Automated Hardening of a Linux Web Server

Olenčin, Michal and Perháč, Ján

Abstract: ***This article is dedicated to the design and implementation of automated hardening a Linux web server after the clean installation. The article deals with the analysis of the cybersecurity, focus, and scope of the configuration. Based on that, was made a design of an automated security configuration with a detailed description. The main accomplishment of the work is a configuration script. The script is compared and evaluated with existing solutions and it achieved the best rating in security audits.***

Index Terms: ***cybersecurity, Linux security, Web security***

1. Introduction

THE PAPER is oriented on designing an automated hardening of a Linux web server and subsequently on implementing and evaluation of the designed solution. At first, a motivation for creating this paper is specified, afterward the current state of cybersecurity is analyzed. Then a focus of security and analysis is specified. The results of the analysis are taken into account in proposing a design of our configuration according to a wide range of resources.

The output of the implementation is an open-source script [12], [13] of an automated configuration implemented according to the proposed design. Finally, an evaluation of the implemented configuration was performed. That evaluation consists of a functional test, a compatibility test of the webserver configuration, and security audits of default and designed configurations. Also security audit by the Lynis tool was performed and also a score of existing open-source solutions measured and compared.

Our recent research in the field of computer security was focused on intrusion detection systems [10], [14], its formal description, and developing their models [11], [15] and securing the Linux web servers. In this paper, we are focused on the results of our recent work with the automated securing of the Linux webserver. We start with a brief description of the current state in the area. We compare existing solutions, and analyze technological possibilities. Then, we describe the details of the designs and our implementation. in the final section, we evaluate our implementation. We compare it with other popular solutions in the field by performing standard security audits.

Manuscript received May 2, 2020.

This work was supported by the Faculty of Electrical Engineering and Informatics at the Technical University of Košice under contract No. FEI-2020-70: "Behavioral model of component systems based on coalgebras and linear logic", and by the project KEGA 011TUKE-4/2020: "A development of the new semantic technologies in educating of young IT experts".

M. Olenčin (contact person) is an MSc student at the Faculty of Electrical Engineering and Informatics, Technical University of Košice, Slovakia (e-mail: michal@olencin.com).

J. Perháč (contact person) is an assistant professor at the Faculty of Electrical Engineering and Informatics, Technical University of Košice, Slovakia (e-mail: jan.perhac@tuke.sk).

2. Motivation

The cybersecurity and protection of sensitive data are frequently discussed topics today. There is a growing interest in this area mainly due to the increase in cyberattacks and theft of sensitive data.

After installation of a web server, a basic configuration is used and it is necessary to configure it as needed. The basic server setup includes many security issues that need to be corrected by an appropriate configuration for security reasons. The basic configurations are customized to maximize support and make development easier. The configuration process can be automated using programs written in a scripting language. Creating an automated configuration makes the process of configuration quicker and more effective.

Currently, there are very few solutions dealing with the automation of a Linux web server configuration in the desired complexity and with adequate protection. The main goal of this work is to create a solution on an appropriate level of protection and complexity by creating an automated configuration of a Linux web server.

3. The current state of cybersecurity

We have defined the current state of cybersecurity by summarizing the most interesting results from following reports:

- *"Cisco 2018 Annual Cybersecurity Report"* by Cisco [3],
- *"Executive Summary - 2018 Internet Security Threat Report"* by Symantec [18],
- *"Cloud Security Report"* by Alert Logic [9].

The most interesting results e.g. occurrences of malicious programs dedicated to various areas increased in the years from 2015 to 2017 as following:

- cryptocurrencies by 8500%,
- IOT devices by 600%,

- ransomware attacks by 46%,
- malware on mobile devices by 54%.

3.1. Used technologies

In the first step, we have analyzed the usage of current technologies. We have taken into account the following criteria:

- a Linux distribution,
- a web service,
- database service,
- programming language.

Our main goal was to create a configuration for a wide group of potential users, therefore our main criterion was the popularity of the technology.

Because of long-term support and the high market share, we have chosen an Ubuntu Server as the supported distribution. Version Ubuntu Server 16.04.5 64-bit has been chosen, because of its stability. Ubuntu has 37.8% market share altogether. If the Debian distribution had been added as a supported distribution, the solution would have covered 61.1% of the whole market.

We have chosen an Apache webserver as a support web server, because of its 45.5% market share in its latest version Apache 2.4. The choice of the Nginx as a supported web server was also taken into account, but based on the balance between quantity and quality, Apache won. Both are comparable in performance, flexibility, and reliability [7]. If the Nginx web server had been added as a support web server, the solution would have covered ¾ of the market share.

As the support database service, MariaDB was chosen in its latest version of a database server, specifically MariaDB 10.3. The selections of MySQL and PostgreSQL as the support database server were also taken into account. PostgreSQL is robust, has a complex configuration, but is less powerful compared to others. MariaDB is a fork of MySQL, which is 5% more powerful [6] than MySQL. Also, its popularity tends to increase. But both, MySQL and PostgreSQL, are efficient and reliable database services and their support could be added in the future.

PHP was chosen as the support programming language in the latest version, because of its dominance on the market with 78.9%.

Results are summarized in the table 1 *"Result of selected security focus"* covering a wide range of usability.

TABLE 1: Result of selected security focus

Scope	**Focus**
Linux distribution	Ubuntu Server 16.04.5 64-bit
Web service	Apache 2.4
Database service	MariaDB 10.3
Programming language	PHP 7.2

3.2. Existing Solutions

There are several automated Web Linux server configuration solutions. Unfortunately, most solutions are not complex and only address the configuration of the Apache web service itself, or address a small configuration range. The solutions with the required complexity currently rarely exist.

Among the proprietary license solutions, the Lynis Enterprise solution is available [4]. Unfortunately, this solution is too complex for common needs and requires the purchase of a license, which is a disadvantage. This solution offers security audit, system security, vulnerability management, ample support for Unix operating systems, macOS and many more.

Among the open-source solutions, the *"JShielder"* [17] and the *"Ubuntu hardening"* [16] solutions are available. Both solutions are not overwriting existing configurations and do not use the latest versions of services. Ubuntu hardening solution also includes a smaller range of configurations compared to the JShielder solution.

Each of the mentioned solutions is implemented as a bash script.

While analyzing existing solutions, deficiencies have been identified that proposed solution seeks to eliminate. The proposed solution compared to existing solutions includes:

- configuration backup,
- use of the latest versions of services,
- it overwrites existing configurations,

with open-source availability, with the required complexity and configuration range for basic needs.

4. Design of our solution

The design of our solution is proposed according to the CIS Ubuntu Linux 16.04 LTS Benchmark [1], the CIS Distribution Independent Linux Benchmark [5], the CIS Apache HTTP Server 2.4 Benchmark [2], the book *"Practical Apache, PHP-FPM & Nginx Reverse Proxy: How to Build a Secure, Fast and Powerful Webserver from scratch"* [8], the configuration instructions from the non-profit organization Mozilla, the proposed Lynis configuration, the proposed Tiger configuration and the recommended set of rules for security2 module from the non-profit organization OWASP.

The design is targeted to achieve the highest possible security for common web servers and includes:

- Installation and hardening of the Apache web server and also security2, SSL, evasive, headers, include, http2 and reqtimeout modules.
- Installation and hardening of the MariaDB database server.

- Installation and hardening of the PHP programming language.
- Installation and hardening of the OpenSSH server.
- Security configuration of an operation system:
  - Hardening of directory permissions.
  - Hardening of the file system.
  - Hardening of network protocols.
  - Hardening of the kernel.
  - Hardening of passwords.
  - Hardening of the firewall.
  - Hardening of other areas.

The design of configuration for the Apache webserver includes adding sources with the latest stable version of a package manager, turning off unnecessary modules, limiting leaked information through web server response, setting up the service to run under a separate user, configuring security modules, request limits, logging level, permission rights for the service files, etc.

Among unnecessary modules that are `webdav`, `status`, `autoindex`, `proxy`, `userdir` and `info`.

Among security modules that are `security2`, `ssl`, `evasive`, `headers`, `include`, `http2` and `reqtimeout`.

The tables 2 and 3 presents our configuration of the Apache in detail.

TABLE 2: Configuration of the Apache

Parameter	Value
DocumentRoot	/var/www/html/
FileETag	None
Timeout	10
LimitRequestBody	102400
LimitRequestFieldsize	1024
LimitRequestline	512
LogLevel	notice core:info
Protocols	h2 http/1.1
ServerTokens	Prod
TraceEnable	off

TABLE 3: Configuration of the SSL module

Parameter	Value
SSLCipherSuite	*Based on SSL generator*
SSLCompression	off
SSLHonorCipherOrder	on
SSLInsecure-Renegotiation	off
SSLProtocol	all -SSLv3 -TLSv1 -TLSv1.1
SSLSessionTickets	off
SSLStaplingCache	"shmcb:/var/run/ocsp(128000)"
SSLStapling-ResponderTimeout	5
SSLStaplingReturn-ResponderErrors	off
SSLUseStapling	on

Next, the configuration of the MariaDB database server includes adding a source with the latest stable version of a package manager, running the built-in secure installation, preventing access to services outside the local network and access to local files through the database and configuring the port. The MariaDB database server does not require complex hardening configuration in general.

The table 4 presents our configuration of the MariaDB in detail.

TABLE 4: Configuration of the MariaDB

Parameter	Value
bind-address	127.0.0.1
general-log	0
local-infile	0
port	*Depends on input*

Furthermore, the configuration of the PHP programming language includes also adding a source with the latest stable version of a package manager, disabling a debug and error messages, also disabling dangerous functions, limiting leaked information through web service, configuring session, cookies, and the limit of access to system files, etc.

Among dangerous functions that are for example process control functions, functions of the POSIX standard, program execution functions, functions that provide sensitive information and much more.

The table 5 presents our configuration of the PHP in detail.

TABLE 5: Configuration of the PHP

Parameter	Value
allow_url_fopen	0
allow_url_include	0
disable_functions	*Depends on design of configuration*
display_errors	0
display_startup_errors	0
expose_php	0
html_errors	0
log_errors	1
open_basedir	/var/www/html
post_max_size	100K
session.use_strict_mode	1
session.cookie_httponly	1
session.cookie_secure	1

Configuration of the OpenSSH (version 7.2) server includes installation of the package with a model for server, limiting leaked information through services, port configuration, configuring user authorization, logging level, authorization message, cryptography algorithms, enabling only public key authentication and much more. Upon public key authentication, i.e. a key pair authentication is important to have a private key secured against theft. For maximum security of private key can be used universal 2nd factor *(U2F)*, one-time password *(OTP)* or another form of multi-factor authentication *(MFA)* can be used.

Tables 6, and 7 presents our configuration of the OpenSSH in detail.

Operation system configuration consists of the following:

TABLE 6: Configuration of the OpenSSH authentication

Parameter	Value
ChallengeResponseAuthentication	no
GSSAPIAuthentication	no
HostbasedAuthentication	no
PasswordAuthentication	no
PubkeyAuthentication	yes
RhostsRSAAuthentication	no
RSAAuthentication	no

TABLE 7: Configuration of the OpenSSH

Parameter	Value
Banner	/etc/issue.net
AllowAgentForwarding	no
AllowGroups	ssh
AllowTcpForwarding	no
Ciphers	*Depends on design*
ClientAliveInterval	300
Compression	no
DebianBanner	no
IgnoreRhosts	yes
KexAlgorithms	*Depends on design*
LoginGraceTime	60
LogLevel	VERBOSE
MACs	*Depends on design*
MaxAuthTries	2
MaxSessions	2
PermitRootLogin	no
Port	*Depends on input*
Protocol	2
TCPKeepAlive	no
UseDNS	no
UsePrivilegeSeparation	SANDBOX
X11Forwarding	no

- Hardening of directory permissions includes the configuration of sticky bit on all world-writable directories, set of basic permissions rights and configuring the permissions for the sensitive system files.
- The file system is in this way configured so that it includes disabling some unnecessary files systems, configuring USB devices and mounting partitions.
  - Among unnecessary files systems are the following: `cramfs`, `squashfs`, `freevxfs`, `jffs2`, `hfs`, `hfsplus` and `hfsplus`.
- The hardening of network protocols includes disabling some unnecessary network protocols.
  - Among unnecessary network protocols that are the following: DCCP, SCTP, RDS and TIPC.
- The hardening of the kernel consists of enabling address space layout randomization, disabling SysRq key, configuring packets and so on.
- The hardening of other areas includes uninstalling unnecessary packages, installing additional security packages and so forth.
  - The following packages belong to the additional security packages included: acct, aide, aide-common, apt-show-versions, arpwatch, auditd, clamav, clamav-daemon, debsums, fail2ban, htop, ntp, sysstat, unattended-upgrades and usbguard.
  - The designed configuration also configures some additional security packages, e.g. auditd, unattended-upgrades and usbguard.
  - Among unnecessary packages that are the following: avahi-daemon, binutils, cups, cups-common, gcc, git, make, snapd and telnet.

The designed configuration, however, might be incompatible with some web applications and websites. This is caused by individual requirements for the configurations of those applications and websites. Additional configuration adjustments are needed after the automated configuration is applied.

5. Implementation and evaluation of our solution

Our solution is implemented in the form of a bash script, and it is implemented according to the designed configurations. The implementation also includes a password generator. This generator is useful when SSH key is generated and the password for MariaDB root account is set up. Further, the whole source code is commented for easy understanding and orientation. In the implementation, the port for OpenSSH and MariaDB is obtained from the user. If the user presses the enter key, the default port is set.

Furthermore, the evaluation was done on a virtual private server for the most realistic conditions and also on a virtual machine. As per functional testing, the script is fully functional on Ubuntu Server 16.04.5 distribution, but also in its newer version 18.04.2. Later we have tested our solution on the newest version for the latest non-lts version Ubuntu 19.04. Our solution for this version is also fully functional. We have selected the Oracle VM VirtualBox as a hosted hypervisor for a virtual machine. For a cloud infrastructure, the DigitalOcean provider was selected.

After the compatibility test of the web server configuration, we have found out that the designed configuration requires small additional configuration adjustments depending on a web application or a website. The compatibility test was done on both types of web pages, with static and also dynamic contents.

Security audits were performed using Lynis tool, Nmap tool, SSL Server Test by the Qualys, Mozilla HTTP Observatory tool and Mozilla SSH Observatory tool. Results of security audits are summarized in the table 8 *“Result of all security audits in a designed configuration”* and in the table 9 *“Result of all security audits in a default configuration”*.

TABLE 8: Result of all security audits in a designed configuration

Security audit	Result
Lynis	97 / 100
Nmap	a few
SSL Server Test	A+
Mozilla HTTP	A+
Mozilla SSH	A

TABLE 9: Result of all security audits in a default configuration

Security audit	Result
Lynis	56 / 100
Nmap	a lot
SSL Server Test	B
Mozilla HTTP	F
Mozilla SSH	C

The Nmap tool audit results are rated based on found leaking information without any grading scale. The SSL Server Test audit gives grades A-, B, C, D, E, F *(sorted from the best to the worst)* or T *(if found some trust issues)*. The Mozilla HTTP audit gives grades A+, A, A-, B+, B, B-, C+, C, C-, D+, D, or D- *(sorted from the best to the worst)*. The Mozilla SSH audit gives grades A, B, C, D, or F *(sorted from the best to the worst)*.

Within the security audit using the Lynis we also measured a score of existing open-source solutions. These measurements have been adapted to achieve as highest score as possible. The Lynis security audit was performed with version 2.7.1 of the tool. We have compared our solution with other existing configurations, and the default configuration with the following results:

- The default configuration has achieved a score of **56** / **100**.
- The Ubuntu hardening solution has achieved a score of **72** / **100**.
- The JShielder solution has achieved a score of **89** / **100**.
- Our solution of configuration has achieved a score of **97** / **100**, which is the best rating when compared to existing solutions.

Unfortunately, the designed configuration did not achieve full score because of the absence of GRUB password for booting, presence of an antivirus program and separate mounting of partitions `/tmp`, `/home` and `/var`. Creating a GRUB password is unwanted on remote computers. Configuring separate mounting of partitions was performed during the installation of the operating system. Additional configuration of mounting partitions is complicated. In the designed configuration a ClamAV as antivirus is installed, but the security audit by the Lynis tool did not detect it.

The security audit for the Nmap tool detected much less sensitive information leaked for the designed configuration compared to the default configuration, such as information of server distribution and a version of Apache.

The security audit performed by the SSL Server Test scored on the designed configuration an **A+** rating. The default configuration achieved a **B** rating for using weak cryptography algorithms and not providing forward secrecy to all browsers. With the security audit done by the Mozilla HTTP Observatory tool the designed configuration has achieved an **A+** rating. The default configuration achieved an **F** rating for missing security-related web service response headers. In the security audit by the Mozilla SSH Observatory tool, the designed configuration has achieved an **A** rating. The default configuration achieved a **C** rating for using weak cryptography algorithms.

Results of security audits are summarized in table 10 *"Result of Lynis security audit"*.

TABLE 10: Result of Lynis security audit

Solution	Score
Implemented solution	97
JShielder	89
Ubuntu hardening	72
Default configuration	56

To summarize the implemented solution has the following advantages compared to other existing solutions:

- The design is drawn up according to the use of a large number of resources. For instance, CIS benchmarks guidelines, the proposed Lynis configuration, etc.
- The implemented solution is designed with the required complexity and configuration range for common needs.
- The implemented solution is free and open-source.
- The design uses the latest versions of services.
- The design overwrites existing configurations.
- The design includes configuration backup.

6. Conclusion

In this work, we analyzed the current state of cybersecurity, and we concluded that is always necessary to address cyber security due to the increasing incidence of cyber attacks.

In the analysis of existing solutions, deficiencies have been identified that have been eliminated by the proposed solution. Our solution has following properties:

- it is an open-source solution,
- it creates a backup of existing configurations,
- it uses the latest service versions,
- it overwrites existing configurations,

- it is designed with the required complexity and scope of configuration for common needs included.

Based on the analysis, the design of the solution was made by using recommended configurations from the Lynis audit and guidelines from the non-profit organization CIS. Besides that, we used also recommended instructions from the non-profit organization Mozilla, recommended configurations from the Tiger audit, recommended set of rules for `security2` module from the non-profit organization OWASP and recommended configuration guidelines from the book *"Practical Apache, PHP-FPM & Nginx Reverse Proxy"* [8].

In conclusion, the result of this work is a fully functional configuration script, which in comparison to existing solutions achieved the best rating in Lynis security audits.

The script is easily expandable and editable, covers a large scope of applications and utilities by market share, comprehensively solves the security of the webserver and the operating system.

On another hand, the script requires additional configuration, requires a knowledge of Linux configuration and partially rewriting the configurations of the Apache web service.

The work may be extended by including the compatibility for Debian distribution, including the Nginx web service support and including support for other database services such as MySQL and PostgreSQL.

However, the used security analysis tools used do not cover the full range of potential vulnerabilities. The security of an entity depends on the weakest element. The designed solution has improved web server security against known vulnerabilities, but vulnerability to unknown errors cannot be eliminated. In addressing the security problem, care must be taken to continually improve and mend the security of existing solutions, monitor these solutions and maintain cyber hygiene.

References

[1] CIS. Cis ubuntu linux 16.04 lts benchmark, December 2017.
[2] CIS. Cis apache http server 2.4 benchmark, July 2018.
[3] Cisco. Cisco 2018 annual cybersecurity report, February 2018.
[4] CISOfy. Lynis enterprise, 2019.
[5] CIS. Cis distribution independent linux benchmark, December 2017.
[6] Russel J.T. Dyer. *Learning MySQL and MariaDB*. O'Reilly Media, 1 edition, April 2015.
[7] Douglas Kunda, Sipiwe Chihana, and Muwanei Sinyinda. Web server performance of apache and nginx: A systematic literature review. *IISTE*, 8(2), March 2017.
[8] Adrian Ling. *Practical Apache, PHP-FPM & Nginx Reverse Proxy: How to Build a Secure, Fast and Powerful Webserver from scratch*. Adrain Ling Kong Heng, 1 edition, May 2015.
[9] Alert Logic. Cloud security report, 2017.
[10] Daniel Mihályi and Valerie Novitzká. A coalgebra as an intrusion detection system. *Acta Polytechnica Hungarica*, 7(2):71–79, 2010.
[11] Daniel Mihályi and Valerie Novitzká. Towards to the knowledge in coalgebraic model ids. *Computing and Informatics*, 33(1):61–78, 2014.
[12] Michal Olenčin. Alfavio/Automated-securing-of-Linux-web-server v.1.0.0, June 2019.
[13] Michal Olenčin and Ján Perháč. Automated configuration of a linux web server security. In *2019 IEEE 15th International Scientific Conference on Informatics*, pages 304–308. IEEE, 2019.
[14] Ján Perháč, Daniel Mihalyi, and Lukaš Mat'aš. Resource oriented bdi architecture for ids. In *2017 IEEE 14th International Scientific Conference on Informatics*, pages 293–298. IEEE, 2017.
[15] Ján Perháč and Daniel Mihályi. Coalgebraic modeling of ids behavior. In *2015 IEEE 13th International Scientific Conference on Informatics*, pages 201–205, November 18-20, 2015, Poprad, Slovakia, Danvers: IEEE, 2015.
[16] Joel Radon. Ubuntu hardening, May 2019.
[17] Jason Soto. Jshielder, March 2019.
[18] Symantec. Executive summary - 2018 internet security threat report, March 2018.

Michal Olenčin received his bachelor's degree in Informatics from the Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Košice, Slovak Republic in 2019. His research interests include cybersecurity, computer security, security of operating systems.

Ján Perháč received his PhD. in Informatics from the Technical University of Košice, in 2019. Currently, he is an assistant professor at the Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Slovakia. His research topics include computer security, category theory, logical systems, type theory, and semantics of programming languages.