<td>Files are uploaded inside the database. If not, they should uploaded to a directory outside of the web-root without execution permissions.</td>
<td></td>
</tr>
<tr>
<td>Hide web-server version and signatures</td>
<td></td>
</tr>
<tr>
<td>Disable Database and Server errors.</td>
<td></td>
</tr>
<tr>
<td>Update the server in a regular basis</td>
<td></td>
</tr>
<tr>
<td>Disable unnecessary server modules</td>
<td></td>
</tr>
<tr>
<td>Limit the allowed Request size</td>
<td></td>
</tr>
<tr>
<td>Enable logging</td>
<td></td>
</tr>
<tr>
<td>Set connection Time-Out</td>
<td></td>
</tr>
</tbody>
</table>
Limit the maximum number of concurrent connections

<table>
<thead>
<tr>
<th>Setting</th>
<th>Yes/no (If no, Specify the reason)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Listening only to local IP</td>
<td></td>
</tr>
<tr>
<td>Applications are connected to the DB only as their own restricted user</td>
<td></td>
</tr>
</tbody>
</table>