International Journal of Computational Intelligence & Digital Systems
Volume 1, Number 1, 2006
P-ISSN: xxxx-xxxx E-ISSN : xxxx-xxxx
Open Access: https://journal.nexuspublishing.co.id/ijcd

Implementation and Security Evaluation of The SIMETRI web-based Information System Using LAMP Stack and SSL/TLS on a Debian Server

Rachmad Hidayat[1]
[1] Pengadilan Negeri Curup
[2] Universitas Pat Petulai
[3] Universitas Terbuka
Correspondence: rachmaddt@hotmail.com

Article Info
Article history:
Received Jun 12th, 20xx
Revised Aug 20th, 20xx
Accepted Aug 26th, 20xx

Keyword:
SIMETRI; LAMP Stack; SSL/TLS; Debian Server; Web Security; Judicial Information System

ABSTRACT
Secure and reliable digital infrastructure is essential for electronic-based public services, particularly in judicial institutions that manage sensitive administrative and legal data. This study aimed to implement and evaluate the SIMETRI web-based information system using a LAMP Stack architecture consisting of Debian 11, Apache HTTP Server 2.4, MariaDB 10.5, and PHP 5.6, combined with SSL/TLS-based communication security. An implementation-based case study was conducted at Pengadilan Negeri Curup. System evaluation included functional testing, ApacheBench performance testing, SSL/TLS validation, server resource monitoring, and security auditing using Lynis. The system operated successfully over HTTPS using TLS 1.3. Performance testing recorded 186.34 requests per second, 268 ms time per request, a 4.2 MB/s transfer rate, and zero failed requests. All principal SIMETRI modules operated properly, while the security assessment produced good to very good results for kernel hardening, file permissions, network security, web server security, and cryptography. The results demonstrate that the configuration is feasible for a secure on-premise judicial information system under the tested internal workload. The study also identifies the legacy PHP runtime, self-signed certificate, and absence of a dedicated Web Application Firewall as risks requiring phased mitigation.

INTRODUCTION

Digital transformation in public-sector institutions has become a strategic requirement for improving administrative efficiency, service transparency, institutional accountability, and information governance. In judicial institutions, this transformation is especially consequential because information systems support not only routine administration but also legal documents, internal correspondence, case-related information, electronic archives, reporting processes, and public-service workflows. The infrastructure supporting these activities must therefore provide confidentiality, integrity, availability, accountability, and continuity rather than merely making an application accessible.

One initiative implemented in the judicial environment is the Integrated Management Information System, known as SIMETRI. SIMETRI supports administrative functions within the court environment. At Pengadilan Negeri Curup, the system is used for electronic document management, incoming and outgoing correspondence, internal document classification, electronic-signature workflows, guest-book management, legal-aid service administration, and application configuration. Its operational value depends on the reliability of the server platform, database service, network path, and controls surrounding user access.

Systems-administration literature emphasizes that dependable services require more than software installation. They require explicit service ownership, reproducible configuration, separation of duties, least-privilege access, monitoring, backup, change control, and documented recovery procedures

(Limoncelli et al., 2017; Nemeth et al., 2018). At the operating-system level, process isolation, memory management, storage management, account permissions, and controlled resource allocation provide the foundation on which application reliability is built (Silberschatz et al., 2018). These principles are directly relevant to an on-premise court server because a single configuration error or unmanaged dependency can affect both service availability and the integrity of institutional records.

The LAMP Stack, consisting of Linux, Apache, MariaDB or MySQL, and PHP, remains widely used for web-based systems because it is modular, cost-efficient, and supported by a mature operational ecosystem. Apache HTTP Server provides a configurable application-delivery layer, while MariaDB supplies relational data management for structured application records. Database theory highlights the importance of controlled authorization, consistency, recovery, and integrity constraints in protecting organizational data (Elmasri & Navathe, 2022). Linux administration references similarly recommend clear file-system organization, minimal service exposure, patch management, logging, and tested backup procedures as core production practices (Hertzog & Mas, 2022; Nemeth et al., 2018).

Debian 11 was selected because it provides a stable server environment and supports the hardware compatibility requirements of the HP ProLiant DL320e Gen8 V2. Apache HTTP Server 2.4 and MariaDB 10.5 were selected as the web and database services. PHP 5.6 was retained because the current SIMETRI codebase depends on its runtime behavior and extensions. However, legacy compatibility creates a security-management problem: an unsupported runtime must be treated as technical debt and surrounded by compensating controls while a migration path is prepared. Security engineering literature advises reducing the trust placed in obsolete components through segmentation, restrictive access, monitoring, and layered controls rather than assuming that a perimeter control alone will neutralize the risk (Anderson, 2020; Stallings & Brown, 2018).

Security is particularly important for judicial information systems because the data may include sensitive administrative records, legal documents, credentials, and institutional correspondence. In a client-server web architecture, application traffic traverses network components where delay, loss, interception, and manipulation are possible; security and performance must therefore be considered across the application and transport layers (Kurose & Ross, 2021; Tanenbaum & Wetherall, 2021). SSL/TLS protects HTTP communication by establishing an authenticated and encrypted channel. TLS 1.3 reduces obsolete protocol options and supports modern authenticated encryption, but secure deployment still depends on certificate trust, protocol selection, key management, server configuration, and correct redirection behavior (Rescorla, 2018; Ristić, 2022; Stallings, 2022).

The broader security objective is defense in depth. A secure channel protects data in transit, but it does not replace host hardening, database restrictions, secure application code, logging, backup, or vulnerability management. Anderson (2020) and Stallings and Brown (2018) describe dependable security as a property of the complete system, including people, processes, configuration, and failure assumptions. This perspective is important for SIMETRI because the use of HTTPS can reduce network exposure while risks in the legacy PHP application, file permissions, user accounts, or database queries remain.

Performance evaluation also requires a defined workload and cautious interpretation. Jain (1991) explains that a meaningful computer-system evaluation should identify the system under test, workload, factors, metrics, and experimental limitations. Accordingly, the ApacheBench figures in this study are interpreted as evidence for the tested configuration and workload, not as a universal capacity rating for every future usage pattern. Response time and throughput are complementary: increasing concurrency can improve aggregate throughput while also increasing waiting time when server or network resources approach saturation (Jain, 1991; Kurose & Ross, 2021).

Previous technical studies and operational guidance indicate that open-source server architectures can support government information systems when configuration, maintenance, and evaluation are systematically managed. Nevertheless, implementation-based studies in judicial institutions remain limited, especially studies addressing physical on-premise deployment, legacy runtime compatibility, encrypted communication, performance measurement, and host-level security auditing within one case. The novelty of this study is the integrated evaluation of SIMETRI on a physical Debian server using functional, performance, transport-security, resource-utilization, and hardening evidence.

This study aimed to implement and evaluate the SIMETRI web-based information system using LAMP Stack and SSL/TLS on a Debian server. The evaluation focused on five aspects: functional

validity, web-server performance, SSL/TLS communication security, server-resource utilization, and system-security auditing. The findings are intended as a practical reference for judicial institutions and other public agencies considering secure and maintainable on-premise web-system deployment.

RESEARCH METHODS

This study used an implementation-based case-study approach. The object of the study was the deployment of the SIMETRI web-based information system at Pengadilan Negeri Curup. The case-study approach was selected because the research examined a real server environment, including hardware preparation, software configuration, application integration, validation, and operational evaluation. The unit of analysis was the deployed SIMETRI service and its supporting server stack.

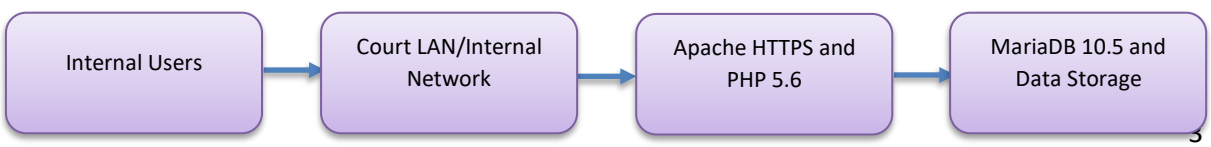
The system was deployed on an HP ProLiant DL320e Gen8 V2 server. The on-premise server hosted web services, database services, server-side scripting, application files, and HTTPS communication. The environment consisted of Debian 11 Bullseye 64-bit, Apache HTTP Server 2.4, MariaDB 10.5, PHP 5.6, and OpenSSL 1.1.1k. The environment is summarized in Table 1.

Table 1 Environment of System Implementation

Component	Specification
Server model	HP ProLiant DL320e Gen8 V2
Processor	Intel Xeon E3-1240 V2, 3.4 GHz, 4 cores, 8 threads
Memory	8 GB RAM
Storage	1 x 300 GB RAID 0 and 1 x 1 TB RAID 0
Network	2 x Gigabit Ethernet ports
Operating system	Debian 11 Bullseye 64-bit
Web server	Apache HTTP Server 2.4
Database server	MariaDB 10.5
Programming language	PHP 5.6
Security component	OpenSSL 1.1.1k
Hosted application	SIMETRI

The implementation process consisted of four stages. First, hardware and operating-system preparation included BIOS and firmware updates, RAID configuration, and Debian installation. Second, the LAMP components were installed and configured. Apache used a dedicated virtual host, MariaDB was subjected to basic hardening, and the required PHP 5.6 modules were enabled for application compatibility. Third, the Apache SSL module and digital certificate were configured, HTTPS virtual-host settings were applied, and HTTP requests were redirected to HTTPS. Fourth, SIMETRI was placed in the designated web-root directory and connected to the database environment.

The architecture in Figure 1 represents the principal trust and service boundaries. Internal users access SIMETRI through the court local-area network. Apache terminates HTTPS and invokes PHP application logic, while MariaDB stores structured application data on the same on-premise server environment. This arrangement minimizes external exposure, but it also means that host-level compromise could affect multiple service layers; consequently, local permissions, service binding, logging, and backup remain important controls (Anderson, 2020; Nemeth et al., 2018).



HP ProLiant DL320e V2 – Debian 11.11 on-premise server environment

Picture 1 Deployment Architecture of SIMETRI Server

System evaluation used five methods. Functional testing covered user login, the administrative dashboard, incoming and outgoing correspondence, internal documents, electronic signatures, the guest book, legal-aid services, and application configuration. Each module was accessed through HTTPS and evaluated according to whether its principal operation completed successfully.

Performance testing used ApacheBench with 1,000 total requests and 50 concurrent connections. The recorded metrics were requests per second, time per request, transfer rate, and failed requests. Following the workload-specification principle described by Jain (1991), the benchmark parameters are reported explicitly so that the result can be interpreted within its test boundary. The test was a synthetic point-in-time benchmark on one configuration; it was not designed as a long-duration endurance test, capacity-planning forecast, or substitute for production workload traces.

SSL/TLS validation used the OpenSSL client to confirm protocol negotiation and encrypted communication. Validation focused on HTTPS availability, TLS 1.3 activation, certificate presentation, and HTTP-to-HTTPS redirection. Ristić (2022) notes that protocol activation should be assessed together with certificate trust and deployment configuration; therefore, the self-signed certificate was treated as acceptable only for a controlled internal environment and was recorded as a risk for broader use.

Resource monitoring observed CPU, RAM, disk I/O, and load average during normal operation and testing. The purpose was to identify obvious resource saturation or bottlenecks, not to derive a complete queueing model. Systems-administration literature recommends establishing operational baselines and reviewing deviations over time; the observations in this study constitute an initial baseline rather than continuous capacity monitoring (Limoncelli et al., 2017).

Security auditing used Lynis to review kernel hardening, file permissions, network security, web-server security, and cryptography. The audit was interpreted as a host-hardening assessment and not as proof that the web application was free from SQL injection, cross-site scripting, authentication flaws, or business-logic vulnerabilities. This distinction follows the defense-in-depth principle that different controls and test methods address different failure modes (Anderson, 2020; Stallings & Brown, 2018).

The analysis was descriptive-quantitative. Quantitative evidence consisted of benchmark values, response-time observations, and security scores. Qualitative evidence consisted of configuration validation and identified risks. The outcomes were compared with the technical objectives: stable web-service operation, successful database integration, active encrypted communication, no failed requests in the specified benchmark, functioning application modules, and identifiable risks accompanied by mitigation recommendations.

RESULTS AND DISCUSSION

The implementation of SIMETRI using the LAMP Stack on Debian 11 was completed successfully. Apache functioned as the client-facing web layer, MariaDB stored structured application data, and PHP 5.6 executed server-side application logic. Debian provided the operating-system environment required by the HP ProLiant server and the application dependencies. From an operating-system perspective, the deployment combined process execution, file-system permissions, memory use, storage access, and network services; these interacting resources explain why application reliability depends on both configuration correctness and continuing system administration (Silberschatz et al., 2018).

Apache HTTP Server 2.4 used a dedicated virtual host for SIMETRI. The web-root directory was placed on a dedicated data partition, separating application data from core operating-system files. This separation can simplify backup selection, storage monitoring, permission review, recovery, and troubleshooting. It is consistent with administration guidance that encourages predictable file-system organization, controlled ownership, and documented service configuration (Hertzog & Mas, 2022;

Nemeth et al., 2018). HTTP-to-HTTPS redirection was also enabled so that users were directed to the encrypted endpoint.

MariaDB 10.5 was integrated with PHP 5.6. Basic database hardening removed anonymous users, disabled remote root login, removed the test database, and reloaded privilege tables. Database access was limited to the local server. These measures reduce unnecessary exposure and align with the principles of least privilege and controlled authorization. They do not, however, eliminate application-layer database risks: secure query construction, input validation, account separation, backup, and recovery testing remain necessary for preserving integrity and availability (Anderson, 2020; Elmasri & Navathe, 2022).

PHP 5.6 was loaded successfully with the modules required by SIMETRI. Compatibility was achieved, but the runtime has reached end-of-life and no longer receives normal upstream security maintenance. The operational decision should therefore be treated as a temporary compatibility exception. Good administration practice requires documenting the exception, restricting exposure, monitoring changes, maintaining recovery options, and creating a tested migration plan rather than allowing a legacy dependency to become permanent by default (Limoncelli et al., 2017; Stallings & Brown, 2018).

SSL/TLS was successfully enabled through Apache. The service was accessible using HTTPS, OpenSSL validation showed TLS 1.3 negotiation, and unencrypted HTTP access was redirected. These controls protect confidentiality and integrity while data travels between users and the server. The result is consistent with TLS deployment guidance emphasizing modern protocols and encrypted-by-default access (Ristić, 2022; Stallings, 2022). However, the certificate was self-signed. It encrypted traffic but did not provide the same third-party trust assurance as a certificate issued by a trusted certification authority.

Table 2 Performance Testing Result Using ApacheBench

Parameter	Result	Interpretation
Number of requests	1,000	Total simulated requests
Concurrent connections	50	Simultaneous client connections
Requests per second	186.34	Observed throughput in the specified test
Time per request	268 ms	Observed mean response measure
Transfer rate	4.2 MB/s	Observed data-transfer rate
Failed requests	0	No failed request detected

ApacheBench completed 1,000 requests with 50 concurrent connections and reported zero failed requests. The measured throughput was 186.34 requests per second, the reported time per request was 268 ms, and the transfer rate was 4.2 MB/s. For the tested internal workload, these values indicate that the server responded without immediate request failure and provided adequate responsiveness for the observed use case.

The benchmark should nevertheless be interpreted within its experimental boundary. Jain (1991) cautions that a benchmark result is meaningful only in relation to its workload, configuration, and metrics. The result does not prove that all SIMETRI pages have identical cost, that performance will remain unchanged with larger databases, or that the server can sustain the same rate indefinitely. Application-layer delay can include processing time, database-query time, network transmission, and queueing; these components can change as concurrency and workload complexity increase (Kurose & Ross, 2021; Tanenbaum & Wetherall, 2021). A more complete capacity study would use repeated runs, warm-up control, percentile latency, multiple concurrency levels, realistic transactions, and long-duration observation.

Resource monitoring showed low CPU use during normal operation, memory consumption within the installed capacity, and no obvious disk-I/O bottleneck during the observed period. This suggests that the server retained headroom under the tested conditions. The result is an initial operational baseline rather than evidence of unlimited growth. Limoncelli et al. (2017) recommend

trend-based monitoring because gradual increases in database size, log volume, user concurrency, and backup workload may reveal constraints that are not visible in a short test.

Table 3 Functional Testing of SIMETRI Modules

Module	Function	Status
User Login	Internal user authentication	Successful
Admin Dashboard	Displaying administrative statistics	Successful
Incoming Letter Module	Managing and viewing incoming letters	Successful
Outgoing Letter Module	Managing approval and delivery status	Successful
Internal Document Module	Viewing categories of internal documents	Successful
Electronic Signature Module	Processing electronic-signature workflow	Successful
Guest Book Module	Managing guest-book and photo feature	Successful
Legal Aid Post Module	Managing legal-aid service processes	Successful
Configuration Module	Managing application settings	Successful

Functional testing confirmed that all principal SIMETRI modules completed their tested operations through HTTPS. The login module authenticated internal users, the dashboard displayed administrative information, and the document modules supported correspondence and internal-document workflows. Electronic-signature, guest-book, legal-aid, and configuration functions also operated successfully. Observed module response times ranged from approximately 0.4 to 0.9 seconds, depending on process and query complexity.

These outcomes demonstrate end-to-end integration across the web server, PHP runtime, database service, application code, and network path. Functional success does not by itself establish security or data correctness under every exceptional condition. Database-system literature distinguishes normal transaction processing from broader concerns such as concurrency control, integrity enforcement, recovery, and authorization (Elmasri & Navathe, 2022). Future testing should therefore include invalid input, interrupted transactions, duplicate submissions, role-based access, backup restoration, and failure recovery.

Table 4 Security Audit Result Using Lynis

Security Area	Score	Status
Kernel hardening	84/100	Good
File permissions	88/100	Good
Network security	90/100	Very good
Web-server security	92/100	Very good
Cryptography	85/100	Good

The Lynis assessment produced good to very good results across the evaluated areas. Web-server and network-security scores were the highest, indicating that service exposure and Apache-related controls were comparatively strong in the assessed configuration. Cryptography also received a good score because encrypted communication and modern protocol support were active.

The scores should be treated as diagnostic indicators rather than a security guarantee. A hardening audit evaluates configuration against a defined profile, whereas application vulnerabilities may require code review, authenticated scanning, penetration testing, and business-logic analysis. Defense-in-depth literature stresses that no single control or audit can cover all threat paths (Anderson, 2020; Stallings & Brown, 2018). Accordingly, Lynis should be repeated after major changes and combined with log review, patch management, access-control review, backup tests, and application-security testing.

Three principal risks were identified. First, PHP 5.6 is unsupported and may contain unpatched vulnerabilities. Second, the self-signed certificate is suitable for controlled internal use only when clients deliberately trust it; wider deployment should use a certificate issued by a trusted authority. Third, the absence of a dedicated Web Application Firewall leaves fewer application-layer detection and blocking options for attacks such as SQL injection and cross-site scripting. The OWASP Top 10 provides an application-risk framework, while a WAF such as ModSecurity can add a layer of protection but cannot replace secure code and access control (Open Web Application Security Project, 2021; Stallings & Brown, 2018).

Risk mitigation should proceed in phases. The immediate phase should restrict SIMETRI to the internal network, keep MariaDB bound locally, enforce HTTPS, review file ownership, preserve logs, and test backups. The short-term phase should deploy ModSecurity in a tested configuration, centralize or periodically review logs, document administrator accounts, and replace the self-signed certificate if access expands. The strategic phase should assess source-code compatibility and migrate from PHP 5.6 to a supported PHP release. Limoncelli et al. (2017) emphasize that successful change management requires testing, rollback, documentation, and communication; therefore, runtime migration should be staged rather than performed directly on the production server.

The findings support the feasibility of a LAMP-based on-premise deployment for an internal judicial application when the environment is carefully configured. The technical design reflects several book-based principles: a stable and documented operating platform (Hertzog & Mas, 2022; Nemeth et al., 2018), separation and monitoring of operational services (Limoncelli et al., 2017), relational-data controls (Elmasri & Navathe, 2022), layered network and host security (Anderson, 2020; Stallings & Brown, 2018), encrypted client-server communication (Ristić, 2022; Stallings, 2022), and workload-bounded performance interpretation (Jain, 1991). This alignment strengthens the argument that the result is not merely a successful installation but an implementation informed by established systems, networking, database, and security principles.

The study has limitations. It evaluates one physical server, one institution, and one synthetic benchmark configuration. It does not include a production-traffic trace, repeated statistical trials, long-duration stress testing, failover testing, disaster-recovery measurement, source-code security review, or a formal penetration test. Lynis results may also vary with audit version and profile. Consequently, the numerical results should not be generalized to other hardware, database sizes, network conditions, or user populations without additional testing. These limitations define a future research agenda involving multi-level load tests, percentile latency, realistic transaction scripts, vulnerability assessment, backup-restoration tests, and post-migration comparison on a supported PHP runtime.

CONCLUSION

The SIMETRI web-based information system was successfully deployed on an HP ProLiant DL320e Gen8 V2 server using Debian 11, Apache HTTP Server 2.4, MariaDB 10.5, PHP 5.6, and SSL/TLS. HTTPS operated with TLS 1.3, the principal application modules completed their functional tests, and ApacheBench completed 1,000 requests at 50 concurrent connections with 186.34 requests per second, 268 ms time per request, a 4.2 MB/s transfer rate, and zero failed requests. The Lynis assessment produced good to very good results for the evaluated host-hardening areas. These findings indicate that the configuration is feasible for the tested internal judicial workload. However, feasibility is conditional on continuing administration and risk reduction. PHP 5.6 should be migrated to a supported release through staged compatibility testing; the self-signed certificate should be replaced if access expands; and application-layer protection, backup testing, monitoring, and periodic security assessment should be strengthened. Because the evaluation concerned one server and a bounded synthetic workload, future work should use repeated realistic load tests, application-security assessment, recovery testing, and comparative evaluation after the runtime migration.

REFERENCES

Anderson, R. (2020). Security engineering: A guide to building dependable distributed systems (3rd ed.). Wiley.

Apache Software Foundation. (2024). Apache HTTP Server documentation. https://httpd.apache.org/docs/

Badan Siber dan Sandi Negara. (2023). Panduan keamanan sistem informasi pemerintahan daerah. Badan Siber dan Sandi Negara Republik Indonesia.
Elmasri, R., & Navathe, S. B. (2022). Fundamentals of database systems (7th ed.). Pearson.
Hertzog, R., & Mas, R. (2022). The Debian administrator's handbook: Debian Bullseye from discovery to mastery. Freexian.
Hewlett Packard Enterprise. (2024). HPE ProLiant DL320e Gen8 V2 server documentation.
Internet Security Research Group. (2024). Let's Encrypt documentation. https://letsencrypt.org/docs/
Jain, R. (1991). The art of computer systems performance analysis: Techniques for experimental design, measurement, simulation, and modeling. Wiley.
Kurniawan, R., & Siregar, F. (2022). Securing government web applications using SSL and LAMP Stack. Jurnal Teknologi Informasi Indonesia, 8(2), 45-54.
Kurose, J. F., & Ross, K. W. (2021). Computer networking: A top-down approach (8th ed.). Pearson.
Limoncelli, T. A., Hogan, C. J., & Chalup, S. R. (2017). The practice of system and network administration: DevOps and other best practices for enterprise IT (3rd ed.). Addison-Wesley Professional.
Lynis Project. (2024). Lynis security auditing and hardening documentation. CISOfy. https://cisofy.com/lynis/
MariaDB Foundation. (2024). MariaDB server documentation. https://mariadb.org/documentation/
National Institute of Standards and Technology. (2019). Guidelines for the selection, configuration, and use of Transport Layer Security (TLS) implementations (SP 800-52 Rev. 2).
Nemeth, E., Snyder, G., Hein, T. R., Whaley, B., & Mackin, D. (2018). UNIX and Linux system administration handbook (5th ed.). Addison-Wesley Professional.
Nugroho, D. (2023). Implementation of SSL certificates in electronic government systems. Jurnal Rekayasa Sistem Informasi, 5(1), 33-42.
Open Web Application Security Project. (2021). OWASP Top 10: The ten most critical web application security risks. https://owasp.org/Top10/
PHP Group. (2024). PHP manual: PHP 5.6 documentation. https://www.php.net/manual/en/
Rescorla, E. (2018). The Transport Layer Security (TLS) Protocol Version 1.3 (RFC 8446). Internet Engineering Task Force. https://www.rfc-editor.org/rfc/rfc8446
Ristić, I. (2022). Bulletproof TLS and PKI: Understanding and deploying SSL/TLS and PKI to secure servers and web applications (2nd ed.). Feisty Duck.
Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). Operating system concepts (10th ed.). Wiley.
Stallings, W. (2022). Network security essentials: Applications and standards (7th ed.). Pearson.
Stallings, W., & Brown, L. (2018). Computer security: Principles and practice (4th ed.). Pearson.
Tanenbaum, A. S., & Wetherall, D. J. (2021). Computer networks (6th ed.). Pearson.
The Linux Foundation. (2023). Linux system administration best practices. The Linux Foundation.