ISSN 2347 - 3983
Volume 8. No. 10, October 2020
International Journal of Emerging Trends in Engineering Research
Available Online at http://www.warse.org/IJETER/static/pdf/file/ijeter148102020.pdf
https://doi.org/10.30534/ijeter/2020/148102020

Automated Security Enhancement Framework for Linux Operating System

Bimandika Hasanah[1], **Abba Suganda Girsang**[2]
[1]Computer Science Department, BINUS Graduate Program – Master of Computer Science, Bina Nusantara University, Jakarta, Indonesia,11480, bimandika.hasanah@binus.ac.id
[2]Computer Science Department, BINUS Graduate Program – Master of Computer Science, Bina Nusantara University, Jakarta, Indonesia,11480, agirsang@binus.edu

ABSTRACT

Cloud computing has evolved at an incredible speed, and, in many organizations, it entwined with the sophisticated technological landscape that supports critical daily operations. The cloud computing environment gives rise to a new type of risk, and the user faces many challenges in protecting their existing IT environment. Security in Cloud computing has become a matter of global interest, and it involves securing information by detecting, preventing, and responding to cyber-attacks. Several research efforts have been made in different domains, each having specific features and peculiarities to address various security breaches. Security enhancement as part of prevention is essential in the cloud environment, especially in the Operating System. Security enhancement on the operating system can be done in several ways, such as updating the operating system, updating security, and Hardening. Hardening the operating system requires a long set of processes and execution. Security benchmarks can be a good fundamental for Hardening, which can improve operating system security. The security benchmark is well researched and tested. There are many recommendations for security benchmark need to implement to the operating system, to execute all security benchmark manually will take time and much effort. To simplify and accelerate the hardening process, this paper will propose a framework to automate the Hardening for Linux OS, which can support parallel execution in several servers.

Key words: Hardening, Linux, Container, Security Benchmark

1. INTRODUCTION

Cloud computing has recently emerged as a new paradigm for hosting and delivering utility-oriented IT services to users over the Internet [1]. The main objective of cloud computing technology is to ensure availability, high reliability, and scalability of infrastructural facilities, which include hardware, software, platforms, services, and software that could be distributed to different computing locations.[2] Cloud computing has several advantages. For instance, a cloud server that makes it possible for a person or a company to have a server without have any real physical server. It can reduce the cost and easy to maintain.[3]. The rapid growth in field of "cloud computing" also increases severe security concerns. Security is needed not only for services provided over cloud but also for the cloud resources. Providing services over cloud implies providing secured services internally [4]. Almost every part of cloud computing has a security vulnerability, and it can be a weak point or gateway for attackers to interfere system or stole the data. Cloud computing systems are often built from the existing single machine OS, usually Linux [5]. The Linux operating system was not able to shine in the desktop world. However, it is currently the go-to operating system for the cloud, and upon which the largest cloud infrastructures in the world have been built on [6]. The reality is that Ubuntu Server is the most widely-used Linux distribution for deploying cloud-based applications. [7].

Security on Linux is a challenge for cloud computing users because, at this time, most operating systems are not designed to make security the main focus. The operating system only focuses on features, usability, communication, and functionality. The security of Linux depends on many configuration files, both at the system level and application level. Most important is security of the Linux system is never static. Once you secure your Linux system, it does not perpetually stay secure because operational and functional changes had done through threats or new exploits are available for packages or applications hence need of secure system [8]. An effort to enhance the security of the Linux operating system can be accomplished by Hardening. Hardening the operating system is a process of security improvement through various ways to make the operating system more resistant to security threats. Besides that, Hardening is a step that can increase the performance of the server to the maximum level, close security holes, and close access points that will make the server more challenging to exploit.

The term "Hardening" refers to securing the system. Like any other operating system, application-level security flaws leave Linux vulnerable to a variety of malicious attacks. Over the years, many tools and techniques have been developed to "harden" Linux hosts in an attempt to mitigate the risk posed by buggy software [9]. Hardening processes, like removal of unnecessary usernames and logins, software, and services, that results in a robust, well-managed, and provides enterprise-grade security service for the virtual systems [10]. Security benchmarks are one of the best approaches that can be used for Hardening. A security benchmark is a set of (standard) recommendations against which the security strength of different systems can be compared. The recommendations are coupled with auditing activities specifying how to collect data for evaluating the recommendations. The result of a security benchmark evaluation is a score that represents the security strength of a specific product/service/deployment [11]. Another better approach could be using industry-standard benchmarks for secure configuration. The Center for Internet Security (CIS) providing security benchmarks for various platforms. These benchmarks are well researched and tested [12]. The Center for Internet Security is a cross-industry organization that promotes best practices for securing systems. They publish the Critical Controls, a list of 20 key practices for running a secure IT organization, and the CIS benchmarks, a set of consensus-based security hardening guidelines for Unix/Linux and Windows OS, mobile devices, network devices, cloud platforms, and common software packages. These guidelines are designed to meet the requirements of a range of different regulations [13]. The CIS Benchmark was developed through the voluntary efforts of contributors from experts in the field of cybersecurity, technology vendors, members of the public and private communities, and the CIS Benchmark Development team. Each release of the latest benchmark is based on the new vulnerability founded.

CIS Benchmark on several Linux OS is published around 220 security recommendations for securing the operating system. To implement all security recommendation manually, it will take times, effort, and has a high rate for error execution. To simplify and accelerate the execution process of Hardening, This paper will propose a framework that can automate and implement Hardening for Linux OS. This Framework supports parallel execution on several servers at the same time. This Framework will adopt the CIS Benchmark as the fundamental of security recommendation, and also This Framework will take advantage of configuration management and wrapped up with container technology for automation and agility.

Not only the CIS Benchmark, but the Framework is also possible to automate other security rules or security policies.

2. RELATED WORKS

A variety of research has been conducted to find the method for hardening the operating system.

2.1 Linux Hardening

There is some research [9], [14] in Linux Hardening, and one of the research [14] focuses on practical securing Linux production systems. It discusses basic Linux security requirements for systems that need to pass various audits in an enterprise environment. This research also presents to detect the system's vulnerabilities by scanning configuration files and server files to determine the computer activities by scanning the log files, thereby securing the system by replacing the vulnerable attributes with secured attributes. Application security is ensured by scrutinizing the signatures of various applications and displaying all the functionalities in GUI format, making it more user friendly. An essential step in securing a Linux system is to determine the primary function or role of the Linux server. Linux Administrator should have a detailed knowledge of what is on the system. Otherwise, it will difficult to understand what needs to be secured, and hence securing the Linux system proactively will not be that effective. Therefore, it is very critical to look at the default list of software packages that do not comply with security policy if the Linux server has fewer packages to update and to maintain when security alerts and patches are released.

An essential step in securing a Linux system is determining the function or primary role of a Linux server. To ensure complete security, the Linux operating system must be secured from the following aspects:

• User Security
• Network Security
• Package Security

To ensure user safety, researchers do things like:

1. Vulnerability assessment which is an internal audit of network and system security; results that indicate confidentiality, integrity, network availability.

Table 1: Vulnerability Assestments

Vulnerability	**Attacks**	**Counter measure**
No separate partition for /boot, /, /home, /tmp, and /var/tmp	System crash and data loss	Create separate partition for /boot, /, /home, /tmp, and /var/tmp
Unnecessary software's	Software vulnerability attack	Install minimum software's
maliciously	System instability,	Install Signed

altered package	System crash, and data loss, data still	Packages
No BIOS password	Stealing/Changing Data Using a Bootable Linux CD	Give BIOS password
Single User Mode access	Access as root user without password	Password protecting BIOS
Access to the GRUB Console	change its configuration or to gather information using the ***cat*** command.	Password protecting GRUB
Access to Insecure Operating Systems	If it is a dual-boot system, an attacker can select an operating system at boot time (for example, DOS)	Password protecting GRUB
A weak password, no password or default password	Cracking of weak passwords	1) Enforcing Stronger Passwords 2) Restricting Use of Previous Passwords 3) Locking User Accounts After Too Many Login Failures
No password Aging	Use of Cracked password over a long period of time	Apply good password Aging
root access to individual users	1) Machine Misconfiguration 2) Running Insecure Services	1) Root Disallowing Access 2) Disallow Remote Root Login 3) Disabling root access via any console device (TTY)

2. Password Security: passwords are the primary method used by Linux companies to verify user identity. This is why password security is essential for the protection of users, workstations, and networks.

3. Monitoring logs: To determine the operational status of the system and applications that are in progress, monitoring logs play an important role.

For network security, this research focuses on several things, including:

• Disabling unused services like some network protocols are inherently less secure than others. Including services that: Send usernames and passwords over an unencrypted network - Many older protocols, such as Telnet and FTP, do not encrypt authentication sessions and should be avoided whenever possible.

• Using a Firewall is an important step to protect network security. A firewall can be used to improve access control between two or more networks. The Linux kernel uses the net filter facility to filter packages, allowing some of them to be received or bypassing the system while stopping other packages.

• The use of TCP Wrapper which adds a layer of protection by defining which hosts are allowed or are not allowed to connect to "wrapped" network services. One of the wrapped network services is the xinetd super server. This service is called a super server because it controls connections to a subset of network services and further improves access control.

While in the security of the package, the researchers proposed the use of RPM, which is an open packaging system, which runs on Red Hat Enterprise Linux and also Linux & UNIX systems. The utility only works with packages created to be processed by rpm packages. RPM manages the installed package database & their files. This software provides functionality to retrieve the number of packages installed and their verification reports generated. Software packages are published through repositories. All famous repositories support package signing.

The main contribution of this research is to designing and building a secure file system and network that was developed with the express goal of enhancing file data security and network security in the Linux kernel. The main objective is to detect the vulnerabilities in the system by scanning configuration files and server files to determine the computer activities by scanning the log files, thereby securing the system by replacing the vulnerable attributes with secured attributes. In network security, we provide security for web server, ssh server, etc. application security is ensured by scrutinizing the signature of various applications and displaying all the functionalities in GUI format, making it more user friendly.

Another research [14] in linux hardening is aims to explore and highlight the basic security configurations that must be carried out to strengthen the security posture of a standard Linux Operating System installation. This research project explores the main weaknesses and default configurations that never change when building a production Linux Server, making the target server easy to hack through the internet. By following some industry best practices and fiddling with several security configurations, Linux Servers can be secured appropriately. This research project explores and suggests best practices for Hardening standard Linux services such as Secure Shell (SSH), Apache Web Server, and configuring host-based firewalls (IPTABLES) to block connections to unwanted ports and block bad traffic. Various operating systems can be used in a server system. However, this project focus on the Linux Operating System, CentOS 5.9 Final was chosen as the operating system for this project.

Some technical aspects carried out during the research are:

• Perform unwanted packages and services that are installed on the server. When the installation type, for example, "Desktop GUI" is selected, many applications and services are installed on the server. The server must be installed with special applications needed for the proper operation of the services required from the server. So to keep the server simple and reduce threats from vulnerabilities related to unused applications, we must remove unwanted packages and services.

• The second phase is to do hardening security, which is done using operating system updates. The system must be updated regularly. New patches must be applied, when launched. To update a Linux system, the yum utility can be used.

• Establish security requirements and provide complex passwords on the operating system to ensure that the user's password is secure, security policies must be enforced by management and also by the system administrator. By configuring the server for stringent password requirements that cannot be avoided. Password security includes password aging requirements, which means forcing passwords to expire every 90 days, password complexity.

• Secure SSH, Secure Shell protocols are the most common remote administration and management tools used for Linux and Unix-based operating systems. SSH provides security and session encryption features between server and client, to use security and encryption features it is necessary to configure /etc/ssh/sshd_config, the configuration that is usually done to increase ssh include, disable root login, restrictions on user access and authentication using RSA key.

• Kernel Security Parameters, ensure kernel security parameters are set correctly. Parameters such as net.ipv4.tcp_syncookies will protect from SYN attacks which are denial of service attacks, disable source routing, disable ICMP redirect messages, enable IP Spoofing protection, ignore ICMP message requests if possible, ignore broadcast stroom requests in ICMP, protect against ping flooding, activates bad error message protection and allows logging of fake, source routed and redirected packets to analyze the source of the attack. This kernel parameter can be applied by adding appropriate keyword pairs and values to the /etc/sysctl.conf file. To apply changes without restarting the server, run the sysctl –p command.

• Login Alert, it is useful for knowing who is logged into the server. If the attacker gets access to the server, they can clear the log, and it might be difficult for administrators to know that the server has been compromised. This is very useful if you apply a script that will notify administrators about all logins via email. Because the script is executed as soon as the user is logged in, an email is sent, and thus the attacker may not be able to cover the track. The administrator can immediately find out that someone has accessed the server. The script path must be set in /etc /profile.

• TCP Wrapper "TCP wrapper adds an additional layer of protection by determining which hosts are allowed or not connected to" wrapped "network services (Centos, TCP Wrappers, and xinetd). The TCP wrapper provides an additional layer of security for services using the libwrap library. Services like SSH, portmaps, telnet can be protected using TCP Wrappers. In addition to proper firewall configuration, the use of TCP Wrappers can add a layer of security. With TCP wrappers, we can determine which networks or hosts are allowed to use certain services on the server. This is done by rejecting all hosts in /etc/hosts.deny and selectively allowing hosts and networks in the /etc/hosts. Allow file.

2.2 Security Benchmark

A security benchmark is a set of (standard) recommendations against which the security strength of different systems can be compared. The recommendations are coupled with auditing activities specifying how to collect data for evaluating the recommendations. The result of a security benchmark evaluation is a score that represents the security strength of a specific product/service/deployment. A research [11] also conducted to secure OpenStack by researching a security benchmark for OpenStack. This research defines a security benchmark for OpenStack, which is an example and refinement of the CIS Benchmark based on OpenStack's security guidelines in describing its evaluation using a guaranteed platform called Moon Cloud.

The security benchmark for OpenStack first maps the three profiles (Virtual, Cloud, End User) identified by the CIS benchmark on OpenStack's core services to overcome its uniqueness, including the concept of shared responsibility and the cloud layer, as follows.

• Virtual: this profile relates to all physical nodes where the

OpenStack service is installed, explicitly handling hardware configuration, Linux OS, system virtualization, and system service configuration.

• Cloud: this profile is related to OpenStack services and add-ons. This involves OpenStack core services, admin operations, and defined configurations.

• Users: this profile is related to using OpenStack users. It discusses how users can secure their OpenStack projects, including VMs, virtual storage, and network configurations.

The Researcher then made a generic CIS benchmark recommendation and added several new recommendations to the OpenStack core services based on the three profiles specified. Following the CIS approach, the recommendations mostly cover confidentiality, integrity, and access control attributes, while they can be expanded to assess any security attribute. In this research, the evaluation process is implemented, collecting evidence about the target system, for example, by testing monitoring activities on certain services. The evidence gathered makes it possible to verify whether the recommendation is. The Researcher used Container technology to evaluate and hardening Openstack with Python programming language, which can be seen at this link: https://github.com/SESARLab/openstack-security-benchmark. After Hardening, the Researcher uses a verification tool called Mooncloud.

2.3 Approach to Container Technology

Container technology has many advantages in improving the performance of an application. [15] Approaches to a system that comes with challenges that must be addressed, such as the higher complexity of development to continue production, stringent requirements, automation from every aspect, isolation of failures, testing challenges and so on. Fortunately, Docker has been introduced with key features and tools that can help overcome these challenges, some of the advantages of using Docker include:

• Speed up automation
Docker Container is naturally very suitable for microservice architecture because each can be used as a container for granular application deployment containing services because each launch and manufacture of a container can be done using a design script and with some supporting tools Docker container can accelerate the culture of automation in each cycle for software development

• Speed up Independence
Each container is an isolated box that can contain run time for certain services, with this advantage the developer team can work independently to implement services with any technology or language, process, tools they like and are different in each container

• Speed up portability
Docker places applications and all dependencies on portable containers between many platforms, including Linux and cloud distributions, different stakeholders from applications such as developers, testers, administrators and others can quickly run the same application on VMs, local computers, bare-metal servers or in the desired cloud computing.

• Speed up the use of resources
In the Docker each container consists of only the applications and dependencies needed by the application, the container runs as process isolation on the host operating system and shares the kernel with other containers, so even though the container is placed in a VM, in addition to utilizing resources in the VM the containerization technique also makes more portable and efficient, in bare metal, the lightweight nature of containers helps to run more instances than VMs that use higher resources

• Security
Many things offered by the Docker allow developers to do flexibly, maximize code security at various levels. When creating code, developers can freely use penetration, a test tool to test stress in any part of the build cycle. Because the source for building a docker image is explained explicitly and descriptively in the Docker build, the distribution component (docker file, docker-compose) developers can handle the docker image distribution more efficiently. They can also implement security policies as needed, also able to quickly harden services which cannot be made by putting it in the container docker, adding reliable security insurance to the service.

With these advantages, Docker is becoming a fast-developing technology and has been widely adopted by companies and institutions to increase the speed of the development process and resource efficiency.

Docker technology approach can be implemented in many software both for applications, websites, big data, and even security, one of the uses of the Docker as a platform used for vulnerability assessment. A research [16] conducted to develops a security testing framework using service-based services cloud for web developers to do security scans for their web applications without having any prior knowledge or prior experience in using and configuring security testing tools.

Based on the literature review above, the following is summarized in the following table 2

Table 2: Summary of Literature review

Author	**Publication**	**Method**
Arora et al., 2014	Linux Hardening	Explore the basic Linux security

		requirements for systems that need to pass various audits in the corporate
Nepal, 2014	Linux Server & Hardening Security	Explore and highlight basic Security configurations that are must be done to strengthen the security posture of a standard Linux Operating System Installation
Anisetti et al. 2017	A Security benchmark for Openstack	Creating a platform called moon cloud Using security benchmarking and inspired by the CIS Benchmark as a parameter in OpenStack and leverage a docker for the evaluation
Jaramillo et al. 2016	Leveraging microservices architecture by using Docker Technology	Suggest the use of Docker container technology for software development because it has many advantages
Pathirathna et al. 2018	Security Testing as a Service with Docker Containerization	Develop a vulnerability scanning system using Docker Container technology

2.4 Research Methodologies

This research is continuing several previous research such as Linux Hardening [9], Linux Server & Security Hardening [14], and Server Hardening: Securing Unix-like workstations [17] by proposing the new methods that are expected to have several advantages including:

• Optimization of execution time

• Support for parallel execution on several or even hundreds of servers both virtual machines and physical machines simultaneously

• Support Automation in the hardening process

This research will focus on improving the speed during the hardening process and also improving security on the Linux Operating System by proposing framework architecture. The Framework will use configuration management with a docker container baseline. The time of execution will be measured in second units.

This research will compare speed between the proposed Framework and old method that used to harden the Linux operating system. For the assessment of vulnerability will be using OpenVAS, Because the ability to detect vulnerabilities for various operating systems and applications makes it a popular tool among pentesters [18], Open Vulnerability Assessment System is not only a tool but a complete framework consisting of several services and tools, offering comprehensive and robust vulnerability scanning and vulnerability management solutions [19], we can be used OpenVAS to measure security and scan for vulnerabilities in the operating system before and after hardening process.

3. THEORY AND METHOD

3.1 Vulnerability

Sigmoidal vulnerability growth trends tend to follow the popularity of operating systems. Once the popularity reaches an inflection point, decreasing rate of adoption makes the given operating system less lucrative for exploitation development, which leads to fewer and fewer vulnerabilities [20], Significant growth of the number of vulnerabilities found in modern operating systems shows severe challenges and risks that users must face.

Vulnerability is a security flaw in a computing system, which can be consequently attacked and exploited by an attacker. There are various ways in which vulnerabilities can be exploited. Attackers can get commands executed in the normal way, or overcome restrictions in order to gain forbidden access to data, or trigger denial of service and system service termination [21]. Every software has its vulnerabilities, and vulnerabilities occur when developers make mistakes on the logic of the coding or use imperfect validation so that the software created has an unknown weakness. Several institutions focus on finding vulnerabilities in software, operating system, and device that provide datasets which can be checked by the wider community and can be used to make improvements to the vulnerabilities found. There is a number on each vulnerability founded and data set can be seen transparently, it is managed by several institutions such as CVE (Common Vulnerability and Exposure), NVD (National Vulnerability Database), VNDB (Visual Novel Database) and several Linux operating systems that manage findings own vulnerability also provides recommendations such as RHSA (Red Hat Security Advisory), SUSE-RU (Recommendation Update).

3.2 Hardening

Hardening in the operating system is a process of increasing defense and security in the operating system. Hardening is used to anticipate attacks to the operating system by configuring, update and create rules and policies to help securely govern the system, and removing unnecessary software and services, will make the server has minimum exposure then decrease a potential security risk.

Several methods can be applied to enhance security through Hardening, one of which is by manually configuring Linux using terminals, this method is very time-consuming because many configurations need to be changed. Another method that is widely used is to use a shell script to automate system hardening on Linux, by using configuration management this method is beneficial for improving security. However, some things are not optimal in this method is the level of difficulty in making scripts that will be very difficult for beginners to understand Linux.

Configuration management can be an alternative to hardening the system to simplify and accelerate the process. Scripts in the management configuration will be accessed using the internet and obtain transparent access also, keep updated; the Framework will be an ecosystem that supports increased security on Linux operating systems.

3.3 Container

Many techniques, methods, and technology have been developed to improve the scalability and elasticity of application deployments and operations in cloud computing. One of them is container-based virtualization. Container-based virtualization can provide higher density virtual environments and better performance than hypervisor-based virtualization [22]. Containers are an encapsulation of an application with its dependencies. At first glance, they appear to be just a lightweight form of virtual machines (VMs)—like a VM, and a container holds an isolated instance of an operating system (OS), which we can use to run applications [23]. In container virtualization, instead of having an entire Operating System guest OS, containers isolate the guest but do not virtualize the hardware. For running container one needs a patched kernel and user tools, the kernel provides process isolation and performs resource management. Thus all containers are running under the same kernel, but they still have their file system, processes, memory, etc. [24].

Docker is one of the popular Container-based virtualizations for deploying and managing applications. Docker provides convenient tools to combine files in images and run containers from images on end hosts. Each end host runs a daemon process that accepts and processes user commands [25]. Docker also enables a secure packaging and deployment of applications, supporting the DevOps model of speeding up the development life-cycle through rapid change, from prototype to production [26].

The Docker solution is consist of the following components, such as Docker engine and Docker Hub.

The Docker engine is for enabling the realization of purpose-specific as well as generic Docker containers. The Docker Hub is a fast-growing repository of the Docker images that can be combined in different ways for producing publicly findable, network-accessible, and widely usable containers [27]. Using Docker in the software development process is very beneficial because it can solve problems encountered in previous methods such as library dependencies, inaccurate documentation, dispel code rot on image versions, adoption and reuse [28], Docker can also be installed on personal computers for large business projects because it is a modern alternative container in the development of software in the microarchitecture services[15].

4. IMPLEMENTATION

4.1 Framework Architecture

The framework stacks are consist of several components, Such as, security benchmark which used as source policy hardening, Researchers and contributors whose role is to translate documents into code, source code management used to store code. Image repository containers used to store containers images from this Framework, the container host can use a various device such as a bare-metal physical server, virtual machine on cloud computing or private notebook as long as the device can be installed the Docker software and has a sufficient resource of CPU, memory, storage, and network. Each component of the Framework is integrated with other components and has a dependency with another, Framework Architecture and workflow described in Figure 1.

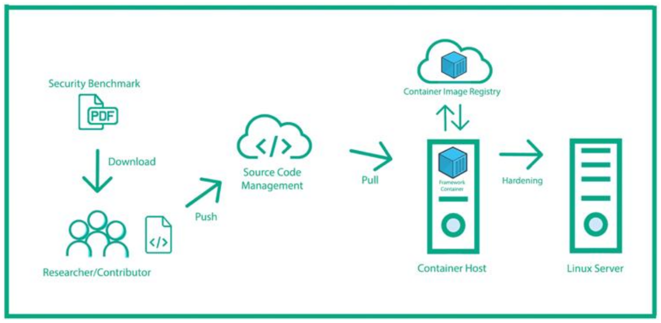

Figure 1: Proposed Framework Architecture and Workflow

The workflow starts by downloading the security benchmark and do convert it into code. After the coding process is complete, the code will be uploaded to the management source code, Management source code not only for store the code but also to add new features to the proposed Framework also to make the Framework accessible and used by everyone, apart from that everyone also can contribute to the development of this Framework and can also report bugs, add new ideas on approval of Researcher or contributor.

The Framework uses the CentOS as a baseline and installed configuration management for the execution of the Hardening process on the Linux OS. There is an automation function on the container which aims to update the code when first launched.

Currently, Framework supports four Linux Operating System and Deep inside Framework container, The hardening function divided into six functions as described in Figure 2.

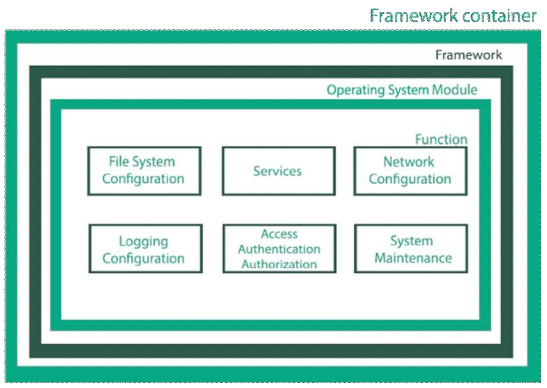

Figure 2: Software Architecture

The six hardening function is instantiation and refinement of the CIS Security benchmark with several additional functions to make the system vulnerable, The. Theil of each function are:

• The file system configuration function is used to protect the server from excess resources on disk and forcing the usability of mounting options on every mounted partition to prevent and minimize data loss if the Linux server crashed. This function also configure sticky bit on world-writable directories prevents users from deleting or renaming files in the directory that do not belong to users, setting software update by ensuring patch management system is configured and maintained, configure integrity checking tools to detecting unauthorized changes of configuration files by alerting when the files are changed, and the last function is for enforcing the Mandatory Access Control (MAC) to provides an additional layer of access restrictions to processes on top of the base Discretionary Access Controls.

• Service functions primarily for disabling unnecessary services to protect the system against as yet unreported vulnerabilities. Prevent the exploitation of the discovered vulnerability in the future, and if the service disables, it cannot be exploited.

• Network Configuration function is to enhance the network security on the system by configuring through kernel parameters, access-list control, and firewall settings.

• Logging and Auditing function is used to automatically monitor logs for intrusion attempts and other suspicious system behavior, by configuring Linux logging software and audit software, because It is essential that all logs can be monitored regularly and correlated to determine trends. A seemingly innocuous entry in one log could be more significant when compared to an entry in another log.

• Access, Authentication, and Authorization primarily to control operating system determines how the operating system implements accesses to resources by satisfying the security objectives of integrity, availability, and secrecy and for secure the mechanism determining access levels or user privileges related to system resources including files, services, computer programs, data and application features.

• System Maintenance functions intended as maintenance and is intended to be checked on a frequent basis to ensure system stability.

4.2 Proposed Framework Execution

The initiation of the hardening process is started with launching the container from the Container Host. If the framework image does not exist in the local server, then Docker will download the framework image from the Docker Hub, and the container can be formed and run. The hardening process will under the SSH (Secure Shell) protocol. Remote initiation is from Container Framework to the Linux Server by running the command

```
#docker run -it -d --name=linux_harden - -net=host
bimandika/linux_harden
```

All command flag in the above can be adjusted as needed. The most important thing the --net parameter, which purposes connecting the container network segment with the target server network segment. In this case target server has the same segment with container-host, so the --net variable is the container-host network. For other networks, topology can refer to docker networking documentation. The other requirement to be fulfilled is granting authorization access to the Framework on the target server by SSH (Secure Shell). Framework public RSA-key must be planted on the target server. Then target server IP or domain needs to register on /home/hosts file inside the Framework. The framework will read this inventory file before execution hardening. When all requirements are met, to execution hardening the target server can be done by simply use this command

```
#harden [OS]
```

[OS] is a variety of current supported Linux operating system by the Framework currently only support Ubuntu 18.04, CentOS 7, Debian 10, and Fedora 32.

After the hardening process finish, the container can be destroyed to save the storage space of the container host.

4.2 Proposed Framework Evaluation

To verify that design and implementation meet the goals, an experiment was conducted to evaluate Framework. First, Comparing the execution time of frameworks and old methodologies for Hardening on several operating systems and Do the simultaneous execution of Hardening up to 100 Target servers to verify the design goal of parallel execution. Then for security verification, we evaluate target server vulnerability before and after Hardening.

The time consumption of hardening execution is described in Table 3.

Table 3: Comparison of Execution Time

Operating System	Single Server Execution Time		Time-consuming Percentage
	Old Method (s)	Proposed Method (s)	
Ubuntu 18.04	838	293	35%
CentOS 7	557	286	51%
Debian 10	616	195	32%
Fedora 32	370	155	42%

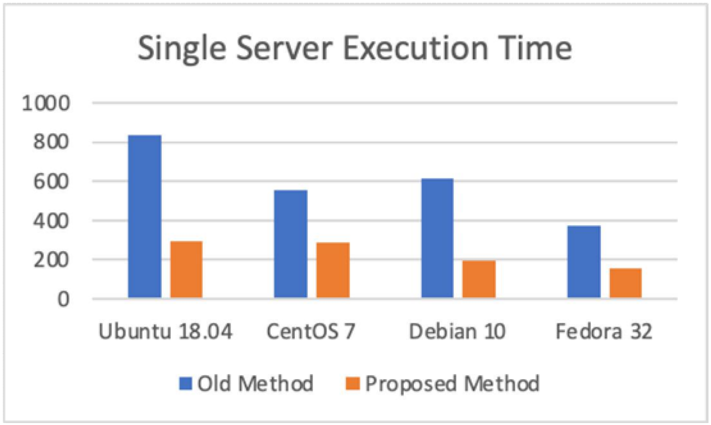

Figure 4: Chart Comparison Execution Time of Single Server

The evaluation process using the same specification of the target server with the same operating system in the same environment. The unit used in this evaluation is second. The execution time needed to harden the Framework is less than 55% on all operating systems, with the average 40% time-consuming. In comparison, the overall Framework has an average of 60% of time optimization than the old method.

While for hardening 100 Server the forecast of time execution can be shown in figure 5

Table 4: Comparison of Parallel Execution time

Number of Servers	Execution Time (s)
10 Server	510
20 Server	609
50 Server	1.313
100 Server	2.834

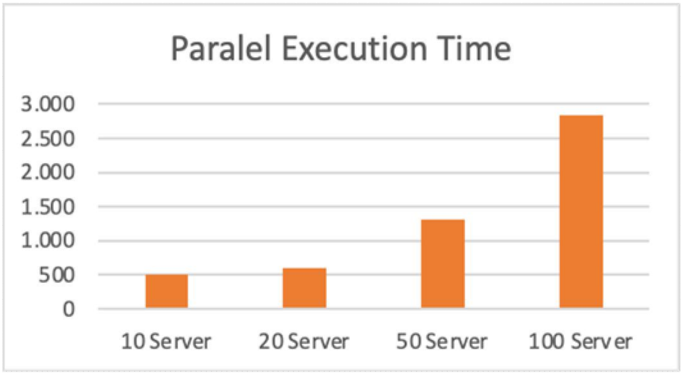

Figure 5: Chart Comparison of Parallel Execution Time

On the parallel execution framework consuming minimum 34 Second per server and maximum 51 second per server and has a better performance than execution on single server. Then in the table below described vulnerability assessment results using OpenVAS before and after the hardening execution

Table 5: Known Vulnerability on Operating System

Operating System	+Vulnerability					
	Before Hardening			After Hardening		
	High	Medium	Low	High	Medium	Low
Ubuntu 18.04	70	84	11	0	1	0
CentOS 7	105	105	8	0	2	0
Debian 10	0	1	1	0	1	0
Fedora 32	2	21	2	0	9	0

Hardening can reduce more than 80% of a known vulnerability, and it enhances the security of the operating system. Less vulnerability means less gateway or weak point that can be used by the attacker to interfere with the system.

5. CONCLUSION AND FUTURE WORK

Cloud Computing will never be utterly secure, especially in Operating System, because the application and system make changes though threats or exploit new applications or packages installed. Hardening can reduce the incorrect values that increase the risk. Therefore, this research describes how to quickly strengthen Linux servers' security by taking advantage of several technologies. Vulnerability

A few enhancements to make this framework work better, compatibility to another operating system, and other security policy will make this framework have a broader scope for driving other operating systems more secure.

REFERENCES

1 K. Sharma, **Emerging Cloud Computing Paradigm: Vision, Research Challenges and Development Trends**, *Int. J. Res. Eng. Technol.*, vol. 03, no. 05, pp. 892–899, 2014, doi: 10.15623/ijret.2014.0305162.

2 J. K. R. Sastry and B. Trinathbasu, **Extended openstack architecture for enforcing comprehensive security within cloud computing system**, *Int. J. Emerg. Trends Eng. Res.*, vol. 8, no. 7, pp. 3271–3279, 2020, doi: 10.30534/ijeter/2020/64872020.

3 B. W. K. Malubaya and G. Wang, **Real-time parking information system with cloud computing open architecture approach**, *Int. J. Emerg. Trends Eng. Res.*, vol. 8, no. 1, pp. 18–22, 2020, doi: 10.30534/ijeter/2020/04812020.

4 H. Pooja and S. Rani, **Avoiding the Security Attacks by Hardening the Cloud**, *International Journal of Emerging Technology and Advanced Engineering w website: www.ijetae.com (ISSN 2250-2459,ISO 9001:2008Certified Journal, Volume 4, Issue 4, April 2014).* 2014.

5 Z. N. Chen *et al.*, **Evolution of Cloud Operating System: From Technology to Ecosystem**, *J. Comput. Sci. Technol.*, vol. 32, no. 2, pp. 224–241, 2017, doi: 10.1007/s11390-017-1717-z.

6 H. M. Musse and L. A. Alamro, **Cloud computing: Architecture and operating system**, *Proceedings - 2016 Global Summit on Computer and Information Technology, GSCIT 2016*. pp. 3–8, 2017, doi: 10.1109/GSCIT.2016.7.

7 D. A. Tevault, *Mastering Linux Security and Hardening Secure your Linux server and protect it from intruders, malware attacks, and other external threats*. Packt Publishing Ltd, 2018, [Online]. Available: www.packtpub.com.

8 A. T.Deshmukh and P. N . Mahalle, **Enhancing Security in Linux OS**, *Int. J. Comput. Appl.*, vol. 117, no. 12, pp. 34–37, 2015, doi: 10.5120/20609-3239.

9 N. Arora, T. Bhosale, V. Sharma, and J. Supe, **Linux Hardening**, *Int. J. Recent Innov. Trends Comput. Commun.*, vol. 2, no. May, pp. 1019–1022, 2014.

10 R. L. Paikrao and V. H. Patil, **Security as a Service Model for Virtualization Vulnerabilities in Cloud Computing**, *2018 International Conference On Advances in Communication and Computing Technology, ICACCT 2018*. pp. 559–562, 2018, doi: 10.1109/ICACCT.2018.8529573.

11 M. Anisetti, C. A. Ardagna, E. Damiani, and F. Gaudenzi, **A Security Benchmark for OpenStack**, *IEEE International Conference on Cloud Computing, CLOUD*, vol. 2017-June. pp. 294–301, 2017, doi: 10.1109/CLOUD.2017.45.

12 S. Rahalkar, *Network vulnerability assessment : identify security loopholes in your network's infrastructure*. Packt Publishing Ltd, 2018, doi: August 31, 2018.

13 L. Bell, M. Brunton-Spall, R. Smith, and J. Bird, *Agile Application Security: Enabling Security in a Continuous Delivery Pipeline*. " O'Reilly Media, Inc.," 2017.

14 A. K. Nepal, **Linux Server & Hardening Security**, no. August, pp. 0–65, 2014, doi: 10.13140/2.1.5079.2329.

15 D. Jaramillo, D. V Nguyen, and R. Smart, **Leveraging microservices architecture by using Docker technology**, *SoutheastCon 2016*. pp. 1–5, 2016.

16 P. P. W. Pathirathna, V. A. I. Ayesha, W. A. T. Imihira, W. M. J. C. Wasala, N. Kodagoda, and E. A. T. D. Edirisinghe, **Security testing as a service with docker containerization**, *International Conference on Software, Knowledge Information, Industrial Management and Applications, SKIMA*, vol. 2017-Decem. pp. 1–7, 2018, doi: 10.1109/SKIMA.2017.8294109.

17 Y. Bhardwaj, **Server Hardening: Securing Unix-like workstations**, *Int. J. Comput. Sci. Eng.*, vol. 3, no. 3, pp. 24–32, 2015.

18 M. U. Aksu, E. Altuncu, and K. Bicakci, **A First Look at the Usability of OpenVAS Vulnerability Scanner**, *Workshop on Usable Security (USEC) 2019*. 2019, doi: 10.14722/usec.2019.23026.

19 S. Rahalkar, **OpenVAS**, in *Quick Start Guide to Penetration Testing*, Springer, 2019, pp. 47–71.

20 O. Alhazmi, Y. Malaiya, and I. Ray, **Security vulnerabilities in software systems: A quantitative perspective**, *IFIP Annual Conference on Data and Applications Security and Privacy*. pp. 281–294, 2005.

21 A. Gorbenko, A. Romanovsky, O. Tarasyuk, and O. Biloborodov, **From Analyzing Operating System Vulnerabilities to Designing Multiversion Intrusion-Tolerant Architectures**, *IEEE Trans. Reliab.*, vol. 69, no. 1, pp. 22–39, 2020, doi: 10.1109/TR.2019.2897248.

22 T. Bui, **Analysis of Docker Security**, *arXiv Prepr. arXiv1501.02967*, 2015, [Online]. Available: http://arxiv.org/abs/1501.02967.

23 A. Mouat, *Using Docker: Developing and Deploying Software with Containers, O'Reilly*. " O'Reilly Media, Inc.," 2016.

24 S. Navin and W. Bibin, *Docker Hands on: Deploy, Administer Docker Platform*. Amazon Digital Services, 2015.

25 A. Anwar *et al.*, **Improving Docker Registry Design based on Production Workload Analysis**, *Proceedings of the 16th USENIX Conference on File and Storage Technologies, FAST 2018*. pp. 265–278, 2018.

26 D. Merkel, **Docker: lightweight Linux containers for consistent development and deployment**, *Linux J.*, vol. 2014, no. 239, p. 2, 2014, doi: 10.1097/01.NND.0000320699.47006.a3.

27 A. Thakur, *Docker, Creating Structured Containers*. Packt Publishing Ltd, 2016.

28 C. Boettiger, **An introduction to Docker for reproducible research**, *Oper. Syst. Rev.*, vol. 49, no. 1, pp. 71–79, 2015, doi: 10.1145/2723872.2723882.