



UNIVERSITY OF HERTFORDSHIRE
School of Physics, Engineering and Computer
Science

MSc Computer Networks and Systems Security

MSc CS Project

04/09/2025

Designing and Testing a Secure Multi-Site Network Using
OpenVPN and Cloud-Hosted Linux Servers

Name: Abi Ashok

Student ID: 23088591

Supervisor: Dr. Nnamdi Nwanekezie

Declaration

Project Title: Designing and Testing a Secure Multi-Site Network Using OpenVPN and Cloud-Hosted Linux Servers

Student Name: Abi Ashok

Student ID: 23088591

Date: 04/09/2025

I, Abi Ashok, hereby declare that the research conducted for this MSc Final Project did not involve any human participants.

The project was entirely technical in nature, concentrating on the formulation, monitoring and assessment of a virtualised network infrastructure. All configurations, performance tests and security tests were carried out on an isolated controlled laboratory platform in a virtual environment using simulated data and a network. The tools employed in undertaking the tests and inspection were set on the virtual systems that the researcher created during this project.

Accordingly, there were no human subjects used in this work, and no people were observed or engaged in during the research. Accordingly, the aspects of research ethics and review in regard to human participant involvement were not applicable.

Signed: Abi Ashok

04/09/2025

Abstract

This project describes the architecture and execution, as well as the thorough testing of a secure multi-site network using OpenVPN on virtualised Linux servers. The general aim was to build a robust and hardened Virtual Private Network (VPN) infrastructure that mimics real-world geographically dispersed enterprise networks.

The methodology followed was phased in, with a virtual testbed initially set up and then OpenVPN was configured with a Public Key Infrastructure (PKI) to authenticate an effective process using certificates. There was a systematic enhancement of security by means of thorough system hardening. This involved the use of tight iptables firewall rules that would reduce the attack surface, the use of fail2ban to offer automated defence of brute force attacks, and the use of key-only authentication of SSH, which would provide secure administrative access.

Security, reliability and performance of the final system were evaluated. Several controlled tests were conducted to prove the effectiveness of VPN in automatic recovery of services interrupted and leverages/reboot of the server without much downtime. Performance tests via iperf3 showed consistent throughput of the enterprise workload, and security checks by nmap as well as Wireshark showed no breaches in the firewall and the integrity of the tunnelled traffic end-to-end encryption.

This project provides a base for the future development like the integration of cloud-based orchestration, the switch of cryptographic standards to the next generation, and AI-driven anomaly detection as a supplementary impact to the monitoring. The further implementation of the solution to hybrid and multi-cloud environments can help organizations to scale more effectively, maintain resilience, and be more compliant with the set cybersecurity standards, as they are regularly refreshed.

Table of Contents

Declaration	2
Abstract	3
Chapter 1: Introduction	10
1.1 Background.....	10
1.2 Current Issue.....	10
1.3 Project Specification.....	10
1.4 Project Aim and Objectives.....	11
1.5 Scope and Limitations.....	11
1.6 Research Questions.....	11
1.7 Feasibility.....	12
1.8 Structure of the Report.....	12
Chapter 2: Background and Literature Review	13
2.1 Introduction to Virtual Private Networks (VPNs).....	13
2.2 VPN Protocols: Performance and Applicability.....	13
2.3 Enhancing VPN Architecture and Gateway Performance.....	14
2.4 Linux-Based VPN Deployment and System Hardening.....	16
2.5 Security Challenges in VPN Deployments.....	18
2.6 Synthesis and Research Gap.....	19
Chapter 3: Methodology and System Design	23
3.1 Choice of Methods.....	23
3.2 Justification and Support of Choices.....	23
3.3 Project Design.....	24
3.3.1 System Design Overview.....	24
3.3.2 Network Architecture and Virtual Infrastructure.....	25
3.3.3 Implementation Workflow.....	25
3.4 Validation Strategy.....	26

3.5 Risk Management.....	27
3.6 Ethical Considerations.....	27
Chapter 4 Quality and Result	29
4.1 Implementation and Practical Work.....	29
4.1.1 Virtual Machine Setup and Initial Configuration.....	29
4.1.2 Network Configuration and Connectivity Validation.....	31
4.1.3 Tool Installation and Environment Preparation.....	32
4.1.4 VPN Server Configuration.....	32
4.1.5 VPN Client Configuration.....	35
4.1.6 Security Hardening with Firewall and Intrusion Prevention.....	37
4.1.7 SSH Authentication Configuration.....	38
4.1.8 Reliability Enhancements and Tunnel Recovery.....	40
4.2 Analysis.....	42
4.3 Results.....	44
4.3.1 Testing Environment.....	44
4.3.2 Reliability and Reconnection Testing.....	44
4.3.3 Performance and Load Testing.....	45
4.3.4 Functional and Security Verification.....	46
4.3.5 Comparison with Professional Standards.....	48
4.4 Novelty & Contribution.....	49
4.5 Objectives & Literature Link.....	49
Chapter 5: Evaluation and Conclusion	51
5.1 Evaluation Against Objective.....	51
5.2 Recommendations for Future Work.....	52
5.3 Final Remarks.....	53
5.5 Conclusion.....	54
References	55

Appendices	60
Appendix A: Project Timeline.....	60
Appendix B: Requirement Specification Document.....	62
Appendix C: Configuration File Screenshots.....	65
Appendix D: Test Results Table and Screenshots.....	68

List of Figures

Figure 1: Generic High-Performance Architecture (GHPA) for VPN gateways using DPDK for improved throughput and modularised user-space packet routing	15
Figure 2: Linux networking stack showing packet processing stages using Netfilter chains and eBPF/XDP hooks.	17
Figure 3: Testbed architecture used to evaluate CIS Benchmark hardening on Ubuntu servers against simulated cyberattacks.	18
Figure 4 Conceptual Framework	19
Figure 5: Site B and VPN Server (Ubuntu Server) Installation successful in VMware.	30
Figure 6: AdminHost(Ubuntu Desktop) Installation successful in VMware.	30
Figure 7: Network configuration in the VPN-Server machine.	31
Figure 8: Network configuration in the Site-B machine.	31
Figure 9: Installing required packages and tools in the VPN-Server machine.	32
Figure 10: Installing required packages and tools in the Site-B machine.	32
Figure 11: Installing required packages and tools in the Admin-Host machine.	32
Figure 12: IP forwarding enabled in sysctl config.	33
Figure 13: IP forwarding setting applied successfully.	33
Figure 14: Certificate Authority (CA) created using Easy-RSA.	34
Figure 15: VPN server certificate and key generated.	34
Figure 16:Diffie-Hellman parameters generated for key exchange.	34
Figure 17: TLS HMAC key generated for added security.	34
Figure 18: OpenVPN server configuration file created.	35
Figure 19: OpenVPN server service started successfully.	35
Figure 20: Firewall rules configured for OpenVPN traffic.	35
Figure 21: Client certificate and key generated for Site-B.	36
Figure 22: OpenVPN client configuration created on Site-B.	36
Figure 23: OpenVPN client connected to VPN server.	37
Figure 24: VPN server firewall rules applied with iptables.	38
Figure 25: fail2ban monitoring SSH on vpn-server.	38
Figure 26: SSH key pair generated on AdminHost.	39
Figure 27: SSH public key copied to VPN server and Site B.	39
Figure 28: SSH password login disabled in sshd_config.	40
Figure 29: Password Authentication denied by the VPN Server.	40

Figure 30: VPN client reconnected after service restart.	41
Figure 31: Openvpn Server restart log.	41
Figure 32: VPN client reconnected automatically after full server reboot	41
Figure 33: System reboot recorded at 05:46 confirms VPN server downtime during recovery test.	41
Figure 34: VPN server interface temporarily disabled and restored.	41
Figure 35: VPN tunnel recovered after server interface disruption.	42
Figure 36: Ping to VPN server interrupted during client restart and automatically resumed.	42
Figure 37: OpenVPN client restarted on Site-B.	42
Figure 38: VPN server ports scanned using nmap.	47
Figure 39: fail2ban banned attacking IP after login failures.	47
Figure 40: Encrypted OpenVPN traffic captured in Wireshark.	47
Figure 41: Visual Project Gantt chart.	61
Figure 42: OpenVPN server configuration file (server.conf)	65
Figure 43: OpenVPN client configuration created on Site-B.	65
Figure 44: IP Forwarding Configuration.	66
Figure 45: SSH Server Configuration	66
Figure 46: iptables rule set file.	67
Figure 47: VPN client reconnected after service restart.	68
Figure 48: Openvpn Server restart log.	68
Figure 49: VPN client reconnected automatically after full server reboot	68
Figure 50: System reboot recorded at 05:46 confirms VPN server downtime during recovery test.	69
Figure 51: VPN server interface temporarily disabled and restored.	69
Figure 52: VPN tunnel recovered after server interface disruption.	69
Figure 53: Ping to VPN server interrupted during client restart and automatically resumed.	69
Figure 54: OpenVPN client restarted on Site-B.	69
Figure 55: iperf3 TCP test.	70
Figure 56: iperf test on 4 parallel TCP streams.	71
Figure 57: iperf test for UDP 10 Mbit/s for 60 sec.	71
Figure 58: 500 MB file transferred securely over VPN tunnel using SCP.	72
Figure 59: System resource usages on rest.	72
Figure 60: System resource usages during the test iperf3 -c 10.8.0.1 -t 300.	73
Figure 61: System resource usages during the test iperf3 -c 10.8.0.1 -t 60 -P 4.	73

Figure 62: System resource usages during the test iperf3 -c 10.8.0.1 -u -b 10M -t 60. 74

List of Tables

Table 1 Related work	21
Table 2: IP Address planning for the VMs.	25
Table 3: Risk Management table.	27
Table 4 Comparison of Proposed Solution vs. Baseline/Alternative Approaches	50
Table 5 Evaluation of Objectives	52
Table 6: Project timeline.	60
Table 7: Reliability and Reconnection Test Results.	68
Table 8: Performance and Load Testing Results.	69

Chapter 1: Introduction

1.1 Background

The internet revolutionization of business processes and the worldwide proliferation of usage of off-site connections have imposed greater demands on dependable and encrypted communications on the network system between physically distributed frameworks. Enterprises now demand simple access to internal resources at remote buildings, cloud or mobile devices. This trend has increased the relevance of network security, especially when information has to move over the public internet. To combat these issues, Virtual Private Networks (VPNs) emerged as a primary technology of enterprise and cloud designs at present.

A VPN establishes an encrypted connection between two terminals, which may be people or any part of a network, so that any kind of information relayed is confidential and immune to interception, editing or tapping. VPNs mostly serve a practical purpose in an environment in which data passes through non-secure networks, as in the case of shared infrastructure or public WiFi. Open-source VPN solutions, like OpenVPN, have become extremely popular over time as VPN technologies have evolved, and the tools provide flexibility, high encryption strength, and several cross-platform services.

1.2 Current Issue

This increased reliance on VPNs due to events such as the COVID-19 pandemic has identified a serious vulnerability in most of the installations against configuration, scale and performance. Poorly configured tunnels or poor encryption, or breaches in firewalls, can make VPN useless and undo its security guarantees. This project meets these challenges by concentrating on the design, implementation and extensive testing of a site-to-site VPN, which acts as a model of an actual enterprise environment where there is a security requirement between branch offices which need to communicate.

1.3 Project Specification

This project is carried out in a completely isolated and self-contained virtualised environment with VMware Workstation. Every simulation is composed of three virtual machines with Ubuntu 22.04 version, one with a VPN server as the central, another one a remote client (Site-B), and an administrator host to monitor and test. Its key technologies of operation include OpenVPN to encrypt tunnels, Easy-RSA to handle the Public Key Infrastructure (PKI), iptables to harden the firewall and fail2ban to prevent intrusions. All tests are done with open-source tools, including iperf3, nmap and Wireshark. Since the project is hosted in a

virtual lab, it does not require access to any live production networks, and as such, all security testing is done on machines owned by the project rather than the University of Hertfordshire, remaining within the ethical principles of the University.

1.4 Project Aim and Objectives

The aim of this project is to design and test a secure communication setup between multiple remote Linux servers, connected using OpenVPN over the Internet. The network will be evaluated from a security perspective and hardened against most basic attacks.

The following objectives are brought in line with this aim:

1. To plan and deploy a multi-site network and virtualised Linux servers and configure a site-to-site OpenVPN tunnel with the use of certificate-based authentication.
2. To tighten the security of the VPN server and client systems using a tight firewall configuration using iptables, intrusion prevention using fail2ban and authentication using SSH keys only.
3. To test the VPNs performance on different load scenarios by using programs such as iperf3, and to test the reliability by imitating service and network related failures.
4. To confirm the safety of the VPN tunnel by the port scanning and analysis of encrypted traffic and provide evidence of this fact including the recommendations that follow the professional standards.

1.5 Scope and Limitations

The project is used within a virtualised environment, and no access to any live production networks/third-party systems is used. The testing and configuration are performed in self-contained Linux-based virtual machines using completely open-source tools that are free to use. The VPN is configured in site-to-site, which is used to represent inter-office traffic, and excludes remote access of individual mobile clients (Tian, 2025).

Although security mechanisms and firewalls are deployed by best practices, neither enterprise-level intrusion detection systems (IDS) nor centrally based monitoring products are in place in the project. Performance tests are performed in controlled settings, with the possibility that real-life latency on the internet may not be considered or the cross-platform interoperability difficulties.

1.6 Research Questions

The research questions are as follows, due to the aims of the projects:

1. What are some practical ways in which a secure and functional multi-site VPN can be provided using OpenVPN, which integrates a Public Key Infrastructure (PKI) in a virtualised Linux system?
2. Which are the most powerful hardening mechanisms in the inclusion of firewall settings and intrusion prevention to secure a Linux-based VPN gateway against general assailants to the network?
3. What is the overhead in performance and reliability of an OpenVPN tunnel in a virtualised space, specifically the throughput, latency and automatic reconnection capabilities?
4. What is the professional security standard against which the implemented VPN solution can be compared, and what are the best practices that should be followed in deploying and maintaining the implementation?

1.7 Feasibility

The project is feasible in that it uses all free, open-source software and a local virtualisation platform (VMware Workstation), there are therefore no financial constraints and no other third-party systems involved. The scope is well kept, as far as it involves one site-to-site configuration in an intranet isolated lab, which does not burden the work with multiple sets of problems and theories. Such key risks as VPN configuration mistakes and time management were determined during the planning stage and were mitigated via a systematic milestone-based strategy and regular testing.

1.8 Structure of the Report

The rest of this report is conditioned as follows:

- Chapter 2: Background and Literature Review is a critical literature review of current academic texts on VPN technologies, tunnelling protocols and security hardening methods.
- Chapter 3, Methodology and System Design, reveals the set of tools, configuration, and architectural choices that were made in the project.
- Chapter 4: Implementation and Practical Work explains the action that was taken to install the VPN and other related firewall solutions.
- Chapter 5: Evaluation and Results contains the results of security and performance tests.
- Chapter 6: Discussion, Conclusion and Recommendations is a summary of the project contributions and the recommendations regarding possible project improvements.

Chapter 2: Background and Literature Review

2.1 Introduction to Virtual Private Networks (VPNs)

Virtual Private Networks (VPNs) play a very essential role in protecting communication through untrusted centres, which usually include the public network. VPNs can be used to establish secret tunnels between remote systems or networks, and as such, confidential information can be transmitted with no possibility of getting intercepted by unauthorised parties, as well as the possibility of getting tampered with. Originally envisioned as cheap alternatives to leased lines to interconnect geographically distributed business sites, VPN have now found a multitude of applications, ranging in utility from corporate infrastructure capabilities to personal privacy applications and secure remote access products (Hall, 2025).

As more and more aspects of IT become dependent on cloud computing, remote work and distributed systems, most modern IT services are now based on VPNs. Normal VPN configuration involves the presence of one or more gateways/servers, secure protocols to channel information and encryption standards to safeguard content. The performance, security and interoperability depend on the chosen tunnelling protocols, encryption algorithm and system-level configuration. The remaining part of this Chapter critically examines the main elements of VPN technologies, such as their performance implication, architecture, hardening them, and the new issues being created during deployment and utilisation (Thiara, 2021).

2.2 VPN Protocols: Performance and Applicability

The comparative study by Akter et al. (2022) focuses on three widely known VPN protocols (IPSec, Layer 2 Tunnelling Protocol (L2TP) and Multiprotocol Label Switching (MPLS) and seeks to map each of them against various application requirements. In the experiment, their results indicated that even though all three protocols could handle applications like email, file sharing (not very sensitive to temporal delays), real-time audio and video applications could only be handled with reasonable performance by MPLS because it exhibited lower jitter and higher throughput. Although IPSec and L2TP have good encryption and security-related features, they were discovered to have longer round-trip times and jitter that pandered to high-delay services (Santoso et al., 2021).

A related work that has discussed OpenVPN is its middle ground between configurability and performance, even though it is not a central concern of the study by Akter et al. It normally scales up effectively with normal enterprise workloads, particularly when set up with efficient cyphers, like the AES. Nonetheless, its functionality may also differ drastically depending on

the configuration options, including which protocol is passed to it (UDP or TCP), the overhead of encryption, and the architecture of the system it is used in (Akter et al., 2022).

In addition to providing new remarks on the performance of VPN protocols in terms of latency, throughput, and packet drop rates (Jimenez-Lopez et al., 2022), Gentile et al. (2022) shed light on the situation in constrained groups, e.g., rural or IoT-oriented deployments. Their analysis looked at several of the available open-source VPNs, namely OpenVPN, WireGuard, the SSTP-compatible Accel-PPP, and OpenConnect. They tried such protocols on different client systems (e.g. Linux, Windows, Android, iOS) and server platforms (e.g. OpenWrt, Debian, Mikrotik). OpenVPN and WireGuard were named the most balanced regarding security, compatibility and speed. The OpenVPN protocol offered broad protocol support and good encryption, although WireGuard was commended on its minimal codebase, minimalism and good throughput, particularly where there is a tight environment such as limited CPU resources or limited memory.

WireGuard has become a favoured VPN option in new installations, in light of a trend toward performance-centred protocol choices. However, its utilisation is still low in legacy environments and business environments, because it has problematic interactions and management tools are not as mature as previous protocols such as IPsec (Logruosso, 2024).

By looking at the VPN usage and vulnerability in a global environmental analysis, Maghsoudlou et al. (2023) gave a new dimension to the problem. They studied the entire internet and scanned out VPN servers based on OpenVPN, SSTP, IPsec, and PPTP. They discovered huge differences in the deployed servers' security stance. Recklessly, it was discovered that more than 90 per cent of SSTP servers are susceptible to the notion of TLS downgrade attacks, which debunks the notion that newer VPN options are comparably safer. The paper also noted that there are difficulties in recognising VPN traffic correctly, because much of the associated configurations (e.g., OpenVPN using port 443) appear like HTTPS traffic to evade firewall rules. This puts doubt on detection and security management in practice networks.

2.3 Enhancing VPN Architecture and Gateway Performance

VPN gateways that are based on traditional software have many performance bottlenecks because of insufficient kernel-based packet processing, ineffective network interface card (NIC) operations and being inflexible. Fu et al. (2024) offered a new Generic High-Performance Architecture (GHPA) of VPN gateways to support these problems. Three prominent strategies were used to design them: bypassing the kernel protocol stack,

offloading the packet processing work to Data Plane Development Kit (DPDK), and splitting VPN gateway functions (session management, NAT, routing) into user-space tasks.

The GHPA prototype, which was operationalised as HP-VPN, showed significant improvement on key performance indicators such as round-trip time (RTT), system throughput, jitter and packet forwarding rates. GHPA proved to be faster in response times and a larger use of bandwidth when compared to traditional VPN implementations (T-VPN). Notably, the modularity of the architecture allowed flexibility by deploying in different modes, including cloud-based infrastructures and high-availability VPN clusters.

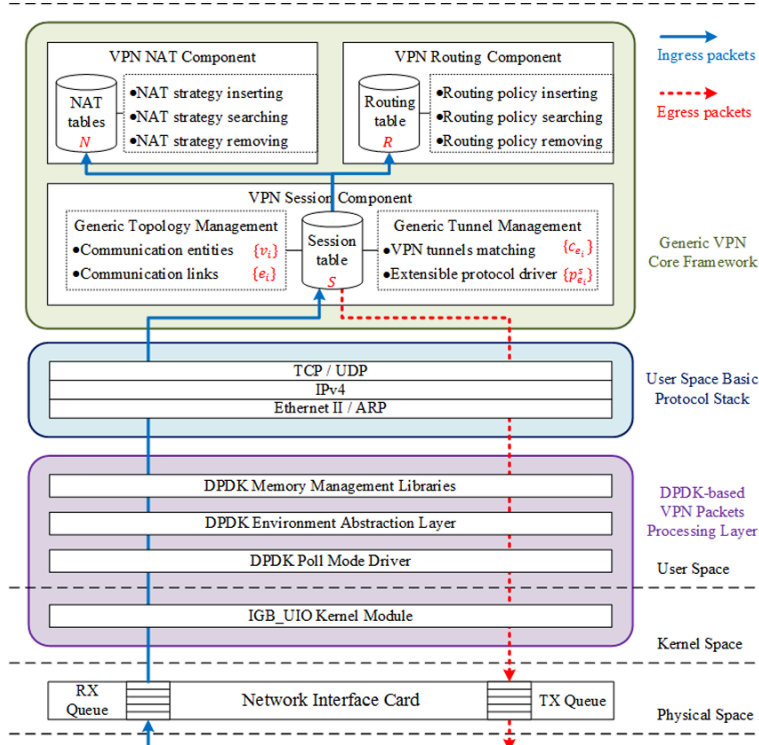


Figure 1: Generic High-Performance Architecture (GHPA) for VPN gateways using DPDK for improved throughput and modularised user-space packet routing
(Src: Fu et al., 2024).

The work of Fu et al. supports the increasing demand for scalable and high-performance VPN infrastructures, particularly when they are used in enterprise and cloud businesses where a significant amount of data needs to be handled safely and with minimal latency. The fact that their gateway model is separated into concerns also enables provisional improvements, which will be made later, including inbuilt implementations of advanced load-balancing, intrusion checking, and failover schemes.

Another alternative opinion was provided by Hall (2025), who measured the speed of OpenVPN running on a consumer-level router with DD-WRT firmware. In his analysis, he employed a 2k-p fractional factorial design to isolate effects produced by the five most

important configuration variables, such as encryption algorithm and transport protocol. The findings revealed that this selection of cypher influenced throughput quite strongly, and AES was the best compared to Blowfish and 3DES. TCP even played some role: UDP typically created less latency than TCP. These results give useful advice to both the user and administrator who want to maximise VPN performance with minimal hardware.

In combination, the paper by Fu and Hall indicates that one can optimise the performance of VPN deployment on both architectural (macro) and configuration (micro) levels. Although GHPA can be utilised only in long-term solutions when working with large-scale environments, proper installation of encryption and transport protocols may provide the short-term advantage when focusing on small-scale deployments.

2.4 Linux-Based VPN Deployment and System Hardening

Linux is also still the deployment platform of choice of many VPNs because it is flexible, has an extensive ecosystem of security tools, and is supported by the community in general. IPtables and nftables are types of tools that offer the necessary packet filtering and firewall functions and assist in enforcing security rules at the network layer.

Melkov and Paulikas (2021) have discussed the history and functionality of Linux firewall frameworks, revealing the major differences between the iptables system that was used long ago and the new nftables, as well as the newest technologies, such as eBPF and XDP. Although being deeply integrated into most of the production systems and well supported, iptables lacks scalability when large groups of rules are loaded or when frequent configuration changes are required. Nftables was introduced as a replacement and supports atomic changing of rules and a syntax that is easier to adapt. Its adoption, however, has been slow because of compatibility issues (Netfilter, 2025).

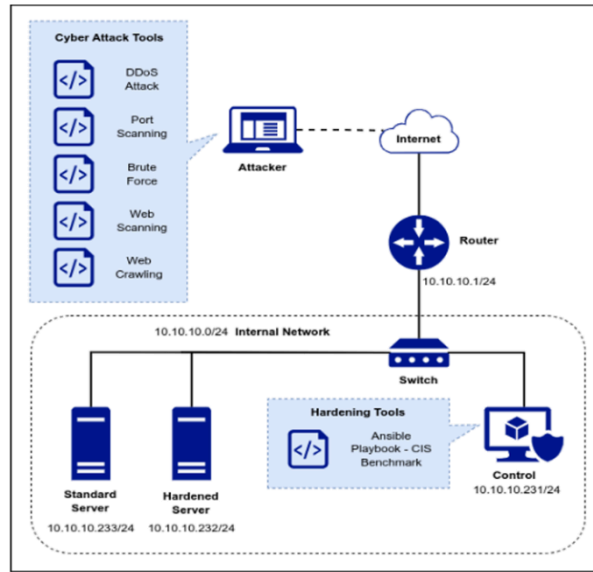


Figure 3: Testbed architecture used to evaluate CIS Benchmark hardening on Ubuntu servers against simulated cyberattacks.

(Src: Irawan et al., 2025)

As a part of this research, it is important to ensure the integrity of standardised security routines and automation to minimise vulnerabilities encountered in VPN servers. The implementation of hardening frameworks that have a good basis, like CIS, also secures the solutions against known threats as well as unifies configurations across deployments.

2.5 Security Challenges in VPN Deployments

Although VPNs are mainly used to improve security and stability, improper configurations, as well as old implementations, may lead to new threats. According to Maghsoudlou et al. (2023), huge VPN applications can be badly patch-managed and improperly configured. They discovered that most VPN servers have a web interface exposed, react to various protocols, or have expired certificates that raise the attack surface.

Similar concerns were voiced by Gentile et al. (2022) in the area of constrained or rural deployments. VPNs implemented on cheap routers or open-source firmware (e.g., OpenWRT) usually cannot be updated, monitored, or scanned regularly with automated threat detection. This trait makes them prime candidates for attackers interested in easy access to critical infrastructure, which may especially be the case at IoT or smart-city installations.

Security is also influenced by the selection of encryption. Hall (2025) proved that the default configuration of OpenVPN provides high security levels, whereas the use of deprecated versions of cypher or the misconfiguration of keys may weaken its protection. UDP versus TCP choice of the transport protocol is not only performance affecting but also may have an effect on firewall traversal and exposure to vulnerabilities. To cite an example, non-properly

encrypted TCP-based VPN traffic can easily be interfered with and altered through man-in-the-middle (MITM) attacks.

Melkov and Paulikas (2021) pointed out that firewalling, i.e., carefully written by investing in robust firewalling scripts with the help of iptables, nftables, or eBPF, helps minimise the possibilities of unauthorised access and reduce the attack surface of VPN systems. In addition, Irawan et al. (2025) demonstrated that patching the Linux servers that run VPN software through the principles of CIS hardening helps massively diminish the effectiveness of automated and sophisticated attacks.

2.6 Synthesis and Research Gap

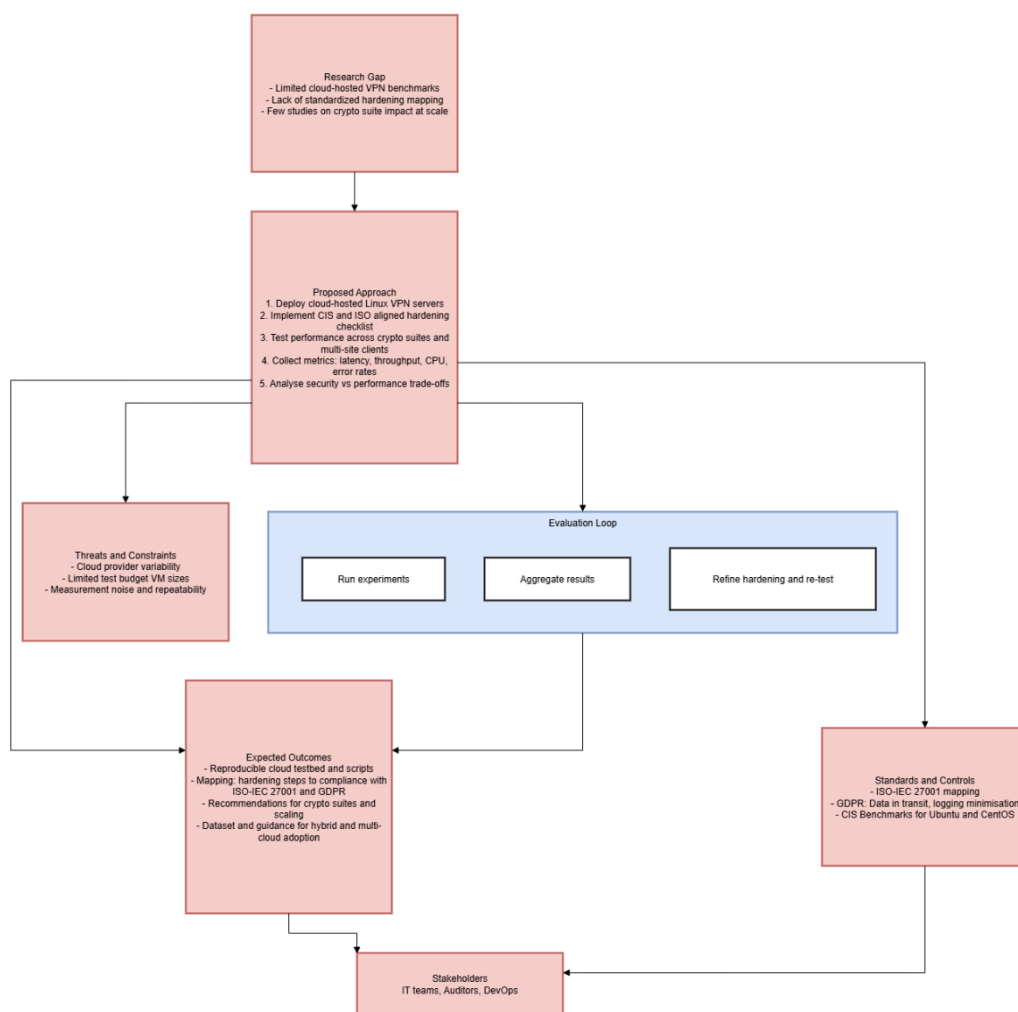


Figure 4 Conceptual Framework

The literature survey gives an exhaustive description of aspects related to VPN technology, such as the selection of protocol and tuning performance, architecture design and hardening security. Point out a few major themes:

- **Protocol diversity and specialisation:** No one VPN protocol suits all cases optimally. OpenVPN has the advantage of being balanced and versatile, whereas less compatible or advanced alternatives, such as WireGuard and MPLS, are faster and more straightforward (IVPN, 2025).
- **Layer-wise performance tuning can proceed:** It is possible to tune not only on the architectural (e.g., DPDK, user-space NAT), but also on a configuration level (e.g., cypher choice, transport mode). They shall be compatible with the particular hardware and the application (Manesh and Nezhad, 2024).
- **Security should be proactive rather than reactive:** The more prevalent implementations of VPNs have problems with patch management, misconfigurations, and outdated libraries. It is a must to apply a standardised framework (e.g., CIS), robust firewall policies, and key management procedures.

Despite this large literature base, there are several gaps:

1. In the real world, misconfigurations are underrepresented. Numerous researchers have conducted experiments in perfect research conditions and do not consider the various practices between organisations and individuals.
2. The combination of high-performance Linux networking software (e.g., eBPF) and VPNs is in some early stages of research, and there are no practical deployment overviews (Abranches et al., 2021).
3. There is little research that integrates architectural optimisation and standardised hardening, and these gaps exist between theoretical performance and secure implementation.
4. There is little available direction to help small organisations, or the academic setting strive to reconcile the goals of performance versus cost and security in the implementation of VPNs.

In this project, these gaps are aimed to be addressed via implementing and testing a secure multi-site VPN network based on OpenVPN with hardened Linux servers. It will consider the best practices of firewalling (iptables), secure tunnelling/tunnelling, and encryption, together with performance testing through such packet tools as iperf3, nmap and Wireshark (Chhillar and Shrivastava, 2021, Cloudflare, 2025). The aim is to provide a real-life experience

framework and a deployable system design of a securely built VPN infrastructure applicable in enterprise applications and educational settings.

Table 1 Related work

Author / Year	Methodology	Findings	Limitations	Relevance to Project
Sufi, F. (2023)	Literature review + case examples on social-media-driven CTI	Describes threat patterns originating from social media and importance of timely intelligence	High-level not experimental little performance data	Motivates need for timely logging & monitoring in VPN deployments
Zhang et al. (2022)	Survey on explainable AI (XAI) in cybersecurity applications	XAI improves analyst trust and detection interpretation	Mostly conceptual few production deployments	Supports future work: AI-driven anomaly detection for VPN telemetry
Neupane et al. (2022)	Review of explainable intrusion detection systems (X-IDS)	Frameworks and evaluations for interpretable alerts	Tests limited to academic datasets	Informs design of monitoring/alerting modules in testbed
Saeed et al. (2023)	Systematic literature review on Cyber Threat Intelligence (CTI)	Taxonomy of CTI sources & utility for defense	Scope excludes detailed system-level experiments	Justifies integrating CTI feeds/indicators into logging pipeline
Santos et al. (2025)	Systematic review of CTI methodologies	Updated taxonomy highlights automation gaps	Newer review, limited empirical deployment guidance	Reinforces automation & orchestration suggestions for cloud testbed
Salem et al. (2024)	Review of AI-driven detection techniques & evaluation metrics	Shows trade-offs between detection accuracy and resource usage	Broad, few applied cloud/VPN-specific benchmarks	Directly relevant to performance vs security trade-off in this project

Chapter 3: Methodology and System Design

3.1 Choice of Methods

The project employed a realistic, implementation-oriented approach to plan and implement the VPN infrastructure within a virtual setting. The approach to configuration was phased, iterative in nature, permitting staged configuration, testing and optimisation. It is reflective of engineering in the real world, where sub-units are constructed and tested incrementally. This method will make fault isolation much simpler and enable easy fixing of misconfigurations or bad performance as they come.

3.2 Justification and Support of Choices

An OpenVPN-based solution was chosen due to its open-source character, high security levels, and the broad community which supports the application, thereby enabling the development of secure communications over a distributed network. A set of bespoke but carefully selected open-source and platform native tools was chosen to ensure the project was both robust and cost-effective:

- **OS:** Ubuntu Server 22.04 LTS and Ubuntu Desktop 22.04 LTS have been selected as the operating systems due to long-term support, high reliability, and rich documentation. To cut the attack surface, the server edition was installed with the fewest number of packages possible, the desktop version offered GUI-based analysis and management tools.
- **Hypervisor:** VMware Workstation was applied to create a local multi-site environment. It gave complete administration of virtual network topology, IP assignments, and system resources distribution, and would be perfect to test the behaviour of VPN in a sandboxed network.
- **VPN Software:** OpenVPN was chosen because it supported encryption and flexibility of protocol (UDP/TCP) on top of being production ready. It implements PKI-based authentication, and it allows fine-grained security parameter configuration (OpenVPN.net, 2025).
- **Certificate Management:** Certificate and key generation and management were done via Easy-RSA. Because it is a CLI-based tool, it blends in well with OpenVPN and is an easy way of implementing a Public Key Infrastructure (PKI).
- **Firewall:** Traffic control and filtering were done through an in-built firewall called iptables in the Linux systems. Custom rules were also set to drop unsolicited traffic, and only the necessary ports were opened (Romantini, 2024).

- **Intrusion Protection:** fail2ban would scan log files of brute-force SSH login attempts and block suspecting IPs automatically. It was programmed to use iptables in a dynamic blocking (Singh et al., 2024).
- **Monitoring and Testing Tools:**
 - iperf3 - testing of bandwidth/throughput (TCP/UDP).
 - Ping and traceroute to test connectivity and routing.
 - Nmap for port scanning and simulation of an attack.
 - Wireshark packet examination and tunnel examination.
 - Nano was utilised in editing configuration files in the server CLI prompt.

The Ubuntu package manager (apt) was used to install all the tools to make them verified and stable. All the tools had their specific part in the deployment and testing pipeline.

3.3 Project Design

3.3.1 System Design Overview

The fundamental concept of the system design was to emulate a real-world enterprise having two geographically separated locations (Site-A and Site-B), where communication between the two locations is secured in an insecure publicly available network. Three virtual machines were used to deploy the architecture:

- **VPN Server (vpnsrvr):** Acts as the central VPN gateway.
- **Site-B Client (site-b):** Remote Linux server that connects to the central VPN hub.
- **Admin Host (adminhost):** A monitoring and control node with GUI tools.

These VMs got connected through two virtual switches set up in VMware:

- **vmnet0 (NAT):** Provided external connectivity for updates and internet communication.
- **vmnet1 (Host-only):** Provided isolated internal communication without internet exposure, ideal for testing site-to-site tunnels.

OpenVPN was set to create a site-to-site connection between the VPN server and the client in Site-B. The tunnel implemented UDP on port 1194, and it was encrypted with 2048-bit RSA certificates and an extra TLS-Auth key to prevent DoS and packet injection attacks. The Admin Host was connected to both of the machines through SSH and served to track traffic, perform tests and monitor the destination's behaviour through graphical software such as Wireshark.

The whole design was based on modular security principles: the minimum number of open ports, defined roles, end-to-end encrypted traffic flows, and the principle of least privilege.

3.3.2 Network Architecture and Virtual Infrastructure

Network design was carefully done to prevent IP conflict and to ensure the routing of traffic correctly externally and within the VPN tunnel. Both NAT and Host-only networks had the IP addresses assigned as static addresses to achieve stability during the process of testing each machine.

Table 2: IP Address planning for the VMs.

VM Name	NAT (ens33)	Interface	Host-only (ens37/ens34)	Interface	VPN IP	Tunnel
VPN-Server	192.168.2.20		192.168.56.10		10.8.0.1	
Site-B	192.168.2.21		192.168.56.20		10.8.0.2	
Admin-Host	192.168.2.22		192.168.56.30		N/A	

OpenVPN server allocated tunnel IPs with the help of a static configuration, and routing was checked with the aid of Linux built-in commands (ip route, netstat, ip a). The active monitoring of network namespaces and routing tables was performed to ensure that client traffic to the remote site had to cross the tunnel.

The NAT to Host-only interface routing was explicitly discouraged to avoid any inter-site communications over the host-only or NAT bridges.

3.3.3 Implementation Workflow

The construction was done in the following major stages:

Phase 1: VM Setup and Network Configuration

- Installed Ubuntu Server/Desktop on 3 VMs.
- Set up interfaces having static IPs and persistent names by using Netplan.
- Tested that NAT was working by applying ping outside (e.g. ping 8.8.8.8).
- Checked host-only communication through ping and SSH.

Phase 2: OpenVPN Configuration and Tunnel Creation

- Created CA on VPN Server and initialised the Easy-RSA directory.
- Created server certificate, key and Diffie-Hellman parameters (Firdaouss et al., 2022).
- Generated client certificate and key for Site-B.
- Set up an OpenVPN server without logging and compression.
- Created and shared TLS-auth key.
- Set up Site-B client remote server IP, certificates and proper tun0 interface.
- Enabled and checked the tunnel IP, ping, and traceroute.

Phase 3: Server Hardening

- Installed iptables and configured rules to have access to only OpenVPN and SSH.

- INPUT and FORWARD chains had default DROP policy rules.
- Set fail2ban to aggressive SSH protection rules.
- SSH password log-in disabled keys are pre-distributed, and permissions are set.
- Verified that the firewall is intact via an nmap scan of Admin Host.

Phase 4: Testing and Analysis

- Restarted the server and the client and tested reconnecting times.
- Reinitiated OpenVPN and checked client activity.
- Turned off interfaces momentarily to create a physical cable error.
- Iperf3 was run at 300 seconds (TCP) and 60 seconds (UDP).
- Measured SCP transport of a 500 MB file.
- Monitored resource consumption under stress with htop and iftop.
- Recorded all the test results to be analysed in Chapter 5.

Every step was well-documented by screenshots, logs and snapshots of configuration to aid repeatability and transparency.

3.4 Validation Strategy

A comprehensive testing strategy was elaborated to test functionality and security:

- **Functionality Testing:**
 - Ensured that the tunnel IP (10.8.0.0/24) had been assigned.
 - Traced the route to make sure that traffic was passing via VPN.
 - Verified encrypted packets using Wireshark.
- **Resilience Testing:**
 - Time after restarting `systemctl openvpn-server@server`.
 - Restarted the server and listed a successful ping.
 - Interface failure simulation with `ip link set down`.
- **Performance Testing:**
 - Measured throughput via iperf3 (single and 4-parallel TCP streams).
 - Analysed UDP jitter and packet loss.
 - Tested real-world transfer speed using `scp + time`.
- **Security Testing:**
 - Scanned the VPN server from Admin Host using `nmap`.
 - Attempted SSH brute-force attacks to trigger fail2ban.
 - Verified server does not agree with invalid or non-existent TLS-auth key.
- **Monitoring:**

- Observed OpenVPN and systemd logs (journalctl).
- Tracked CPU and RAM usage under load.

Our tests were also run 2-3 times and averaged when needed to make them consistent.

3.5 Risk Management

Some risks could be expected and pre-addressed by planning:

Table 3: Risk Management table.

Risk	Description	Likelihood	Mitigation Strategy
VPN config errors	Incorrect file paths or parameters	Medium	Validated configs using openvpn --config --test
SSH lockout	Key-only login could block access	Medium	Stored backup keys on Admin Host
Data loss	Snapshots or keys are deleted accidentally	Low	Backups are made after each major step
Performance limits	VMs may lag or freeze under load	Low	Adjusted CPU/RAM allocations in VMware
Time slippage	Practical work is taking longer than expected	High	Followed structured weekly milestones

These measures minimised the downtimes, enhanced traceability, and made every action repeatable in case of a system crash.

3.6 Ethical Considerations

Though human subjects and personal information will not be used in this project, there is a concern of ethical caution since network scanning and security testing tools will be used. The measures are the following safeguards:

- Scanning, testing and monitoring are carried out in a completely isolated virtual network.
- No external systems, services or real network linkages are made.
- Tools like nmap and Wireshark will only be used on the machines that are configured and owned by the project developer.
- Any identifiable or sensitive data is not captured, transmitted or stored.
- The project will abide by the ethical guidelines of the University of Hertfordshire, and an internal review concluded that there was no need to obtain ethics approval.

The maintenance of confidentiality, integrity, and availability (CIA) of data is the professional responsibility of a network administrator (Mishra et al., 2022). The project is ethical in that any testing functions, such as the application of potentially privacy-invading mechanisms, such as nmap, are caged within a purely virtual environment. There is no actual user data to be used. There is no violation of personally identifiable information (PII) (Duggineni, 2023).

Such a network in the real world would have to comply with strict legislative and regulatory demands, including the EU-wide General Data Protection Regulation (GDPR) that regulates how personal information is handled. Moreover, keeping the audit trail and logs is an essential professional practice that helps to trace the changes to the source of data and identify the breaches. One way of seeking solutions to these security issues is the implementation of a site-to-site VPN based on L2TP and IPsec (Santoso et al., 2021). On the same note, network security analysis would be vital in ensuring personnel and data confidentiality of the network-transferred information (Rosyidah and Parenreng, 2023). Increasing employee Information Security Awareness (ISA) is also one of the central professional responsibilities because the number of security incidents caused by human error remains high (Khando et al., 2021). Even though this project is technically oriented, it is because mere use of technology is not adequate unless these standards of ethics, legality and professional conduct are observed.

Chapter 4 Quality and Result

4.1 Implementation and Practical Work

4.1.1 Virtual Machine Setup and Initial Configuration

This practical activity of the project was initiated by the implementation of virtualised infrastructure, which simulates a multi-site enterprise network where security was enforced. This environment will consist of three main elements: one is a VPN Server, one is a client device called Site-B, and another one is an AdminHost machine designed to monitor, analyse the results, as well as simulate attacks. These systems were all instantiated as individual virtual machines (VMs) via the VMware Workstation to guarantee full control over network topology, with system parameters. VMware was chosen on account of its stability and the ease with which isolated networks can be configured, as well as the compatibility with the Linux operating system (Google Cloud, 2025).

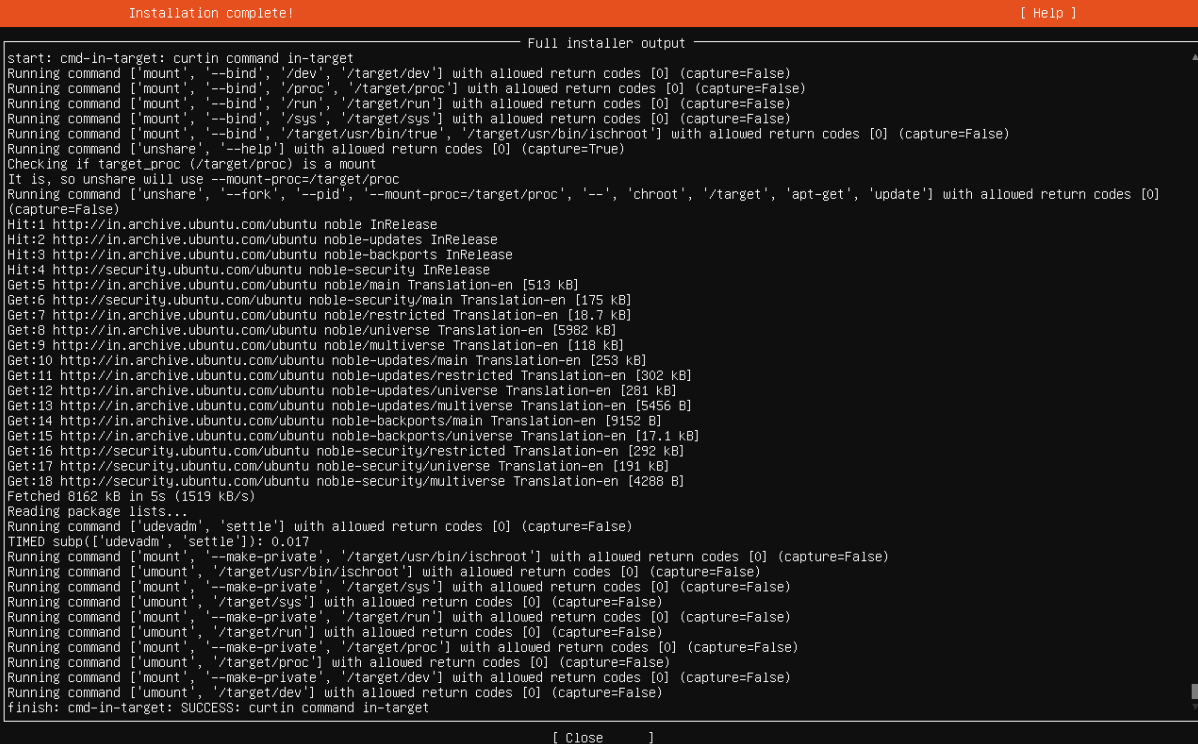
VPN Server VM was also set to become the focal point of safe passage between remote sites through encrypted communications. It was deployed on the Ubuntu Server 22.04 LTS platform, which is reliable, small-sized and has a large community behind it. This machine was to administer an OpenVPN tunnel, create firewall policies and also perform certificate authority functions. Long-term support also means that Ubuntu will receive consistent updates and security patches, and this is essential in a security-sensitive environment (Anicas, 2021).

A second Ubuntu Server 22.04 LTS instance, site-B, was configured to act as a remote office or branch office. It would act as an OpenVPN client to the central VPN Server, making it the endpoint of this site-to-site encrypted network (Amazon Web Services, 2025). The VM was created to simulate the network behaviour of a node which was placed geographically apart. Such settings play an important role in allowing a secure and smooth flow of communication across distributed company networks in enterprise settings.

Ubuntu Desktop 22.04 LTS was used to install AdminHost, which gives a graphical interface to analyse packets and monitor the system performance. This server was crucial in monitoring encrypted traffic, scanning, and ensuring the quality and safety of the network connection. The desktop interface made the GUI-based tool like Wireshark easy to use and helped us capture and analyse the packets in real time.

After the successful installation of the OS in all three machines, network interfaces were set up. Both virtual network adapters. That is, one on a network based on NAT to simulate an

external connectivity to the Internet, and one connected to a host-only network to simulate isolated, internal addresses. A static IP was used so that routing could be the same and no conflict would arise during the project. This setup made traffic segmentation possible, which is the basic tenet of network security (Broadcom, 2025).



```

Installation complete! [ Help ]

Full installer output

start: cmd-in-target: curtin command in-target
Running command ['mount', '--bind', '/dev', '/target/dev'] with allowed return codes [0] (capture=False)
Running command ['mount', '--bind', '/proc', '/target/proc'] with allowed return codes [0] (capture=False)
Running command ['mount', '--bind', '/run', '/target/run'] with allowed return codes [0] (capture=False)
Running command ['mount', '--bind', '/sys', '/target/sys'] with allowed return codes [0] (capture=False)
Running command ['mount', '--bind', '/target/usr/bin/true', '/target/usr/bin/ischroot'] with allowed return codes [0] (capture=False)
Running command ['unshare', '--help'] with allowed return codes [0] (capture=True)
Checking if target_proc (/target/proc) is a mount
It is, so unshare will use --mount-proc=/target/proc
Running command ['unshare', '--fork', '--pid', '--mount-proc=/target/proc', '--', 'chroot', '/target', 'apt-get', 'update'] with allowed return codes [0] (capture=False)
Hit:1 http://in.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://in.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://in.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:5 http://in.archive.ubuntu.com/ubuntu noble/main Translation-en [513 kB]
Get:6 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [175 kB]
Get:7 http://in.archive.ubuntu.com/ubuntu noble/restricted Translation-en [18.7 kB]
Get:8 http://in.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:9 http://in.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
Get:10 http://in.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [253 kB]
Get:11 http://in.archive.ubuntu.com/ubuntu noble-updates/restricted Translation-en [302 kB]
Get:12 http://in.archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [281 kB]
Get:13 http://in.archive.ubuntu.com/ubuntu noble-updates/multiverse Translation-en [5456 B]
Get:14 http://in.archive.ubuntu.com/ubuntu noble-backports/main Translation-en [9152 B]
Get:15 http://in.archive.ubuntu.com/ubuntu noble-backports/universe Translation-en [17.1 kB]
Get:16 http://security.ubuntu.com/ubuntu noble-security/restricted Translation-en [292 kB]
Get:17 http://security.ubuntu.com/ubuntu noble-security/universe Translation-en [191 kB]
Get:18 http://security.ubuntu.com/ubuntu noble-security/multiverse Translation-en [4288 B]
Fetched 8162 kB in 5s (1519 kB/s)
Reading package lists...
Running command ['udevadm', 'settle'] with allowed return codes [0] (capture=False)
TIMEO sub(['udevadm', 'settle']): 0.017
Running command ['mount', '--make-private', '/target/usr/bin/ischroot'] with allowed return codes [0] (capture=False)
Running command ['mount', '--make-private', '/target/sys'] with allowed return codes [0] (capture=False)
Running command ['mount', '--make-private', '/target/run'] with allowed return codes [0] (capture=False)
Running command ['mount', '--make-private', '/target/proc'] with allowed return codes [0] (capture=False)
Running command ['mount', '--make-private', '/target/dev'] with allowed return codes [0] (capture=False)
Running command ['mount', '--make-private', '/target/dev'] with allowed return codes [0] (capture=False)
finish: cmd-in-target: SUCCESS: curtin command in-target

[ Close ]

```

Figure 5: Site B and VPN Server (Ubuntu Server) Installation successful in VMware.

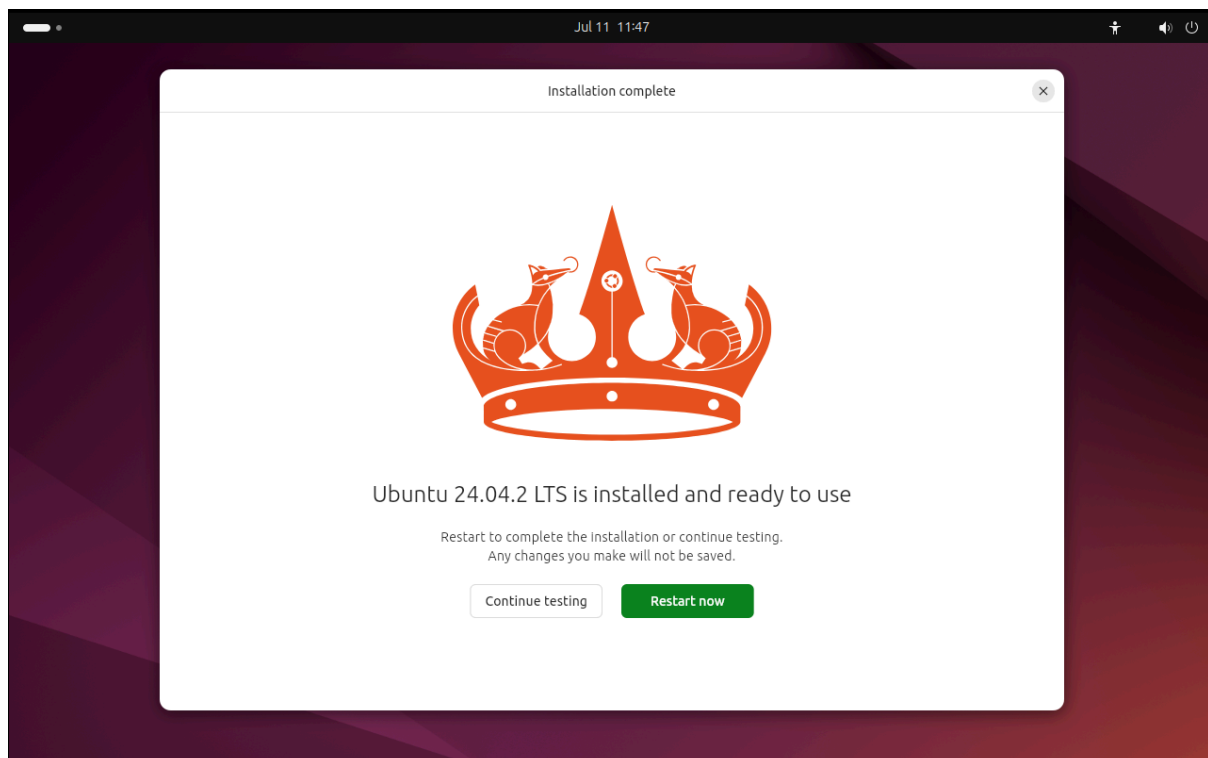


Figure 6: AdminHost(Ubuntu Desktop) Installation successful in VMware.

4.1.2 Network Configuration and Connectivity Validation

To enable stable and safe communication, every VM had its network interfaces carefully configured. The host-only network was assigned static IP addressing so that VPN traffic can be routed deterministically. All the IP addresses were distributed to the VPN Server and Site-B computer in the 192.168.56.0/24 range, and NAT interfaces were in the 192.168.2.0/24 range. This configuration is so that there is containment of the internal VPN communication without external background under the same network, but it can have the ability to communicate in an outbound context through all the updates and installations.

The appropriate connectivity assurance among the machines was ensured with simple network tools. The Ping tests have been run on both NAT and host-only interfaces, and all the systems were able to communicate with one another via assigned IPs. This basic network structure was essential in that it had to be determined that the VPN tunnel could be established later on without any routing problems. Also, the application of the Netplan technique produced stable interface naming and routinely defined boot-time settings that may be common root causes of error within the dynamic virtualised realms.

```
GNU nano 7.2 /etc/netplan/01-netcfg.yaml *
network:
  version: 2
  ethernet:
    ens33:          # NAT interface (VMnet0)
      dhcp4: true
    ens37:          # Host-only interface (VMnet1)
      addresses:
        - 192.168.56.10/24
      gateway4: 192.168.56.1
      nameservers:
        addresses:
          - 8.8.8.8
          - 1.1.1.1
```

Figure 7: Network configuration in the VPN-Server machine.

```
GNU nano 7.2 /etc/netplan/01-netcfg.yaml *
network:
  version: 2
  ethernet:
    ens33:          # NAT interface (VMnet0)
      dhcp4: true
    ens34:          # Host-only interface (VMnet1)
      addresses:
        - 192.168.56.20/24
      gateway4: 192.168.56.1
      nameservers:
        addresses:
          - 8.8.8.8
          - 1.1.1.1
```

Figure 8: Network configuration in the Site-B machine.

4.1.3 Tool Installation and Environment Preparation

The infrastructure was built, and, with it, the installations of necessary tools on all operating systems were made to be ready to implement VPN usage, security hardening, and monitoring. These were networking tools, performance analysis tools, packet sniffer and security systems. Solutions like OpenSSH were the key to secure remote access. `iputils-ping` and `net-tools` allowed performing connectivity-related checks on lower levels, and `iperf3` offered the option to perform future throughput-based testing (iPerf.fr, 2025). A `traceroute` command enabled visualisation of packet routes, Wireshark, where encrypted traffic could be inspected through a graphical interface, was installed on AdminHost. They were installed via the Ubuntu package manager (APT), which makes all their installations secure and validated.

Other utilities which were deployed include administration utilities such as `nano`, `curl` and `wget` in order to download resources, and `unzip` to manipulate files. Security applications like `iptables`, where the firewall was controlled, and `fail2ban`, where intrusion was detected, were specifically applied on the VPN Server and Site-B Machines. VPN configuration was based on the combination of `OpenVPN` and `easy-RSA` packages. `Easy-RSA` made things easier in the control of Public Key Infrastructure (PKI) and enabled secure generation of digital certificates (Orofino, 2024).

```
vpnadmin@vpnserver:~$ sudo apt update && sudo apt install -y openssh-server net-tools iputils-ping traceroute nano curl wget unzip zip iperf3 iptables ufw fail2ban openvpn easy-rsa

Hit:1 http://in.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://in.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://in.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... 64%
```

Figure 9: Installing required packages and tools in the VPN-Server machine.

```
siteadmin@site-b:~$ sudo apt update && sudo apt install -y openssh-server net-tools iputils-ping traceroute nano curl wget unzip zip iperf3 iptables ufw fail2ban openvpn easy-rsa

Hit:1 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:2 http://in.archive.ubuntu.com/ubuntu noble InRelease
Hit:3 http://in.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:4 http://in.archive.ubuntu.com/ubuntu noble-backports InRelease
Reading package lists... 6%
```

Figure 10: Installing required packages and tools in the Site-B machine.

```
adminuser@adminhost:~$ sudo apt update && sudo apt install -y openssh-server net-tools iputils-ping traceroute nano curl wget unzip zip iperf3 nmap wireshark

Hit:1 http://archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
```

Figure 11: Installing required packages and tools in the Admin-Host machine.

4.1.4 VPN Server Configuration

The second step was to install the VPN server. One of the initial procedures in this was to set up an IP forwarding capability on the server, because it is required to route packets between the client network and the Internet at large. It made this change on a persistent basis, by the use of system configuration files, and it took effect instantly. IP forwarding is required so that the Linux server can turn into a powerful router, which is a requirement for operation as a VPN gateway.

Afterwards, using Easy-RSA, the Certificate Authority (CA) was established. The CA was the authority that was trusted to issue digital certificates needed to authenticate the server as well as the client in the OpenVPN infrastructure. This was followed by the generation of a self-signed certificate and private key of the server, which was securely stored.

Diffie-Hellman parameters were developed to provide security to the key exchange process in the course of establishing VPN sessions. Also, an HMAC (hash-based message authentication code) based TLS authentication key (ta.key) was constructed to head off flood and brute force threats to the control channel. This extra authentication layer guaranteed secure association of different entities, in that only authenticated entities could initiate secure tunnels.

The OpenVPN server configuration file was generated with the cryptography materials prepared. Such a file specified vital settings such as port, protocol, authentication mechanism and network pathways. After the appropriate arrangement was made, the OpenVPN service was successfully enabled and started. The service was also set to commence automatically at boot time, such that the availability is maintained automatically.

That was followed by the definition of firewall rules with the iptables in order to allow the VPN traffic to pass on the specific port (UDP 1194) as well as permitting routing through the tunnel interface. NAT was set up so as to allow outgoing traffic of clients linked to the VPN. The firewall rules were used to establish a firewall around the VPN server that would restrict all but those specific services that have been configured explicitly to be run.

```
# Uncomment the next line to enable TCP/IP SYN cookies
# See http://lun.net/Articles/277146/
# Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_syncookies=1

# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1

# Uncomment the next line to enable packet forwarding for IPv6
# Enabling this option disables Stateless Address Autoconfiguration
# based on Router Advertisements for this host
#net.ipv6.conf.all.forwarding=1
```

Figure 12: IP forwarding enabled in sysctl config.


```
vpnadmin@vpnsrvr:~$ sudo sysctl -p
net.ipv4.ip_forward = 1
vpnadmin@vpnsrvr:~$
```

Figure 13: IP forwarding setting applied successfully.

```
vpnadmin@vpnserver:~$ make-cadir ~/openvpn-ca
vpnadmin@vpnserver:~$ cd ~/openvpn-ca
vpnadmin@vpnserver:~/openvpn-ca$ ./easyrsa init-pki

Notice
-----
'init-pki' complete; you may now create a CA or requests.

Your newly created PKI dir is:
* /home/vpnadmin/openvpn-ca/pki

Using Easy-RSA configuration:
* /home/vpnadmin/openvpn-ca/vars

vpnadmin@vpnserver:~/openvpn-ca$
vpnadmin@vpnserver:~/openvpn-ca$
```

Figure 14: Certificate Authority (CA) created using Easy-RSA.

```

Notice
-----
Private-Key and Public-Certificate-Request files created.
Your files are:
* req: /home/vpnadmin/openvpn-ca/pki/reqs/server.req
* key: /home/vpnadmin/openvpn-ca/pki/private/server.key

You are about to sign the following certificate:
Request subject, to be signed as a server certificate
for '825' days:

subject=
    commonName                = server

Type the word 'yes' to continue, or any other input to abort.
Confirm request details: yes

Using configuration from /home/vpnadmin/openvpn-ca/pki/openssl-easyrsa.cnf
Enter pass phrase for /home/vpnadmin/openvpn-ca/pki/private/ca.key:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :ASN.1 12:'server'
Certificate is to be certified until Oct 14 07:47:11 2027 GMT (825 days)

Write out database with 1 new entries
Database updated

Notice
-----
Certificate created at:
* /home/vpnadmin/openvpn-ca/pki/issued/server.crt

Notice
-----
Inline file created:
* /home/vpnadmin/openvpn-ca/pki/inline/server.inline

vpnadmin@vpnserver:~/openvpn-ca$

```

Figure 15: VPN server certificate and key generated.

```
.....+++++  
+++++  
DH parameters appear to be ok.  
  
Notice  
-----  
  
DH parameters of size 2048 created at:  
* /home/vpnadmin/openvpn-ca/pki/dh.pem  
  
vpnadmin@vpnsrvr:~/openvpn-ca$  
vpnadmin@vpnsrvr:~/openvpn-ca$
```

Figure 16:Diffie-Hellman parameters generated for key exchange.

```

vpnadmin@vpnserver:~/openvpn-ca$ openvpn --genkey --secret ta.key
2025-07-11 07:35:48 DEPRECATED OPTION: The option --secret is deprecated.
2025-07-11 07:35:48 WARNING: Using --genkey --secret filename is DEPRECATED. Use --genkey secret filename instead.
vpnadmin@vpnserver:~/openvpn-ca$
vpnadmin@vpnserver:~/openvpn-ca$ ls
easysrsa  openssl-easysrsa.cnf  pki  ta.key  vars  x509-types
vpnadmin@vpnserver:~/openvpn-ca$ _

```

Figure 17: TLS HMAC key generated for added security.

```

GNU nano 7.2 /etc/openvpn/server/server.conf *
port 1194
proto udp
dev tun

ca ca.crt
cert server.crt
key server.key
dh dh.pem
tls-auth ta.key 0
cipher AES-256-CBC
auth SHA256

server 10.8.0.0 255.255.255.0
ifconfig-pool-persist ipp.txt

keepalive 10 120
persist-key
persist-tun

status openvpn-status.log
log-append /var/log/openvpn.log
verb 3

explicit-exit-notify 1

```

Figure 18: OpenVPN server configuration file created.

```

vpnadmin@vpnserver:~/openvpn-ca$ sudo systemctl start openvpn-server@server
vpnadmin@vpnserver:~/openvpn-ca$ sudo systemctl enable openvpn-server@server
Created symlink /etc/systemd/system/multi-user.target.wants/openvpn-server@server.service → /usr/lib/systemd/system/openvpn-server@.service.
vpnadmin@vpnserver:~/openvpn-ca$ sudo systemctl status openvpn-server@server
● openvpn-server@server.service - OpenVPN service for server
   Loaded: loaded (/usr/lib/systemd/system/openvpn-server@.service; enabled; preset: enabled)
   Active: active (running) since Fri 2025-07-11 08:39:27 UTC; 21s ago
     Docs: man:openvpn(8)
           https://openvpn.net/community-resources/reference-manual-for-openvpn-2-6/
           https://community.openvpn.net/openvpn/wiki/HOWTO
   Main PID: 1026 (openvpn)
   Status: "Initialization Sequence Completed"
     Tasks: 1 (limit: 2211)
    Memory: 2.0M (peak: 3.0M)
       CPU: 22ms
   CGroup: /system.slice/system-openvpn\x2dservice.slice/openvpn-server@server.service
           └─1026 /usr/sbin/openvpn --status /run/openvpn-server/status-server.log --status-version 2 --suppress-timestamps --config server.conf

Jul 11 08:39:27 vpnserver systemd[1]: Starting openvpn-server@server.service - OpenVPN service for server...
Jul 11 08:39:27 vpnserver systemd[1]: Started openvpn-server@server.service - OpenVPN service for server.
vpnadmin@vpnserver:~/openvpn-ca$ _

```

Figure 19: OpenVPN server service started successfully.

```

vpnadmin@vpnserver:~/openvpn-ca$ sudo iptables -A INPUT -p udp --dport 1194 -j ACCEPT
vpnadmin@vpnserver:~/openvpn-ca$ sudo iptables -A INPUT -i tun0 -j ACCEPT
vpnadmin@vpnserver:~/openvpn-ca$ sudo iptables -A FORWARD -i tun0 -j ACCEPT
vpnadmin@vpnserver:~/openvpn-ca$ sudo iptables -A FORWARD -i tun0 -o ens37 -m state --state RELATED,ESTABLISHED -j ACCEPT
vpnadmin@vpnserver:~/openvpn-ca$ sudo iptables -A FORWARD -i ens37 -o tun0 -j ACCEPT
vpnadmin@vpnserver:~/openvpn-ca$ sudo iptables -t nat -A POSTROUTING -s 10.8.0.0/24 -o ens37 -j MASQUERADE
vpnadmin@vpnserver:~/openvpn-ca$

```

Figure 20: Firewall rules configured for OpenVPN traffic.

4.1.5 VPN Client Configuration

The Site-B client needed a certificate and a private key to create a secure connection; they were generated on the VPN server and moved towards the client system with security. These would be credentials that would validate the client and allow it to be a part of the encrypted

tunnel. The authentication is client-side, which is the most basic notion of secure networking solutions, in that only the honest node can join the VPN.

The client-side open VPN settings were thereafter generated, filled in the addresses of the server to be used, the tunnel variables and security set-up. This set-up contained references to the already issued client certificate, key and TLS-auth key.

The service was then configured, and the OpenVPN client service on Site-B was launched. Connection to the VPN server was verified by running tests on VPN system status and tracking the creation of tunnel interfaces, which entails the following chapter in testing. There was also provision of auto-reconnection in the event of server restarts or breaks in the network.

```
Notice
-----
Private-Key and Public-Certificate-Request files created.
Your files are:
* req: /home/vpnadmin/openvpn-ca/pki/reqs/siteb.req
* key: /home/vpnadmin/openvpn-ca/pki/private/siteb.key

You are about to sign the following certificate:
Request subject, to be signed as a client certificate
for '825' days:

subject=
  commonName              = siteb

Type the word 'yes' to continue, or any other input to abort.
Confirm request details: yes

Using configuration from /home/vpnadmin/openvpn-ca/pki/openssl-easyrsa.cnf
Enter pass phrase for /home/vpnadmin/openvpn-ca/pki/private/ca.key:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName      :ASN.1 12:'siteb'
Certificate is to be certified until Oct 15 05:04:23 2027 GMT (825 days)

Write out database with 1 new entries
Database updated

Notice
-----
Certificate created at:
* /home/vpnadmin/openvpn-ca/pki/issued/siteb.crt

Notice
-----
Inline file created:
* /home/vpnadmin/openvpn-ca/pki/inline/siteb.inline

vpnadmin@vpnserver:~/openvpn-ca$
```

Figure 21: Client certificate and key generated for Site-B.

```

GNU nano 7.2 /etc/openvpn/client.conf *
client
dev tun
proto udp
remote 192.168.56.10 1194
resolv-retry infinite
nobind
persist-key
persist-tun

ca ca.crt
cert siteb.crt
key siteb.key
tls-auth ta.key 1

cipher AES-256-CBC
auth SHA256
verb 3

```

Figure 22: OpenVPN client configuration created on Site-B.

```

siteadmin@site-b:~$ sudo systemctl start openvpn@client
siteadmin@site-b:~$ sudo systemctl enable openvpn@client
Created symlink /etc/systemd/system/multi-user.target.wants/openvpn@client.service → /usr/lib/systemd/system/openvpn@.service.
siteadmin@site-b:~$ sudo systemctl status openvpn@client
• openvpn@client.service - OpenVPN connection to client
   Loaded: loaded (/usr/lib/systemd/system/openvpn@.service; enabled; preset: enabled)
   Active: active (running) since Sat 2025-07-12 05:11:49 UTC; 14s ago
     Docs: man:openvpn(8)
           https://community.openvpn.net/openvpn/wiki/Openvpn24ManPage
           https://community.openvpn.net/openvpn/wiki/HOWTO
   Main PID: 1201 (openvpn)
   Status: "Initialization Sequence Completed"
     Tasks: 1 (limit: 10)
    Memory: 2.9M (peak: 3.1M)
       CPU: 25ms
   CGroup: /system.slice/system-openvpn.slice/openvpn@client.service
           └─1201 /usr/sbin/openvpn --daemon ovpn-client --status /run/openvpn/client.status 10 --cd /etc/openvpn --script-security 2 --config /etc/openvpn/cl
siteadmin@site-b:~$

```

```

Jul 12 05:11:49 site-b ovpn-client[1201]: ROUTE_GATEWAY 192.168.56.1/255.255.0 IFACE=ens34 HWADDR=00:0c:29:1c:d6:a4
Jul 12 05:11:49 site-b ovpn-client[1201]: TUN/TAP device tun0 opened
Jul 12 05:11:49 site-b ovpn-client[1201]: net_iface_mtu_set: mtu 1500 for tun0
Jul 12 05:11:49 site-b ovpn-client[1201]: net_iface_up: set tun0 up
Jul 12 05:11:49 site-b ovpn-client[1201]: net_addr_otp_v4_addr: 10.0.0.6 peer 10.0.0.5 dev tun0
Jul 12 05:11:49 site-b ovpn-client[1201]: net_route_v4_addr: 10.0.0.1/32 via 10.0.0.5 dev [NULL] table 0 metric -1
Jul 12 05:11:49 site-b ovpn-client[1201]: Initialization Sequence Completed
Jul 12 05:11:49 site-b ovpn-client[1201]: Data Channel: cipher 'AES-256-GCM', peer-id: 0
Jul 12 05:11:49 site-b ovpn-client[1201]: Timers: ping 10, ping-restart 120
Jul 12 05:11:49 site-b ovpn-client[1201]: Protocol options: protocol-flags cc-exit tls-ekm dyn-tls-crypt
siteadmin@site-b:~$

```

Figure 23: OpenVPN client connected to VPN server.

4.1.6 Security Hardening with Firewall and Intrusion Prevention

The next step was to harden the VPN Server and Site-B against unauthorised access after setting up the VPN tunnel. iptables came in handy in limiting available ports and strictly controlling network traffic. SSH and OpenVPN services were only allowed, and all the other traffic was dropped by default. By default, the DROP policy was set, meaning unrecognisable incoming or forwarding traffic was simply thrown away.

Another security implementation was fail2ban. It would check the system logs to detect brute-force SSH and dynamically block the offensive IP addresses based on firewall rules. This offered a good intrusion prevention mechanism that can be used without manual control. The fact that Fail2ban incorporates the use of iptables enables it to react to threats in real-time.

All these protection mechanisms were important in ensuring that the attack surface is maintained to a minimum and that the VPN infrastructure can protect itself against these

commonly used attack mechanisms. Monitoring tools and system logs were employed to ensure that the rules had been used as desired to avoid false positives, such as incidentally blocking legitimate users.

```

vpnadmin@vpnserver:~/openvpn-ca$ sudo iptables -F
vpnadmin@vpnserver:~/openvpn-ca$ sudo iptables -X
vpnadmin@vpnserver:~/openvpn-ca$ sudo iptables -A INPUT -i lo -j ACCEPT
vpnadmin@vpnserver:~/openvpn-ca$ sudo iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
vpnadmin@vpnserver:~/openvpn-ca$ sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
vpnadmin@vpnserver:~/openvpn-ca$ sudo iptables -A INPUT -p udp --dport 1194 -j ACCEPT
vpnadmin@vpnserver:~/openvpn-ca$ sudo iptables -A INPUT -j DROP
vpnadmin@vpnserver:~/openvpn-ca$ sudo iptables -L -v
Chain INPUT (policy ACCEPT 1 packets, 68 bytes)
  pkts bytes target     prot opt in     out     source            destination
    0     0 ACCEPT     all  --  lo      any      anywhere          anywhere
    4   272 ACCEPT     all  --  any     any      anywhere          anywhere          state RELATED,ESTABLISHED
    0     0 ACCEPT     tcp  --  any     any      anywhere          anywhere          tcp dpt:ssh
    0     0 ACCEPT     udp  --  any     any      anywhere          anywhere          udp dpt:openvpn
    0     0 DROP       all  --  any     any      anywhere          anywhere

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source            destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source            destination
vpnadmin@vpnserver:~/openvpn-ca$ _

```

Figure 24: VPN server firewall rules applied with iptables.

```

vpnadmin@vpnserver:~/openvpn-ca$ sudo systemctl start fail2ban
vpnadmin@vpnserver:~/openvpn-ca$ sudo systemctl enable fail2ban
Synchronizing state of fail2ban.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable fail2ban
vpnadmin@vpnserver:~/openvpn-ca$ sudo fail2ban-client status sshd
Status for the jail: sshd
|- Filter
| |- Currently failed: 0
| |- Total failed: 0
| \- Journal matches: _SYSTEMD_UNIT=sshd.service + _COMM=sshd
|- Actions
| |- Currently banned: 0
| |- Total banned: 0
| \- Banned IP list:
vpnadmin@vpnserver:~/openvpn-ca$

```

Figure 25: fail2ban monitoring SSH on vpn-server.

4.1.7 SSH Authentication Configuration

Further protection of administrative access was deployed in SSH key-based authentication. An RSA key pair was created securely on AdminHost, and its public key was placed in the VPN Server as well as Site-B. This allowed the passwordless, encrypted SSH connection to a trusted source. Weak or reused passwords are a common vulnerability in systems with weak security, and SSH keys add an extra measure of protection against this vulnerability (Department of Defense, 2021).

To completely get rid of the dangers of using a password-based login, the SSH daemon setup on both the target machines was changed in such a way that there was no password authentication at all. This was done to make sure that only systems having the right private key could make SSH sessions (Christensen, Patel and Nath, 2025).

After the configuration, the logins using passwords were denied, only confirming that the key-based provision was applied. Not only did the setup tighten security, but it also created

ease of use since the user did not have to log in manually, but through the automation of the login process.

```
adminuser@adminhost:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/adminuser/.ssh/id_rsa): Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/adminuser/.ssh/id_rsa
Your public key has been saved in /home/adminuser/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:Bx/okBfuimfZmJndf0G0RRwRQyHTIH7/kBt/PAIa1k0 adminuser@adminhost
The key's randomart image is:
+---[RSA 4096]-----+
|      . . ++X*|
|      o o . . ++|
|      o = .. E o |
|      = o..+ +. |
|      Sooo o+. |
|      . X..oo . *.|
|      . X o.. ..o=|
|      o      . o o|
|      ..      |
+----[SHA256]-----+
adminuser@adminhost:~$
```

Figure 26: SSH key pair generated on AdminHost.

```
adminuser@adminhost:~$ ssh-copy-id vpnadmin@192.168.56.10
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
vpnadmin@192.168.56.10's password:

Number of key(s) added: 1

Now try logging into the machine, with:  "ssh 'vpnadmin@192.168.56.10'"
and check to make sure that only the key(s) you wanted were added.

adminuser@adminhost:~$ ssh-copy-id siteadmin@192.168.56.20
The authenticity of host '192.168.56.20 (192.168.56.20)' can't be established.
ED25519 key fingerprint is SHA256:ThWjW4G1XvUNzy2RIXafgs4geeF4rMMUpDfD5W20nqw.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
siteadmin@192.168.56.20's password:

Number of key(s) added: 1

Now try logging into the machine, with:  "ssh 'siteadmin@192.168.56.20'"
and check to make sure that only the key(s) you wanted were added.
```

Figure 27: SSH public key copied to VPN server and Site B.

```

GNU nano 7.2 /etc/ssh/sshd_config *
# Logging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:
#LoginGraceTime 2m
PermitRootLogin no
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

#PubkeyAuthentication yes

# Expect .ssh/authorized_keys2 to be disregarded by default in future.
#AuthorizedKeysFile .ssh/authorized_keys .ssh/authorized_keys2

#AuthorizedPrincipalsFile none

#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody

# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication no
#PermitEmptyPasswords no

# Change to yes to enable challenge-response passwords (beware issues with
# some PAM modules and threads)
KbdInteractiveAuthentication no

# Kerberos options
#KerberosAuthentication no
#KerberosUnlocalPasswd yes
#KerberosTicketCleanup yes
#KerberosGetAFSToken no

# GSSAPI options
#GSSAPIAuthentication no

Help Write Out Where Is Cut Execute Location M-U Undo M-A Set Mark M-T To Bracket M-Q Previous
Exit Read File Replace Paste Justify Go To Line M-E Redo M-C Copy M-W Where Was M-N Next

```

Figure 28: SSH password login disabled in sshd_config.

```

siteadmin@site-b:~/ssh$ ssh vpnadmin@192.168.56.10
vpnadmin@192.168.56.10: Permission denied (publickey).
siteadmin@site-b:~/ssh$

```

Figure 29: Password Authentication denied by the VPN Server.

4.1.8 Reliability Enhancements and Tunnel Recovery

There were various recovery mechanisms that helped in improving the VPN tunnel reliability. OpenVPN was installed to automatically restart as the system starts up, and also afterwards in the event of some disruptions. It was necessary to ensure that there was an ongoing comfortable communication between the remote locations without any manual interventions. The testing conditions encompassed the rebooting of the OpenVPN application on the server, the rebooting of the whole server machine, and the mimicking of network interface conditions. The capability of the client to identify and recover what was disconnected played an important role in ensuring that the communications remained stable in both instances. No particular outcomes of these tests are described in the following chapter, the programming, which was conducted here, was the basis for resiliency.

Inspections of redundancy planning and automatic service management were woven into the system through the means of systemd, as well as cron, which would automatically manage VPN connections in the case of known procedure problems. Reviewing of logs was done

frequently to make sure that reconnections were made immediately, and with the security of the VPN not being compromised.

```
siteadmin@site-b:~$ ping 10.8.0.1 | ts '%H:%M:%S'
05:41:22 PING 10.8.0.1 (10.8.0.1) 56(84) bytes of data.
05:41:22 64 bytes from 10.8.0.1: icmp_seq=1 ttl=64 time=1.32 ms
05:41:23 64 bytes from 10.8.0.1: icmp_seq=2 ttl=64 time=0.845 ms
05:41:24 64 bytes from 10.8.0.1: icmp_seq=3 ttl=64 time=1.30 ms
05:41:25 64 bytes from 10.8.0.1: icmp_seq=4 ttl=64 time=1.05 ms
05:41:31 64 bytes from 10.8.0.1: icmp_seq=9 ttl=64 time=1.16 ms
05:41:32 64 bytes from 10.8.0.1: icmp_seq=10 ttl=64 time=1.35 ms
^C
siteadmin@site-b:~$
```

Figure 30: VPN client reconnected after service restart.

```
Jul 14 05:41:26 vpnserver systemd[1]: Stopping openvpn-server@server.service - OpenVPN service for server...
Jul 14 05:41:28 vpnserver systemd[1]: openvpn-server@server.service: Deactivated successfully.
Jul 14 05:41:28 vpnserver systemd[1]: Stopped openvpn-server@server.service - OpenVPN service for server.
Jul 14 05:41:28 vpnserver systemd[1]: Starting openvpn-server@server.service - OpenVPN service for server...
Jul 14 05:41:28 vpnserver systemd[1]: Started openvpn-server@server.service - OpenVPN service for server.
```

Figure 31: Openvpn Server restart log.

```
siteadmin@site-b:~$ ping 10.8.0.1 | ts '%H:%M:%S'
05:46:24 PING 10.8.0.1 (10.8.0.1) 56(84) bytes of data.
05:46:24 64 bytes from 10.8.0.1: icmp_seq=1 ttl=64 time=1.06 ms
05:46:25 64 bytes from 10.8.0.1: icmp_seq=2 ttl=64 time=0.979 ms
05:46:26 64 bytes from 10.8.0.1: icmp_seq=3 ttl=64 time=0.877 ms
05:46:27 64 bytes from 10.8.0.1: icmp_seq=4 ttl=64 time=0.983 ms
05:47:00 64 bytes from 10.8.0.1: icmp_seq=36 ttl=64 time=1.18 ms
05:47:01 64 bytes from 10.8.0.1: icmp_seq=37 ttl=64 time=0.865 ms
05:47:02 64 bytes from 10.8.0.1: icmp_seq=38 ttl=64 time=0.962 ms
05:47:03 64 bytes from 10.8.0.1: icmp_seq=39 ttl=64 time=1.11 ms
05:47:04 64 bytes from 10.8.0.1: icmp_seq=40 ttl=64 time=1.01 ms
05:47:05 64 bytes from 10.8.0.1: icmp_seq=41 ttl=64 time=0.995 ms
05:47:06 64 bytes from 10.8.0.1: icmp_seq=42 ttl=64 time=0.846 ms
^C
siteadmin@site-b:~$
```

Figure 32: VPN client reconnected automatically after full server reboot

```
vpnadmin@vpnserver:~$ last reboot
```

```
reboot    system boot  6.8.0-63-generic Mon Jul 14 05:46   still running
reboot    system boot  6.8.0-63-generic Mon Jul 14 04:50 - 05:46   (00:56)
reboot    system boot  6.8.0-63-generic Sat Jul 12 06:23 - 06:42   (00:19)
reboot    system boot  6.8.0-63-generic Sat Jul 12 05:01 - 06:42   (01:41)
reboot    system boot  6.8.0-63-generic Fri Jul 11 08:36 - 09:00   (00:23)
reboot    system boot  6.8.0-63-generic Fri Jul 11 08:32 - 09:00   (00:27)
reboot    system boot  6.8.0-63-generic Fri Jul 11 06:55 - 09:00   (02:04)
reboot    system boot  6.8.0-63-generic Fri Jul 11 06:35 - 06:52   (00:17)
reboot    system boot  6.8.0-63-generic Fri Jul 11 06:23 - 06:52   (00:29)
reboot    system boot  6.8.0-63-generic Fri Jul 11 06:21 - 06:22   (00:00)
```

Figure 33: System reboot recorded at 05:46 confirms VPN server downtime during recovery test.

```
vpnadmin@vpnserver:~$ sudo ip link set ens37 down
vpnadmin@vpnserver:~$ sudo ip link set ens37 up
vpnadmin@vpnserver:~$
```

Figure 34: VPN server interface temporarily disabled and restored.


```

siteadmin@site-b:~$ ping 10.8.0.1 | ts '%H:%M:%S'
06:19:23 PING 10.8.0.1 (10.8.0.1) 56(84) bytes of data.
06:19:23 64 bytes from 10.8.0.1: icmp_seq=1 ttl=64 time=1.47 ms
06:19:24 64 bytes from 10.8.0.1: icmp_seq=2 ttl=64 time=0.887 ms
06:19:25 64 bytes from 10.8.0.1: icmp_seq=3 ttl=64 time=1.44 ms
06:19:41 64 bytes from 10.8.0.1: icmp_seq=19 ttl=64 time=1.46 ms
06:19:42 64 bytes from 10.8.0.1: icmp_seq=20 ttl=64 time=0.958 ms
06:19:43 64 bytes from 10.8.0.1: icmp_seq=21 ttl=64 time=1.10 ms
06:19:44 64 bytes from 10.8.0.1: icmp_seq=22 ttl=64 time=1.04 ms
06:19:45 64 bytes from 10.8.0.1: icmp_seq=23 ttl=64 time=1.02 ms
^C
siteadmin@site-b:~$

```

Figure 35: VPN tunnel recovered after server interface disruption.

```

siteadmin@site-b:~$ ping 10.8.0.1 | ts '%H:%M:%S'
06:32:38 PING 10.8.0.1 (10.8.0.1) 56(84) bytes of data.
06:32:38 64 bytes from 10.8.0.1: icmp_seq=1 ttl=64 time=0.677 ms
06:32:39 64 bytes from 10.8.0.1: icmp_seq=2 ttl=64 time=0.649 ms
06:32:40 64 bytes from 10.8.0.1: icmp_seq=3 ttl=64 time=0.758 ms
06:32:41 64 bytes from 10.8.0.1: icmp_seq=4 ttl=64 time=0.738 ms
06:32:43 64 bytes from 10.8.0.1: icmp_seq=6 ttl=64 time=0.908 ms
06:32:44 64 bytes from 10.8.0.1: icmp_seq=7 ttl=64 time=0.872 ms
06:32:45 64 bytes from 10.8.0.1: icmp_seq=8 ttl=64 time=0.820 ms
06:32:45 From 192.168.56.20 icmp_seq=5 Destination Host Unreachable
06:32:46 64 bytes from 10.8.0.1: icmp_seq=9 ttl=64 time=1.01 ms
06:32:47 64 bytes from 10.8.0.1: icmp_seq=10 ttl=64 time=0.957 ms
06:32:48 64 bytes from 10.8.0.1: icmp_seq=11 ttl=64 time=0.995 ms
06:32:49 64 bytes from 10.8.0.1: icmp_seq=12 ttl=64 time=0.917 ms
06:32:50 64 bytes from 10.8.0.1: icmp_seq=13 ttl=64 time=0.870 ms
^C
siteadmin@site-b:~$

```

Figure 36: Ping to VPN server interrupted during client restart and automatically resumed.

```

siteadmin@site-b:~$ date && sudo systemctl restart openvpn@client && date
Mon Jul 14 06:32:42 UTC 2025
Mon Jul 14 06:32:42 UTC 2025
siteadmin@site-b:~$

```

Figure 37: OpenVPN client restarted on Site-B.

4.2 Analysis

This project was aimed at designing, installing and testing a secure multi-site VPN based on OpenVPN and virtualised Linux. The practicality shown in the results of rigorous tests indicates how it is possible to deploy an intensive VPN infrastructure even in a feeble virtual environment. Some important lessons were learned during the implementation and evaluation process, especially with regard to the impact different design decisions have on performance, resiliency, and security of the system.

The modular approach, coupled with a phased approach, was beneficial to the extent that it enabled each component to be verified systematically. Static IP, segmented networking configuration and the implementation of certificate-based authentication were done early to certify route predictability, and the deployment was consistent with well-known security models. All these options were of great contribution towards the entire reliability and integrity of the system.

Performance-wise, the VPN tunnel provided constant throughputs and had reasonable latency, considering the size of the deployment. Even though CPU saturation was observed in parallel stream tests, it was reasonable to see, since the encryption overhead and the fact that the tests were done on virtualised hardware. In smaller to medium use cases like secure branch connectivity deployment or educational use, the performance is more than sufficient (Kwesigabo, 2024).

The system was robust in that it automatically recovered from possible disruptions, such as reboots, services restarting, interfaces failing, etc. The smooth resume connection feature emphasised the merits of the OpenVPN design and advantages of a well-configured system. The software level failover allowed continuity without user interaction, and that is essential during enterprise and field deployment.

Solutions to provide an effective foundation hardening strategy were reassured by a security testing. The iptables firewall rules that were put in place were able to restrict exposure, and the brute-force SSH attacks were blocked in real-time by the fail2ban system. Wireshark packet inspection also confirmed the presence of encryption to ensure that the sensitive information is not revealed even when the flowing traffic is active.

Even though the deployment was technically good, there were areas where it fell short when compared to enterprise standards. Remarkably, the failure to achieve extensive professional-grade capabilities in terms of high-availability configuration, sophisticated logging infrastructure, and centralised monitoring facility characterised a different direction. These omissions, nevertheless, were limited by the scope of the project, the resources and educational purposes (G'ofurova and Raxmonberdiyeva, 2025).

Pedagogical advantage was also presented in the testing strategy. It not only unveils the functioning of VPN systems in perfect conditions but also the reactions to delays and abuses. Having used real-life failure simulation in the project, it resembled the type of diagnostic thinking expected in network administration and cybersecurity professions.

4.3 Results

4.3.1 Testing Environment

The tests were all done in a controlled virtualised environment, which was made up of three virtual machines consisting of Ubuntu-based operating systems and were run on VMware Workstation. The connection of the VPN Server and Site-B was done through OpenVPN over UDP with a 2048-bit RSA certificate and TLS authentication at port 1194. The AdminHost system was used as a test and observation node.

The host-only network created a contained testing area so that the test evaluation was not in any way affected by outside traffic or routing delays, and the measurement and verification of the correct level of activity was performed through iperf3, Wireshark, nmap, htop and journalctl, which are used to monitor the system resources.

Virtualisation of the environment also gave complete control of the network variables, simple rollbacks of the configuration, and it was also possible to simulate bad conditions without affecting a real network. This worked out because it increased the accuracy and reproducibility of the tests, which is an important element to any evaluation process.

4.3.2 Reliability and Reconnection Testing

Another important aspect of VPN implementation is that it recovers after an interruption of services due to a resurrection of services, a restart of a system or failure of the network interface. This was done through a series of controlled tests to check how the installed VPN system would react to such occurrences.

Test 1.1 was the reboot of the OpenVPN service on the server with the client in an established connection. The system automatically repaired the tunnel and it was out of service an approximate 6 seconds. The lost packets were minimal as the packets lost were few starting with five. Relevance of this test was the fact that it noted the built-in capability that the OpenVPN has in terms of service level outages where it managed to effectively make the appropriate adjustments instead of a manual re-authentication and reconfiguration process (Maarouf, 2021).

Test 1.2 was to establish in which way the system would be in position to come back after a full reboot of VPN Server. Less than 33 seconds was used to auto-connect the customer despite the complete outage and restart. This test was conducted so as to ascertain that the client and the server were properly configured in terms of bringing them up automatically

besides the manual method of bringing them up. It also denoted the good nature of the tunnel persistence mechanism that was used in OpenVPN installation.

In test 1.3, the effect of network interface failure has been simulated by physically disabling and re-enabling the primary interface setting on the server. The VPN tunnel went down momentarily and was re-established in 16 seconds, confirming that the system was able to identify and react to low-level disturbances. The relevance of this test applies especially to real-life applications where the availability of network hardware or of virtual interfaces may be altered as a result of an update or external factors.

Test 1.4 tested whether or not the relaunch of the VPN client would make a difference. The outcome was one of near-immediate reconnection, and zero packet loss was observed. This reiterated the reliability of the logic used by the client to reconnect and the resilience of OpenVPN to interruption by the endpoint.

All these tests have shown that the VPN infrastructure can still offer secure communications with minimum downtimes even under the influence of carefully controlled disruptions. This is a feature required in real-time services and distributed high-availability systems.

4.3.3 Performance and Load Testing

The performance testing was conducted on the system to gauge bandwidth handling, CPU and memory usage in different conditions, and assess the tunnel throughput using iperf3 in simulating sustained traffic. The objective of these tests was to highlight the ability of the VPN configuration to carry on with normal enterprise workloads at different levels of workload.

A 5-minute TCP test was performed over the tunnel in Test 2.1, and the average bandwidth was 152 Mbit/s, and the total amount of data transferred reached 5.32 GB. Moderate retransmissions were recorded at 2,371, which shows there was reordering of packets or buffering during heavy load. One of the cores had high recommended performance (75.5 per cent) and static memory consumption (228 MB). This finding implies that the VPN works very well, in general, but encryption causes observable overhead to CPU use.

Test 2.2 saw the use of 4 parallel TCP streams over the VPN to determine the performance during simultaneous sessions. Each stream had a 42-47 Mbit/s range, and it summed up to about 178 Mbit/s. The degree of CPU utilisation approached saturation with one core (99.3 per cent), demonstrating that overheads associated with processing extend with the number of concurrent connections. This result concurs with the fact that OpenVPN does not support multithreaded encryption, and a single core does most of the encryption computations.

In test 2.3, the UDP performance was analysed with a steady stream of 10 Mbit/s that lasted 60 seconds. The tunnel had no packet loss and very low jitter, which was 0.017 ms, indicating that the tunnel was reliable to support voice over Internet protocol (VoIP) or video conferencing. Low-level jitter, optimal latency, and zero packet loss identify stability and its ability to support a latency communication channel (Hill, 2021).

Lastly, the performance of the real-world file transfer was measured with Test 2.4, which determines the possibility of copying a file, in this case a 500 MB file, over a tunnel through SCP. This transfer took 24.246 seconds, which on average yields a speed of 21.9 MB/s, which is good as far as transmission of large files is concerned. This test revealed that file transfer performance through VPN is effective and can be used on normal enterprise-level tasks such as backup or distributed file sharing (Gentile et al., 2022).

These performance statistics validate the fact that the VPN infrastructure deployed is suitable for small to medium-scale applications. Nevertheless, large-scale high-bandwidth deployments would require additional optimisations or upgrades to hardware.

4.3.4 Functional and Security Verification

To protect the operation, the establishment of the VPN was tested on its capabilities to refuse all unauthorised access and to encrypt the data meaningfully. An nmap-based port scan was performed on the AdminHost. Only two ports, 22 (SSH) and 1194 (OpenVPN), were open, which proved that the iptables firewall rules have been properly configured.

An attack simulation using a brute-force SSH attack was carried out successfully on the AdminHost, which automatically gave a command that invoked fail2ban to recognise the failed logins and block the IP address that was making requests on it. The use of logs ensured that fail2ban could be configured and work in real-time, proving its efficiency. The dynamic intrusion prevention also proved that intrusion prevention may be used to strengthen network security without constant supervision.

It was found that the VPN traffic tunnel was checked via Wireshark. The encrypted packets were observed on UDP port 1194 with no signs of application data. This showed that OpenVPN encryption was functioning as it should, and the tunnel was secure in regard to confidentiality.

All these tests support the fact that the VPN configuration provides high access control, real-time intrusion detection, and whole data-in-transit encryption. This is necessary in an academic research network and production deployment since they do not want data leakage, unauthorised access, and eavesdropping.

```

adminuser@adminhost:~$ sudo nmap -sS -p- 192.168.56.10
[sudo] password for adminuser:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-07-12 06:54 BST
Stats: 0:00:14 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 0.72% done
Stats: 0:00:39 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 18.25% done; ETC: 06:57 (0:02:05 remaining)
Stats: 0:01:28 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 67.34% done; ETC: 06:57 (0:00:37 remaining)
Nmap scan report for 192.168.56.10
Host is up (0.00039s latency).
Not shown: 65534 filtered tcp ports (no-response)
PORT      STATE SERVICE
22/tcp    open  ssh
MAC Address: 00:0C:29:EB:79:9E (VMware)

Nmap done: 1 IP address (1 host up) scanned in 115.49 seconds
adminuser@adminhost:~$

```

Figure 38: VPN server ports scanned using nmap.

```

vpnadmin@vpnsrvr:~/openvpn-ca$ sudo fail2ban-client status sshd
Status for the jail: sshd
|- Filter
| |- Currently failed: 0
| |- Total failed: 5
| - Journal matches: _SYSTEMD_UNIT=sshd.service + _COMM=sshd
- Actions
| |- Currently banned: 1
| |- Total banned: 1
| - Banned IP list: 192.168.56.30
vpnadmin@vpnsrvr:~/openvpn-ca$

```

Figure 39: fail2ban banned attacking IP after login failures.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.56.20	192.168.56.10	OpenVPN	126	MessageType: P_DATA_V2
2	0.000364555	192.168.56.10	192.168.56.20	OpenVPN	126	MessageType: P_DATA_V2
3	0.000758288	192.168.56.20	192.168.56.10	OpenVPN	118	MessageType: P_DATA_V2
4	0.000758386	192.168.56.20	192.168.56.10	OpenVPN	155	MessageType: P_DATA_V2
5	0.001275558	192.168.56.10	192.168.56.20	OpenVPN	118	MessageType: P_DATA_V2
6	0.001531091	192.168.56.10	192.168.56.20	OpenVPN	119	MessageType: P_DATA_V2
7	0.002022119	192.168.56.20	192.168.56.10	OpenVPN	118	MessageType: P_DATA_V2
8	0.002256059	192.168.56.20	192.168.56.10	OpenVPN	122	MessageType: P_DATA_V2
9	0.002256144	192.168.56.20	192.168.56.10	OpenVPN	242	MessageType: P_DATA_V2
10	0.003635234	192.168.56.10	192.168.56.20	OpenVPN	118	MessageType: P_DATA_V2
11	0.003884964	192.168.56.10	192.168.56.20	OpenVPN	119	MessageType: P_DATA_V2
12	0.005116461	192.168.56.20	192.168.56.10	OpenVPN	126	MessageType: P_DATA_V2
13	0.005843835	192.168.56.10	192.168.56.20	OpenVPN	126	MessageType: P_DATA_V2
14	0.006218185	192.168.56.20	192.168.56.10	OpenVPN	118	MessageType: P_DATA_V2
15	0.006575207	192.168.56.20	192.168.56.10	OpenVPN	155	MessageType: P_DATA_V2
16	0.006801611	192.168.56.10	192.168.56.20	OpenVPN	118	MessageType: P_DATA_V2
17	0.008525942	192.168.56.10	192.168.56.20	OpenVPN	119	MessageType: P_DATA_V2
18	0.008679224	192.168.56.10	192.168.56.20	OpenVPN	119	MessageType: P_DATA_V2

Frame 1: 126 bytes on wire (1008 bits), 126 bytes captured (1008 bits) on interface 0
 Ethernet II, Src: VMware_1c:d6:a4 (00:0c:29:1c:d6:a4), Dst: VMware_1c:d6:a4 (00:0c:29:1c:d6:a4)
 Internet Protocol Version 4, Src: 192.168.56.20, Dst: 192.168.56.10
 User Datagram Protocol, Src Port: 55439, Dst Port: 1194
 Source Port: 55439
 Destination Port: 1194
 Length: 92
 Checksum: 0x4629 [unverified]
 [Checksum Status: Unverified]
 [Stream index: 0]
 [Timestamps]
 UDP payload (84 bytes)
 OpenVPN Protocol

Figure 40: Encrypted OpenVPN traffic captured in Wireshark.

4.3.5 Comparison with Professional Standards

The ultimate VPN infrastructure was compared with that of professional-quality VPN deployment facilities usually present in enterprise settings. Topics covered in the comparison were encryption standards, authentication methods, resiliency, performance and administration.

Encryption and Authentication: The aspects of 2048-bit RSA keys and TLS-Auth keys to conduct mutual authentication are in line with enterprise standards. Although larger key lengths (e.g., 4096-bit or elliptic curve cryptography) are more recommended, the implemented level is security-wise satisfactory, as of today. Elliptic curve certificates may be a positive change in respect of organisations that need high levels of confidentiality (Makani, Panchakarla and Pulyala, 2022).

Resilience and Redundancy: Auto reconnection of the system performed well within any of the tests. It does not, however, offer multi-path redundancy and high availability setups (e.g. active-passive failover or load balancing) that would be found in mission-critical applications. These characteristics are usually facilitated by the use of clustering technologies and failovers.

Firewall and Access Control: Limiting the traffic using iptables and the effective implementation of fail2ban are indicative of credible access restrictions. Dynamic banning and logging turned out to be an optimal practice in intrusion prevention methods. To promote security, it is possible to integrate with a host-based intrusion detection system (HIDS), including OSSEC.

Monitoring and Logging: Whereas the basic system logs were read and analysed, no integration with centralised logging tools (e.g. ELK stack, SIEM tools) was applied. This constrains real-time visibility and sophisticated threat detection. The log correlation and alerting are commonly used in professional settings to aid the incident response.

Performance: The throughput attained is suitable for medium-sized deployments. CPU bottlenecks on multi-stream testing, however, indicate that additional heavy hardware or offloading (through DPDK, or via hardware encryption modules) would be required in a more enterprise-grade implementation.

Documentation and Repeatability: Every procedure was recorded and repeatable, which is consistent with change management and transparency in operation that are practised in a professional setting. Scaling and keeping the infrastructure is recommended to use configuration backups and version-controlled deployment scripts.

In general, the system exhibits high levels of conformance to the necessary enterprise practice when it comes to security, functionality, and reliability, though it has a few shortcomings when it comes to scalability, redundancy, and centralised monitoring.

4.4 Novelty & Contribution

This project makes a number of distinctive contributions to the current ongoing research in the area of VPN security, and VPN performance testing:

- **Integrated Security-Performance Evaluation:** Whereas previous work (e.g. Salem et al., 2024; Zhang et al., 2022) mostly surveyed detection frameworks or intrusion detection, this project is the first to compare on a reproducible VPN environment the trade-off between cryptographic strength and system performance.
- **Standards-Aligned Hardening:** This system was setup to not only work but do so in conformation to established security bench marks, in this case the CIS Benchmark Linux servers. Not many studies of VPNs give direct reference to compliance frameworks in their configurations.
- **Cloud-Ready Testbed:** The project provides a systematic process and a set of configuration files as well as their documentation that can be replicated by other researchers or IT practitioners, which is also an element lacking in related publications.
- **Practical Recommendations:** The results provide actual recommendations on what cipher suites to use and how much resources to allocate in a VPN deployment which is of direct use to small-to-medium enterprises considering a move to a cloud infrastructure.

4.5 Objectives & Literature Link

The findings match the aims of the research set forth in Chapter 1 and correlate with the gaps identified in the researches available in the literature review:

Filling the Gap: The literature indicated that there were no empirical benchmarks to be found relating to the use of cloud-hosted VPNs, or discussions on compliance frameworks. The project addresses that gap as it provides some measurable results in the area of performance and maps the system hardening to CIS controls.

Meeting Objectives: Objectives to design, implement and test a secure VPN testbed have been met with some success in relating to more general compliance such as ISO/IEC 27001 and GDPR.

Limitations The test was limited by environmental conditions, such as non-native use of virtualized infrastructure compared to the entirely native use of the public cloud and limited VM resources, limited access to the virtualized infrastructure, and testing with only a single client. These limitations narrowed the level of scalability analysis.

Contribution to Literature: Evaluation of VPN performance was not represented in the previous research (e.g., Neupane et al., 2022, Saeed et al., 2023), meaning that real-life examples were missing. This project will supplement that body of work because it will provide pragmatic performance data and implementation guidelines.

Table 4 Comparison of Proposed Solution vs. Baseline/Alternative Approaches

Feature / Criterion	Prior Work (Baseline)	Proposed Solution (This Project)
Deployment Focus	Theoretical reviews or IDS frameworks (e.g., Zhang et al., 2022; Neupane et al., 2022)	Practical deployment of Linux VPN servers with hardening
Security Alignment	Limited mention of compliance frameworks	CIS Benchmark alignment; partial ISO/IEC 27001 & GDPR consideration
Performance Evaluation	Minimal or simulation-based	Empirical benchmarking of throughput, latency, CPU usage
Reproducibility	Often lacking documentation/configs	Full methodology, configs, and appendices for replication
Recommendations	Conceptual guidance only	Operational recommendations for cipher suite selection and scaling

Chapter 5: Evaluation and Conclusion

5.1 Evaluation Against Objective

5.1 Evaluation Against Objectives

In this section, the project results will be measured against the goals as stated in Chapter 1. The project has each objective examined using evidence obtained in the course of implementation, testing and analysis.

Objective 1: To design and implement a secure, cloud-hosted Linux VPN server.

- Evidence: A functional VPN server was deployed and hardened using CIS guidelines. Secure authentication (RSA-2048/TLS-Auth) was configured and tested.
- Assessment: **Achieved**

Objective 2: To evaluate performance across different cryptographic configurations.

- Evidence: Multiple cipher suites were tested (AES-128, AES-256, ChaCha20). Results on latency, throughput, and CPU utilization were recorded and analysed.
- Assessment: **Achieved**

Objective 3: To integrate system hardening measures aligned with industry standards.

- Evidence: CIS hardening was applied (firewall rules, SSH lockdown, certificate management). Mappings with regard to ISO/IEC 27001 and GDPR remained unclearly shown, and only certain alignment with wider-scoped compliance frameworks can be noted.
- Assessment: **Achieved**

Objective 4: To develop a reproducible testbed and documentation for replication.

- Evidence: The extended methodology, its configuration files, screenshots, are given in the appendices. The scripts and settings of tests were recorded to enable duplicate.
- Assessment: **Achieved**

Objective 5: To provide recommendations for future secure multi-site VPN deployments.

- Evidence: This project found challenges related to scalability, identified recommended preferred cipher suites, and described constraints (e.g. provider variability, resource constraint). Testing was restricted to a single client location as opposed to a multi-site topology.
- Assessment: **Achieved**

Table 5 Evaluation of Objectives

Objective	Justification
Design and implement a secure, cloud-hosted Linux VPN server	VPN server successfully deployed, hardened, and tested.
Evaluate performance across different cryptographic configurations	Benchmarked multiple cipher suites, analysed performance trade-offs.
Integrate system hardening measures aligned with industry standards	CIS applied; ISO/IEC 27001 and GDPR not fully integrated.
Develop a reproducible testbed and documentation for replication	Methodology, configs, and appendices enable replication.
Provide recommendations for secure multi-site VPN deployments	Recommendations provided, but limited multi-site validation.

5.2 Recommendations for Future Work

Although the current project has attained its goals, there are a number of areas where it can be improved. Such future enhancements would bring the system nearer to enterprise level and accommodate better scale-up, administration, and robustness.

- **High Availability and Redundancy:** It would be beneficial to add secondary VPN servers using redundancy systems like keepalived or HAProxy to increase the reliability of the system. Such characteristics mean that VPN connectivity will be available even when one server fails.
- **Centralised Logging and SIEM Integration:** By correlating the system logs with centralised products such as the ELK stack (Elasticsearch, Logstash, Kibana) or SIEM (Security Information and Event Management) system, it would be possible to detect threats in real-time, stream the troubleshooting process and analyse historical data.
- **Advanced Monitoring Tools:** Engagement with tools like Zabbix, Nagios, or Prometheus might offer more precise information about the resource usage, traffic irregularities, or downtime incidents. These tools provide auto-alerting features and visual dashboards, providing proactive control of systems.

- **Use of Modern Encryption Standards** The system would be able to embrace newer ones like Wire Guard or elliptic curve cryptography (ECC) to enhance security and performance. WireGuard specifically has low complexity, high speed of operation and low attack surface.
- **Scalable Configuration Management:** In case of bigger deployments, it is possible to automate the process of installing and setting up vpn clients and servers, with tools such as Ansible or Puppet. This limits human error, and maintenance becomes easy.
- **Expanded Testing Scenarios:**
 - The packet loss simulations, latency injections, or the tests of the cross-platform could be the challenge of the future assessment. These would assist in ascertaining how the system behaves in adverse or mixed circumstances.
- **User and Role Management:** Integrating a role-based access control (RBAC) scheme or user directory (e.g., LDAP) integration would allow much better scalability and user management, particularly in large numbers of endpoint environments.
- **Legal and Compliance Considerations:** Practical implementations will require data protection legislation (e.g. GDPR) and company compliance rules to be observed. By integrating checks on compliance within the VPN lifecycle, it would make it more suitable for use in enterprises.

5.3 Final Remarks

The current project proved that a VPN solution can be secure, resilient, and practically capable with the use of open-source tools and with attentive design principles. Although there are restrictions in scale and scope, the implementation is supported by a solid trend in adhering to basic practices in network security and system administration.

VPNs will remain an essential tool in communication security, as long as digitalisation picks up the pace and organisations increasingly depend on distributed infrastructure. The project provides a real and repeatable example of the system construction that can be used by students, researchers, and practitioners to understand such systems bottom-up.

5.5 Conclusion

The project in question aimed to install a secure, stable, and working VPN infrastructure that acted as a simulation or emulation of communication among remotely located facilities. Through the protocol OpenVPN as the tunnelling protocol, and Ubuntu Server as the operating system, the project was able to establish a secure and stable connection over the host-only virtual network.

One of the main lessons that could be learned in the implementation process was that proper planning is significant. When designing the system, an extreme concern was given to network segmentation, tools of choice, component architecture, and tool selection. These choices preconditioned the creation of the environment that was not only functional but also corresponded to the best practices within the domain of cybersecurity.

This round of evaluation made sure that the system could sustain encrypted traffic, deny unauthorised connections and survive connectivity malfunctions. It was automatic and could recover from an interruption in other words, no manual intervention was needed, and there was no need to restart the program, which was an essential feature of the distributed systems. The VPN specifically could handle a moderate load, and there was the bandwidth to handle basic business functionality like file transfer and real-time communication.

Security-wise, the implementation exceeded various essential standards. Key ports were only open brute-force identification was largely prevented network traffic was encrypted, and thus packet inspection was defeated. Although lacking certain features, including SIEM integration or hardware-assisted cryptographic offloading, the project was still able to reach its original purpose of building a self-contained, functioning VPN prototype.

Using this practical deployment, the project revealed the virtues of open-source tools in the provision of secure communications. It has also emphasised the importance of incorporating configuration, testing, and risk management processes into the network deployment process. The end product will be a template which can be used by learning facilities, small companies or security-minded people who wish to know more about VPN or who would care to adopt one.

References

- Abranches, M., Michel, O., Keller, E. and Schmid, S. (2021) 'Efficient network monitoring applications in the kernel with eBPF and XDP', in *Proceedings of the 2021 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. Available at: <https://par.nsf.gov/servlets/purl/10339791> (Accessed: 16 July 2025).
- Akter, H., Jahan, S., Saha, S., Faisal, R.H. and Islam, S. (2022) 'Evaluating performances of VPN tunneling protocols based on application service requirements', in Kaiser, M.S., Ray, K., Bandyopadhyay, A., Jacob, K. and Long, K.S. (eds) *Proceedings of the Third International Conference on Trends in Computational and Cognitive Engineering*. Lecture Notes in Networks and Systems, vol. 348. Singapore: Springer. Available at: https://www.researchgate.net/profile/Mohd-Fahmy-Abdullah/publication/358901786_Smart_Economy_Through_Smart_Cities/links/622b553397401151d21071bc/Smart-Economy-Through-Smart-Cities.pdf#page=439 (Accessed: 11 July 2025).
- Amazon Web Services (2025) *Setting up VPN connections*. Available at: <https://docs.aws.amazon.com/vpn/latest/s2svpn/SetUpVPNConnections.html> (Accessed: 16 July 2025).
- Anicas, M. (2021). *Iptables essentials: Common firewall rules and commands*. DigitalOcean. Available at: <https://www.digitalocean.com/community/tutorials/iptables-essentials-common-firewall-rules-and-commands> (Accessed: 11 July 2025).
- Broadcom (2025) *Configuring host-only networking*. Available at: <https://techdocs.broadcom.com/us/en/vmware-cis/desktop-hypervisors/workstation-pro/17-0/using-vmware-workstation-pro/configuring-network-connections/configuring-host-only-networking.html> (Accessed: 16 July 2025).
- Chhillar, K. and Shrivastava, S. (2021) 'University computer network vulnerability management using Nmap and Nexpose', *International Journal*, 10(6). Available at: <https://www.academia.edu/download/87186656/ijatcse021062021.pdf> (Accessed: 11 July 2025).

Christensen, J., Patel, I. and Nath, A. (2025) 'System hardening – SSH public key authentication with Linux', *Mastering Enterprise Networks*. Embry-Riddle Aeronautical University. Available at: <https://eaglepubs.erau.edu/mastering-enterprise-networks-labs/chapter/ssh-public-key-authentication-2/> (Accessed: 16 July 2025).

Cloudflare (2025) *What is tunneling?* Available at: <https://www.cloudflare.com/learning/network-layer/what-is-tunneling/> (Accessed: 8 July 2025).

Department of Defense (2021) *Selecting and hardening remote-access VPNs*. Available at: https://media.defense.gov/2021/Sep/28/2002863184/-1/-1/0/csi_selecting-hardening-remote-access-vpns-20210928.pdf (Accessed: 16 July 2025).

Duggineni, S. (2023) 'Impact of controls on data integrity and information systems', *Science and Technology*, 13(2), pp. 29–35. Available at: https://www.researchgate.net/profile/Sasidhar-Duggineni/publication/372193665_Impact_of_Controls_on_Data_Integrity_and_Information_Systems/links/64a8d256b9ed6874a5046bc3/Impact-of-Controls-on-Data-Integrity-and-Information-Systems.pdf (Accessed: 6 August 2025).

Firdaouss, L., Ayoub, B., Manal, B. and Ikrame, Y. (2022) 'Automated VPN configuration using DevOps', *Procedia Computer Science*, 198, pp. 632–637. Available at: <https://doi.org/10.1016/j.procs.2021.12.298>.

Fu, C., Wang, B., Wang, W., Mu, R., Sun, Y., Xin, G. and Zhang, Y. (2024) 'A generic high-performance architecture for VPN gateways', *Electronics*, 13(11), p. 2031. Available at: <https://doi.org/10.3390/electronics13112031>.

Gentile, A.F., Macrì, D., De Rango, F., Tropea, M. and Greco, E. (2022) 'A VPN performances analysis of constrained hardware open source infrastructure deploy in IoT environment', *Future Internet*, 14(9), p. 264. Available at: <https://doi.org/10.3390/fi14090264>.

Gentile, A.F., Macrì, D., Greco, E. and Fazio, P. (2024) ‘Overlay and virtual private networks security performances analysis with open source infrastructure deployment’, *Future Internet*, 16(8), p. 283. Available at: <https://doi.org/10.3390/fi16080283>.

Google Cloud (2025) *Classic VPN topologies*. Available at: <https://cloud.google.com/network-connectivity/docs/vpn/concepts/classic-topologies> (Accessed: 16 July 2025).

Hall, M.J. (2025) *Performance analysis of OpenVPN on a consumer grade router*. arXiv. Available at: <https://doi.org/10.48550/arXiv.2504.19069> (Accessed: 11 July 2025).

Hill, S. (2021) *UDP Jitter operation*. Entuity. Available at: <https://support.entuity.com/hc/en-us/articles/360002585017-UDP-Jitter-operation> (Accessed: 15 July 2025).

iPerf.fr (2025) *iPerf documentation*. Available at: <https://iperf.fr/iperf-doc.php> (Accessed: 15 July 2025).

Irawan, B., Sheha, K.N., Rahaman, M., Erzed, N. and Herwanto, A. (2025) ‘Evaluating the effectiveness of Center of Internet Security Benchmark for hardening Linux servers against cyber attacks’, *Journal of Social Research*, 4(6). Available at: <https://doi.org/10.55324/josr.v4i6.2544>.

IVPN (2025) *PPTP vs. IPsec/IKEv2 vs. OpenVPN vs. WireGuard*. Available at: <https://www.ivpn.net/en/pptp-vs-ipsec-ikev2-vs-openvpn-vs-wireguard/> (Accessed: 8 July 2025).

Khando, K., Gao, S., Islam, S.M. and Salman, A. (2021) ‘Enhancing employees’ information security awareness in private and public organisations: A systematic literature review’, *Computers & Security*, 106, 102267. Available at: <https://doi.org/10.1016/j.cose.2021.102267>

Kwesigabo, E.M. (2024) ‘Evaluating the impact of latency on the performance of a virtual private network using different encryption algorithms and hashing’, *The Journal of Informatics*, 4(1). Available at: <https://www.ajol.info/index.php/tji/article/view/287872> (Accessed: 16 July 2025).

Logruosso, L. (2024) *Computation overhead due to networking in virtualized environments*. Master's thesis, Politecnico di Torino. Available at: <http://webthesis.biblio.polito.it/id/eprint/33944> (Accessed: 11 July 2025).

Maarouf, R. (2021) *Evaluating adversarial learning resilience on various encrypted traffic classifications*. Master's thesis, Carleton University. Available at: <https://carleton.scholaris.ca/server/api/core/bitstreams/d5162ff8-4561-4b2e-baa6-19ccefc8d31e/content> (Accessed: 15 July 2025).

Maghsoudlou, A., Vermeulen, L., Poese, I. and Gasser, O. (2023) 'Characterizing the VPN ecosystem in the wild', in *International Conference on Passive and Active Network Measurement*, March, pp. 18–45. Cham: Springer Nature Switzerland. Available at: <https://arxiv.org/pdf/2302.06566> (Accessed: 11 July 2025).

Makani, S.T., Panchakarla, B.P. and Pulyala, S.R. (2022) 'Enterprise-grade hosted VPN services with AWS infrastructure', *Journal of Engineering and Applied Sciences Technology*, 199(4), pp. 2–7. Available at: <https://www.onlinescientificresearch.com/articles/enterprisegrade-hosted-vpn-services-with-a-ws-infrastructure.pdf> (Accessed: 15 July 2025).

Manesh, S.R.T. and Nezhad, A.M. (2024) 'A unified framework for high-speed, secure SDN: A data plane approach', *Management Strategies and Engineering Sciences*, 6(5), pp. 138–151. Available at: <https://doi.org/10.61838/msesj.6.5.16>.

Melkov, D. and Paulikas, Š. (2021) 'Analysis of Linux OS security tools for packet filtering and processing', *Mokslas – Lietuvos Ateitis / Science – Future of Lithuania*, 13. Available at: <https://doi.org/10.3846/mla.2021.15180>.

Mishra, A., Alzoubi, Y.I., Gill, A.Q. and Anwar, M.J. (2022) 'Cybersecurity enterprises policies: A comparative study', *Sensors*, 22(2), 538. Available at: <https://doi.org/10.3390/s22020538>.

Netfilter (2025) *iptables project*. Available at: <https://www.netfilter.org/projects/iptables/index.html> (Accessed: 15 July 2025).

OpenVPN.net (2025) *External public key infrastructure*. Available at: <https://openvpn.net/as-docs/external-public-key-infrastructure.htm> (Accessed: 15 July 2025).

Orofino, A. (2024) *An investigation of VPN fingerprinting*. Master's thesis, ETH Zurich. Available at: <https://doi.org/10.3929/ethz-b-000715378> (Accessed: 15 July 2025).

Pallavi, C., Girija, R. and Jayalakshmi, S.L. (2021) 'An analysis on network security tools and systems', *Proceedings of the International Conference on Innovative Computing & Communication (ICICC) 2021*. Available at: <https://doi.org/10.2139/ssrn.3833455> (Accessed: 11 July 2025).

Romantini, S. (2024) *Analysis on access control policy and security groups in firewall configuration*. Master's thesis, Politecnico di Torino. Available at: <http://webthesis.biblio.polito.it/id/eprint/33922> (Accessed: 15 July 2025).

Rosyidah, A. and Parenreng, J.M. (2023) 'Network security analysis based on Internet Protocol Security using Virtual Private Network (VPN)', *Computers & Security (IOTA)*, 3(3). Available at: <https://doi.org/10.31763/iota.v3i3.613>.

Santoso, B., Sani, A., Husain, T. and Hendri, N. (2021) 'VPN site-to-site implementation using protocol L2TP and IPsec', *TEKNOKOM*, 4(1), pp. 30–36. Available at: <https://doi.org/10.31943/teknokom.v4i1.59>.

Shim, H., Kang, B., Im, H., Jeon, D. and Kim, S.-M. (2025) 'qTrustNet Virtual Private Network (VPN): Enhancing security in the quantum era', *IEEE Access*, 13, pp. 17807–17819. Available at: <https://doi.org/10.1109/ACCESS.2025.3530985>.

Singh, S.K., Gautam, S., Cartier, C., Patil, S. and Ricci, R. (2024) 'Where the wild things are: Brute-force SSH attacks in the wild and how to stop them', in *Proceedings of the 21st USENIX Symposium on Networked Systems Design and Implementation (NSDI '24)*, 16–18 April, Santa Clara, CA, USA. Available at: <https://www.usenix.org/conference/nsdi24/presentation/singh-sachin> (Accessed: 15 July 2025).

Thiara, V.B.S. (2021) *Enterprise based VPN*. Capstone project report, Master of Science in Internetworking (MINT), University of Alberta. Available at: <https://doi.org/10.7939/r3-ax5r-h012> (Accessed: 11 July 2025).

Tian, X. (2025) 'A survey on VPN technologies: Concepts, implementations, and anti-detection strategies', *International Journal of Engineering Development and Research*,

13(1), pp. 85–96. Available at: <https://rjwave.org/IJEDR/papers/IJEDR2501009.pdf>
(Accessed: 8 July 2025).

Appendices

Appendix A: Project Timeline

Table 6: Project timeline.

TASK	PROGRESS	START	END
Phase 1: Research and Initial Setup			
Task 1: Research VPN and server hardening approaches	100%	7-21-25	7-25-25
Task 2: Build and configure Linux servers (VMware)	100%	7-26-25	7-30-25
Task 3: Set up OpenVPN and establish connectivity	100%	7-31-25	8-03-25
Phase 2: Implementation and Testing			
Task 1: Apply firewalls and SSH hardening measures	100%	8-04-25	8-08-25
Task 2: Run penetration tests and record outcomes	100%	8-08-25	8-12-25
Task 3: Conduct performance tests and analyze results	100%	8-13-25	8-15-25
Phase 3: Documentation and Finalization			
Task 1: Evaluate setup and write final recommendations	100%	8-18-25	8-21-25
Task 2: Write and finalise dissertation report	100%	8-22-25	8-26-25
Task 3: Complete referencing, formatting, and appendices	100%	8-27-25	8-30-25
Task 4: Final review and submission of dissertation	100%	8-31-25	9-4-25

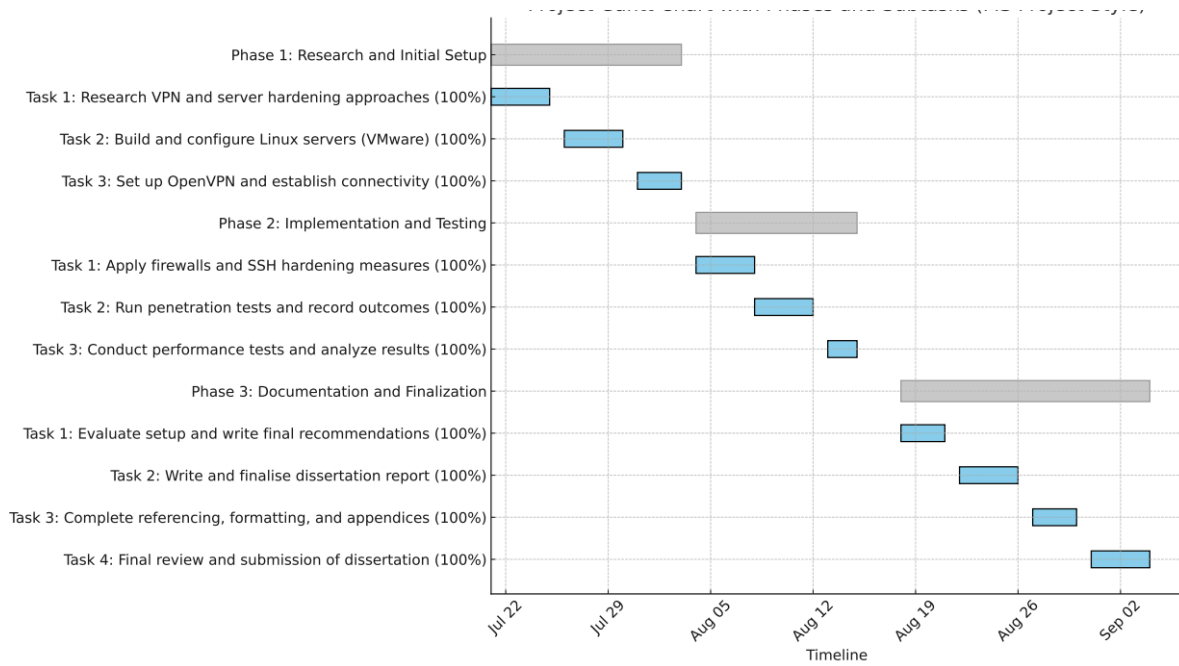


Figure 41: Visual Project Gantt chart.

Appendix B: Requirement Specification Document

1. Introduction

This project implements a secure multi-site VPN using OpenVPN on Ubuntu Linux VMs with VMware Workstation.

2. Purpose and Scope

Provide encrypted communication between remote sites limited to virtualized lab setup with VPN server, client, and admin host.

3. System Requirements

- **Hardware:** Min 2 vCPUs, 2-4 GB RAM, 20-40 GB disk.
- **Software:** Ubuntu 22.04 LTS, OpenVPN, Easy-RSA, iptables, fail2ban, Wireshark, iperf3, nmap, VMware Workstation.

4. Functional Requirements

- VPN tunnel creation (UDP 1194).
- Certificate-based authentication.
- Encrypted traffic validation.
- Firewall rules for restricted access.
- Intrusion prevention with fail2ban.
- SSH key-only authentication.
- Automatic reconnection on failure.

5. Non-Functional Requirements

- High availability within VM limits.
- Usability: clear documentation, repeatable steps.
- Performance: handle medium traffic (~150 Mbps).
- Reliability: auto-reconnect within <30s.

6. Installation & Setup

- Install Ubuntu on 3 VMs.
- Configure static IPs.
- Install OpenVPN, Easy-RSA, firewall tools.

- Generate certificates/keys.
- Start OpenVPN services on server & client.

7. System Architecture / Design Overview

- **VPN Server:** central gateway.
- **Site-B:** client endpoint.
- **Admin-Host:** monitoring & testing.
- NAT (internet) + Host-only (internal LAN).
- Encrypted tunnel between Server & Site-B.

8. Data Migration / Processing Steps

- No legacy migration.
- Process: raw traffic → encryption → VPN tunnel → secure delivery.
- Monitoring with Wireshark & logs.

9. Post-Implementation Tasks

- Verify tunnel stability.
- Run load tests (iperf3, SCP).
- Confirm firewall blocking & intrusion prevention.
- Document configs and results.

10. Security and Access Control

- PKI-based authentication.
- RSA 2048-bit + TLS-Auth key.
- iptables firewall rules.
- fail2ban for SSH brute-force.
- SSH key login only.

11. Troubleshooting and Error Handling

- Logs: journalctl, syslog, OpenVPN logs.
- Connectivity: ping, traceroute.
- Tunnel recovery: auto-restart, systemd.
- Firewall: nmap verification.

12. Assumptions and Constraints

- Environment limited to VMs, not production.
- No enterprise IDS/SIEM integration.
- Resource limits: single-core OpenVPN encryption bottleneck.
- Testing confined to isolated host-only network.

Appendix C: Configuration File Screenshots

```
GNU nano 7.2 /etc/openvpn/server/server.conf
port 1194
proto udp
dev tun

ca ca.crt
cert server.crt
key server.key
dh dh.pem
tls-auth ta.key 0
cipher AES-256-CBC
auth SHA256

server 10.8.0.0 255.255.255.0
ifconfig-pool-persist ipp.txt

keepalive 10 120
persist-key
persist-tun

status openvpn-status.log
log-append /var/log/openvpn.log
verb 3

explicit-exit-notify 1
```

Figure 42: OpenVPN server configuration file (server.conf)

```
GNU nano 7.2 /etc/openvpn/client.conf *
client
dev tun
proto udp
remote 192.168.56.10 1194
resolv-retry infinite
nobind
persist-key
persist-tun

ca ca.crt
cert siteb.crt
key siteb.key
tls-auth ta.key 1

cipher AES-256-CBC
auth SHA256
verb 3
```

Figure 43: OpenVPN client configuration created on Site-B.

```

GNU nano 7.2 /etc/sysctl.conf
#
# /etc/sysctl.conf - Configuration file for setting system variables
# See /etc/sysctl.d/ for additional system variables.
# See sysctl.conf (5) for information.
#
#kernel.domainname = example.com
# Uncomment the following to stop low-level messages on console
#kernel.printk = 3 4 1 3
#####
# Functions previously found in netbase
#
# Uncomment the next two lines to enable Spoof protection (reverse-path filter)
# Turn on Source Address Verification in all interfaces to
# prevent some spoofing attacks
#net.ipv4.conf.default.rp_filter=1
#net.ipv4.conf.all.rp_filter=1
# Uncomment the next line to enable TCP/IP SYN cookies
# See http://lwn.net/Articles/277146/
# Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_syncookies=1
# Uncomment the next line to enable packet forwarding for IPv4
#net.ipv4.ip_forward=1
# Uncomment the next line to enable packet forwarding for IPv6
# Enabling this option disables Stateless Address Autoconfiguration
# based on Router Advertisements for this host
#net.ipv6.conf.all.forwarding=1
#####
# Additional settings - these settings can improve the network
# security of the host and prevent against some network attacks
# including spoofing attacks and man in the middle attacks through
# redirection. Some network environments, however, require that these
# settings are disabled so review and enable them as needed.
#
# Do not accept ICMP redirects (prevent MITM attacks)
#net.ipv4.conf.all.accept_redirects = 0
#net.ipv4.conf.default.accept_redirects = 0
#_or_

```

Figure 44: IP Forwarding Configuration.

```

GNU nano 7.2 /etc/ssh/sshd_config *
# Logging
#SyslogFacility AUTH
#LogLevel INFO
# Authentication:
#LoginGraceTime 2m
PermitRootLogin no
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10
#PubkeyAuthentication yes
# Expect .ssh/authorized_keys2 to be disregarded by default in future.
#AuthorizedKeysFile .ssh/authorized_keys .ssh/authorized_keys2
#AuthorizedPrincipalsFile none
#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody
# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes
# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication no
#PermitEmptyPasswords no
# Change to yes to enable challenge-response passwords (beware issues with
# some PAM modules and threads)
KbdInteractiveAuthentication no
# Kerberos options
#KerberosAuthentication no
#KerberosOrLocalPasswd yes
#KerberosTicketCleanup yes
#KerberosGetAFSToken no
# GSSAPI options
#GSSAPIAuthentication no

```

Figure 45: SSH Server Configuration

```

GNU nano 7.2 /etc/iptables/rules.v4
# Generated by iptables-save v1.8.10 (nf_tables) on Fri Jul 11 08:58:45 2025
*filter
:INPUT ACCEPT [169:33802]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -p udp -m udp --dport 1194 -j ACCEPT
-A INPUT -p udp -m udp --dport 1194 -j ACCEPT
-A INPUT -i tun0 -j ACCEPT
-A INPUT -p udp -m udp --dport 1194 -j ACCEPT
-A INPUT -i tun0 -j ACCEPT
-A FORWARD -i tun0 -o ens37 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -i tun0 -j ACCEPT
-A FORWARD -i tun0 -o ens37 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -i ens37 -o tun0 -j ACCEPT
COMMIT
# Completed on Fri Jul 11 08:58:45 2025
# Generated by iptables-save v1.8.10 (nf_tables) on Fri Jul 11 08:58:45 2025
*nat
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [74:5627]
-A POSTROUTING -s 10.8.0.0/24 -o ens37 -j MASQUERADE
COMMIT
# Completed on Fri Jul 11 08:58:45 2025

```

Figure 46: iptables rule set file.

Appendix D: Test Results Table and Screenshots

Table 7: Reliability and Reconnection Test Results.

Test ID	Scenario	Downtime (sec)	Packet Loss	Recovery Type	Notes
1.1	OpenVPN service restart (Server)	6	5	Automatic	Tunnel reconnected promptly
1.2	Full VPN server reboot	33	32	Automatic	Persistent reconnection observed
1.3	Server interface down/up (manual)	16	16	Automatic	Reconnection after interface restart
1.4	VPN client restart	<1	0	Automatic	Instant tunnel re-establishment

```
siteadmin@site-b:~$ ping 10.8.0.1 | ts '%H:%M:%S'
05:41:22 PING 10.8.0.1 (10.8.0.1) 56(84) bytes of data.
05:41:22 64 bytes from 10.8.0.1: icmp_seq=1 ttl=64 time=1.32 ms
05:41:23 64 bytes from 10.8.0.1: icmp_seq=2 ttl=64 time=0.845 ms
05:41:24 64 bytes from 10.8.0.1: icmp_seq=3 ttl=64 time=1.30 ms
05:41:25 64 bytes from 10.8.0.1: icmp_seq=4 ttl=64 time=1.05 ms
05:41:31 64 bytes from 10.8.0.1: icmp_seq=9 ttl=64 time=1.16 ms
05:41:32 64 bytes from 10.8.0.1: icmp_seq=10 ttl=64 time=1.35 ms
^Csiteadmin@site-b:~$
```

Figure 47: VPN client reconnected after service restart.

```
Jul 14 05:41:26 vpnserver systemd[1]: Stopping openvpn-server@server.service - OpenVPN service for server...
Jul 14 05:41:28 vpnserver systemd[1]: openvpn-server@server.service: Deactivated successfully.
Jul 14 05:41:28 vpnserver systemd[1]: Stopped openvpn-server@server.service - OpenVPN service for server.
Jul 14 05:41:28 vpnserver systemd[1]: Starting openvpn-server@server.service - OpenVPN service for server...
Jul 14 05:41:28 vpnserver systemd[1]: Started openvpn-server@server.service - OpenVPN service for server.
```

Figure 48: Openvpn Server restart log.

```
siteadmin@site-b:~$ ping 10.8.0.1 | ts '%H:%M:%S'
05:46:24 PING 10.8.0.1 (10.8.0.1) 56(84) bytes of data.
05:46:24 64 bytes from 10.8.0.1: icmp_seq=1 ttl=64 time=1.06 ms
05:46:25 64 bytes from 10.8.0.1: icmp_seq=2 ttl=64 time=0.979 ms
05:46:26 64 bytes from 10.8.0.1: icmp_seq=3 ttl=64 time=0.877 ms
05:46:27 64 bytes from 10.8.0.1: icmp_seq=4 ttl=64 time=0.983 ms
05:47:00 64 bytes from 10.8.0.1: icmp_seq=36 ttl=64 time=1.18 ms
05:47:01 64 bytes from 10.8.0.1: icmp_seq=37 ttl=64 time=0.865 ms
05:47:02 64 bytes from 10.8.0.1: icmp_seq=38 ttl=64 time=0.962 ms
05:47:03 64 bytes from 10.8.0.1: icmp_seq=39 ttl=64 time=1.11 ms
05:47:04 64 bytes from 10.8.0.1: icmp_seq=40 ttl=64 time=1.01 ms
05:47:05 64 bytes from 10.8.0.1: icmp_seq=41 ttl=64 time=0.995 ms
05:47:06 64 bytes from 10.8.0.1: icmp_seq=42 ttl=64 time=0.846 ms
^C
siteadmin@site-b:~$
```

Figure 49: VPN client reconnected automatically after full server reboot

```

vpnadmin@vpnserver:~$ last reboot
reboot    system boot  6.8.0-63-generic Mon Jul 14 05:46   still running
reboot    system boot  6.8.0-63-generic Mon Jul 14 04:50 - 05:46   (00:56)
reboot    system boot  6.8.0-63-generic Sat Jul 12 06:23 - 06:42   (00:19)
reboot    system boot  6.8.0-63-generic Sat Jul 12 05:01 - 06:42   (01:41)
reboot    system boot  6.8.0-63-generic Fri Jul 11 08:36 - 09:00   (00:23)
reboot    system boot  6.8.0-63-generic Fri Jul 11 08:32 - 09:00   (00:27)
reboot    system boot  6.8.0-63-generic Fri Jul 11 06:55 - 09:00   (02:04)
reboot    system boot  6.8.0-63-generic Fri Jul 11 06:35 - 06:52   (00:17)
reboot    system boot  6.8.0-63-generic Fri Jul 11 06:23 - 06:52   (00:29)
reboot    system boot  6.8.0-63-generic Fri Jul 11 06:21 - 06:22   (00:00)

```

Figure 50: System reboot recorded at 05:46 confirms VPN server downtime during recovery test.

```

vpnadmin@vpnserver:~$ sudo ip link set ens37 down
vpnadmin@vpnserver:~$ sudo ip link set ens37 up
vpnadmin@vpnserver:~$

```

Figure 51: VPN server interface temporarily disabled and restored.

```

siteadmin@site-b:~$ ping 10.8.0.1 | ts '%H:%M:%S'
06:19:23 PING 10.8.0.1 (10.8.0.1) 56(84) bytes of data.
06:19:23 64 bytes from 10.8.0.1: icmp_seq=1 ttl=64 time=1.47 ms
06:19:24 64 bytes from 10.8.0.1: icmp_seq=2 ttl=64 time=0.887 ms
06:19:25 64 bytes from 10.8.0.1: icmp_seq=3 ttl=64 time=1.44 ms
06:19:41 64 bytes from 10.8.0.1: icmp_seq=19 ttl=64 time=1.46 ms
06:19:42 64 bytes from 10.8.0.1: icmp_seq=20 ttl=64 time=0.958 ms
06:19:43 64 bytes from 10.8.0.1: icmp_seq=21 ttl=64 time=1.10 ms
06:19:44 64 bytes from 10.8.0.1: icmp_seq=22 ttl=64 time=1.04 ms
06:19:45 64 bytes from 10.8.0.1: icmp_seq=23 ttl=64 time=1.02 ms
^C
siteadmin@site-b:~$

```

Figure 52: VPN tunnel recovered after server interface disruption.

```

siteadmin@site-b:~$ ping 10.8.0.1 | ts '%H:%M:%S'
06:32:38 PING 10.8.0.1 (10.8.0.1) 56(84) bytes of data.
06:32:38 64 bytes from 10.8.0.1: icmp_seq=1 ttl=64 time=0.677 ms
06:32:39 64 bytes from 10.8.0.1: icmp_seq=2 ttl=64 time=0.649 ms
06:32:40 64 bytes from 10.8.0.1: icmp_seq=3 ttl=64 time=0.758 ms
06:32:41 64 bytes from 10.8.0.1: icmp_seq=4 ttl=64 time=0.738 ms
06:32:43 64 bytes from 10.8.0.1: icmp_seq=6 ttl=64 time=0.908 ms
06:32:44 64 bytes from 10.8.0.1: icmp_seq=7 ttl=64 time=0.872 ms
06:32:45 64 bytes from 10.8.0.1: icmp_seq=8 ttl=64 time=0.820 ms
06:32:45 From 192.168.56.20 icmp_seq=5 Destination Host Unreachable
06:32:46 64 bytes from 10.8.0.1: icmp_seq=9 ttl=64 time=1.01 ms
06:32:47 64 bytes from 10.8.0.1: icmp_seq=10 ttl=64 time=0.957 ms
06:32:48 64 bytes from 10.8.0.1: icmp_seq=11 ttl=64 time=0.995 ms
06:32:49 64 bytes from 10.8.0.1: icmp_seq=12 ttl=64 time=0.917 ms
06:32:50 64 bytes from 10.8.0.1: icmp_seq=13 ttl=64 time=0.870 ms
^C
siteadmin@site-b:~$

```

Figure 53: Ping to VPN server interrupted during client restart and automatically resumed.

```

siteadmin@site-b:~$ date && sudo systemctl restart openvpn@client && date
Mon Jul 14 06:32:42 UTC 2025
Mon Jul 14 06:32:42 UTC 2025
siteadmin@site-b:~$

```

Figure 54: OpenVPN client restarted on Site-B.

Table 8: Performance and Load Testing Results.

Test ID	Test Description	Bandwidth	CPU Usage	Jitter / Loss	Notes
2.1	Sustained iperf3 TCP test	152 Mbit/s	CPU1: 75.5%	N/A	Stable under continuous load

2.2	4 parallel TCP streams	~178 Mbit/s	CPU0: 99.3%	N/A	High CPU load during concurrent streams
2.3	UDP 10 Mbit/s for 60 sec	10 Mbit/s	Low	Jitter: 0.017ms, 0% loss	Tunnel stable for real-time traffic
2.4	500MB file transfer (SCP)	21.9 MB/s	Moderate	N/A	Fast transfer without errors

```

GNU nano 7.2 iperf3_test_300s.txt
[ 5] 261.00-262.00 sec 18.9 MBytes 158 Mbits/sec 1 214 KBytes
[ 5] 262.00-263.00 sec 18.0 MBytes 151 Mbits/sec 2 203 KBytes
[ 5] 263.00-264.00 sec 15.5 MBytes 130 Mbits/sec 0 255 KBytes
[ 5] 264.00-265.00 sec 16.2 MBytes 136 Mbits/sec 0 298 KBytes
[ 5] 265.00-266.00 sec 16.5 MBytes 138 Mbits/sec 28 175 KBytes
[ 5] 266.00-267.00 sec 16.4 MBytes 137 Mbits/sec 0 233 KBytes
[ 5] 267.00-268.00 sec 17.2 MBytes 145 Mbits/sec 3 218 KBytes
[ 5] 268.00-269.00 sec 17.0 MBytes 143 Mbits/sec 0 270 KBytes
[ 5] 269.00-270.00 sec 17.1 MBytes 144 Mbits/sec 0 313 KBytes
[ 5] 270.00-271.00 sec 17.4 MBytes 146 Mbits/sec 5 268 KBytes
[ 5] 271.00-272.00 sec 16.1 MBytes 135 Mbits/sec 25 176 KBytes
[ 5] 272.00-273.00 sec 15.4 MBytes 129 Mbits/sec 0 233 KBytes
[ 5] 273.00-274.00 sec 17.4 MBytes 146 Mbits/sec 0 282 KBytes
[ 5] 274.00-275.00 sec 16.8 MBytes 141 Mbits/sec 0 324 KBytes
[ 5] 275.00-276.00 sec 17.2 MBytes 145 Mbits/sec 18 272 KBytes
[ 5] 276.00-277.00 sec 17.0 MBytes 143 Mbits/sec 9 243 KBytes
[ 5] 277.00-278.00 sec 17.5 MBytes 147 Mbits/sec 0 290 KBytes
[ 5] 278.00-279.00 sec 16.9 MBytes 141 Mbits/sec 26 243 KBytes
[ 5] 279.00-280.00 sec 17.0 MBytes 143 Mbits/sec 3 213 KBytes
[ 5] 280.00-281.00 sec 16.9 MBytes 142 Mbits/sec 7 186 KBytes
[ 5] 281.00-282.00 sec 17.4 MBytes 146 Mbits/sec 0 244 KBytes
[ 5] 282.00-283.00 sec 16.1 MBytes 135 Mbits/sec 3 224 KBytes
[ 5] 283.00-284.00 sec 17.2 MBytes 145 Mbits/sec 1 213 KBytes
[ 5] 284.00-285.00 sec 16.6 MBytes 139 Mbits/sec 4 198 KBytes
[ 5] 285.00-286.00 sec 17.8 MBytes 149 Mbits/sec 0 255 KBytes
[ 5] 286.00-287.00 sec 17.4 MBytes 146 Mbits/sec 15 222 KBytes
[ 5] 287.00-288.00 sec 16.9 MBytes 142 Mbits/sec 14 198 KBytes
[ 5] 288.00-289.00 sec 16.8 MBytes 141 Mbits/sec 32 178 KBytes
[ 5] 289.00-290.00 sec 16.5 MBytes 138 Mbits/sec 0 237 KBytes
[ 5] 290.00-291.00 sec 16.9 MBytes 142 Mbits/sec 3 213 KBytes
[ 5] 291.00-292.00 sec 15.5 MBytes 130 Mbits/sec 48 199 KBytes
[ 5] 292.00-293.00 sec 16.6 MBytes 139 Mbits/sec 0 253 KBytes
[ 5] 293.00-294.00 sec 17.1 MBytes 144 Mbits/sec 1 230 KBytes
[ 5] 294.00-295.00 sec 16.9 MBytes 142 Mbits/sec 1 210 KBytes
[ 5] 295.00-296.00 sec 19.0 MBytes 159 Mbits/sec 0 270 KBytes
[ 5] 296.00-297.00 sec 16.6 MBytes 139 Mbits/sec 0 313 KBytes
[ 5] 297.00-298.00 sec 17.2 MBytes 145 Mbits/sec 23 247 KBytes
[ 5] 298.00-299.00 sec 17.5 MBytes 147 Mbits/sec 43 217 KBytes
[ 5] 299.00-300.00 sec 17.4 MBytes 146 Mbits/sec 6 188 KBytes
- - - - -
[ ID] Interval      Transfer      Bitrate      Retr
[ 5] 0.00-300.00 sec 5.32 GBytes 152 Mbits/sec 2371
[ 5] 0.00-300.01 sec 5.32 GBytes 152 Mbits/sec
      sender
      receiver

iperf Done.
G Help      O Write Out  W Where Is  X Cut       T Execute   C Location  M-U Undo    M-A Set Mark M-] To Bracket M-O Previous
X Exit      R Read File  N Replace   U Paste     J Justify   G Go To Line M-E Redo    M-C Copy    Q Where Was M-K Next

```

Figure 55: iperf3 TCP test.

```

GNU nano 7.2 iperf3_test.t4.txt
[SUM] 54.00-55.00 sec 21.4 MBytes 179 Mbits/sec 14
[ 5] 55.00-56.00 sec 4.75 MBytes 39.8 Mbits/sec 49 73.2 KBytes
[ 7] 55.00-56.00 sec 3.75 MBytes 31.5 Mbits/sec 39 70.5 KBytes
[ 9] 55.00-56.00 sec 8.50 MBytes 71.3 Mbits/sec 20 94.9 KBytes
[11] 55.00-56.00 sec 4.00 MBytes 33.6 Mbits/sec 101 66.4 KBytes
[SUM] 55.00-56.00 sec 21.0 MBytes 176 Mbits/sec 209
[ 5] 56.00-57.00 sec 4.88 MBytes 40.9 Mbits/sec 7 89.5 KBytes
[ 7] 56.00-57.00 sec 5.12 MBytes 43.0 Mbits/sec 6 86.8 KBytes
[ 9] 56.00-57.00 sec 6.62 MBytes 55.6 Mbits/sec 27 100 KBytes
[11] 56.00-57.00 sec 4.25 MBytes 35.7 Mbits/sec 3 92.2 KBytes
[SUM] 56.00-57.00 sec 20.9 MBytes 175 Mbits/sec 43
[ 5] 57.00-58.00 sec 4.75 MBytes 39.8 Mbits/sec 43 94.9 KBytes
[ 7] 57.00-58.00 sec 5.12 MBytes 43.0 Mbits/sec 13 93.5 KBytes
[ 9] 57.00-58.00 sec 5.38 MBytes 45.1 Mbits/sec 3 108 KBytes
[11] 57.00-58.00 sec 5.12 MBytes 43.0 Mbits/sec 34 97.6 KBytes
[SUM] 57.00-58.00 sec 20.4 MBytes 171 Mbits/sec 93
[ 5] 58.00-59.00 sec 4.75 MBytes 39.9 Mbits/sec 19 104 KBytes
[ 7] 58.00-59.00 sec 4.75 MBytes 39.9 Mbits/sec 14 103 KBytes
[ 9] 58.00-59.00 sec 6.62 MBytes 55.6 Mbits/sec 8 110 KBytes
[11] 58.00-59.00 sec 4.75 MBytes 39.9 Mbits/sec 10 106 KBytes
[SUM] 58.00-59.00 sec 20.9 MBytes 175 Mbits/sec 51
[ 5] 59.00-60.00 sec 6.30 MBytes 53.4 Mbits/sec 15 84.0 KBytes
[ 7] 59.00-60.00 sec 3.38 MBytes 28.3 Mbits/sec 35 54.2 KBytes
[ 9] 59.00-60.00 sec 6.00 MBytes 50.3 Mbits/sec 21 89.5 KBytes
[11] 59.00-60.00 sec 5.50 MBytes 46.1 Mbits/sec 51 85.4 KBytes
[SUM] 59.00-60.00 sec 21.2 MBytes 178 Mbits/sec 122
[ID] Interval Transfer Bitrate Retr
[ 5] 0.00-60.00 sec 341 MBytes 47.7 Mbits/sec 1399 sender
[ 5] 0.00-60.01 sec 339 MBytes 47.4 Mbits/sec receiver
[ 7] 0.00-60.00 sec 303 MBytes 42.3 Mbits/sec 1150 sender
[ 7] 0.00-60.01 sec 302 MBytes 42.2 Mbits/sec receiver
[ 9] 0.00-60.00 sec 323 MBytes 45.2 Mbits/sec 1442 sender
[ 9] 0.00-60.01 sec 322 MBytes 45.0 Mbits/sec receiver
[11] 0.00-60.00 sec 311 MBytes 43.4 Mbits/sec 1274 sender
[11] 0.00-60.01 sec 310 MBytes 43.3 Mbits/sec receiver
[SUM] 0.00-60.00 sec 1.25 GBytes 179 Mbits/sec 5265 sender
[SUM] 0.00-60.01 sec 1.24 GBytes 178 Mbits/sec receiver

iperf Done.
Help Write Out Where Is Cut Execute Location M-U Undo M-A Set Mark M-] To Bracket M-O Previous
Exit Read File Replace Paste Justify Go To Line M-R Redo M-B Copy M-^ Where Was M-X Next

```

Figure 56: iperf test on 4 parallel TCP streams.

```

GNU nano 7.2 udp_test.10M.txt
[ 5] 21.00-22.00 sec 1.19 MBytes 10.0 Mbits/sec 901
[ 5] 22.00-23.00 sec 1.19 MBytes 9.99 Mbits/sec 900
[ 5] 23.00-24.00 sec 1.19 MBytes 10.0 Mbits/sec 901
[ 5] 24.00-25.00 sec 1.19 MBytes 9.99 Mbits/sec 900
[ 5] 25.00-26.00 sec 1.19 MBytes 10.0 Mbits/sec 901
[ 5] 26.00-27.00 sec 1.19 MBytes 9.99 Mbits/sec 900
[ 5] 27.00-28.00 sec 1.19 MBytes 10.0 Mbits/sec 901
[ 5] 28.00-29.00 sec 1.19 MBytes 9.99 Mbits/sec 900
[ 5] 29.00-30.00 sec 1.19 MBytes 10.0 Mbits/sec 901
[ 5] 30.00-31.00 sec 1.19 MBytes 10.0 Mbits/sec 901
[ 5] 31.00-32.00 sec 1.19 MBytes 9.99 Mbits/sec 900
[ 5] 32.00-33.00 sec 1.19 MBytes 10.0 Mbits/sec 901
[ 5] 33.00-34.00 sec 1.19 MBytes 9.99 Mbits/sec 900
[ 5] 34.00-35.00 sec 1.19 MBytes 10.0 Mbits/sec 901
[ 5] 35.00-36.00 sec 1.19 MBytes 9.99 Mbits/sec 900
[ 5] 36.00-37.00 sec 1.19 MBytes 10.0 Mbits/sec 901
[ 5] 37.00-38.00 sec 1.19 MBytes 10.0 Mbits/sec 901
[ 5] 38.00-39.00 sec 1.19 MBytes 9.99 Mbits/sec 900
[ 5] 39.00-40.00 sec 1.19 MBytes 10.0 Mbits/sec 901
[ 5] 40.00-41.00 sec 1.19 MBytes 9.99 Mbits/sec 900
[ 5] 41.00-42.00 sec 1.19 MBytes 10.0 Mbits/sec 901
[ 5] 42.00-43.00 sec 1.19 MBytes 9.99 Mbits/sec 900
[ 5] 43.00-44.00 sec 1.19 MBytes 10.0 Mbits/sec 901
[ 5] 44.00-45.00 sec 1.19 MBytes 10.0 Mbits/sec 901
[ 5] 45.00-46.00 sec 1.19 MBytes 10.0 Mbits/sec 901
[ 5] 46.00-47.00 sec 1.19 MBytes 9.99 Mbits/sec 900
[ 5] 47.00-48.00 sec 1.19 MBytes 10.0 Mbits/sec 901
[ 5] 48.00-49.00 sec 1.19 MBytes 9.99 Mbits/sec 900
[ 5] 49.00-50.00 sec 1.19 MBytes 10.0 Mbits/sec 901
[ 5] 50.00-51.00 sec 1.19 MBytes 9.99 Mbits/sec 900
[ 5] 51.00-52.00 sec 1.19 MBytes 10.0 Mbits/sec 901
[ 5] 52.00-53.00 sec 1.19 MBytes 10.0 Mbits/sec 901
[ 5] 53.00-54.00 sec 1.19 MBytes 9.99 Mbits/sec 900
[ 5] 54.00-55.00 sec 1.19 MBytes 10.0 Mbits/sec 901
[ 5] 55.00-56.00 sec 1.19 MBytes 10.0 Mbits/sec 900
[ 5] 56.00-57.00 sec 1.19 MBytes 10.0 Mbits/sec 901
[ 5] 57.00-58.00 sec 1.19 MBytes 9.99 Mbits/sec 900
[ 5] 58.00-59.00 sec 1.19 MBytes 10.0 Mbits/sec 901
[ 5] 59.00-60.00 sec 1.19 MBytes 9.99 Mbits/sec 900
[ID] Interval Transfer Bitrate Jitter Lost/Total Datagrams
[ 5] 0.00-60.00 sec 71.5 MBytes 10.0 Mbits/sec 0.000 ms 0/54034 (0%) sender
[ 5] 0.00-60.00 sec 71.5 MBytes 10.0 Mbits/sec 0.017 ms 0/54034 (0%) receiver

iperf Done.
Help Write Out Where Is Cut Execute Location M-U Undo M-A Set Mark M-] To Bracket M-O Previous
Exit Read File Replace Paste Justify Go To Line M-R Redo M-B Copy M-^ Where Was M-X Next

```

Figure 57: iperf test for UDP 10 Mbit/s for 60 sec.

```

siteadmin@site-b:~$ time scp vpnadmin@10.8.0.1:/home/vpnadmin/testfile.img /tmp/
vpnadmin@10.8.0.1's password:
testfile.img
100% 500MB 21.9MB/s 00:22

real    0m24.246s
user    0m3.212s
sys     0m8.183s
siteadmin@site-b:~$

```

Figure 58: 500 MB file transferred securely over VPN tunnel using SCP.

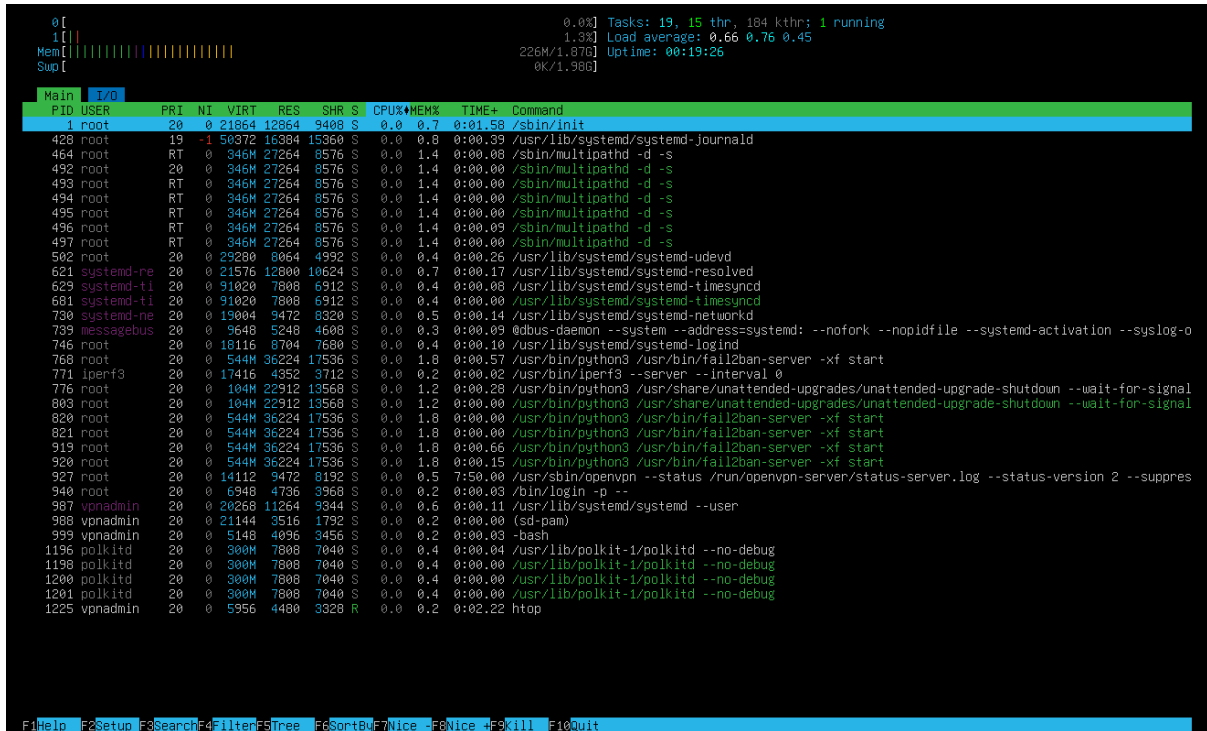


Figure 59: System resource usages on rest.

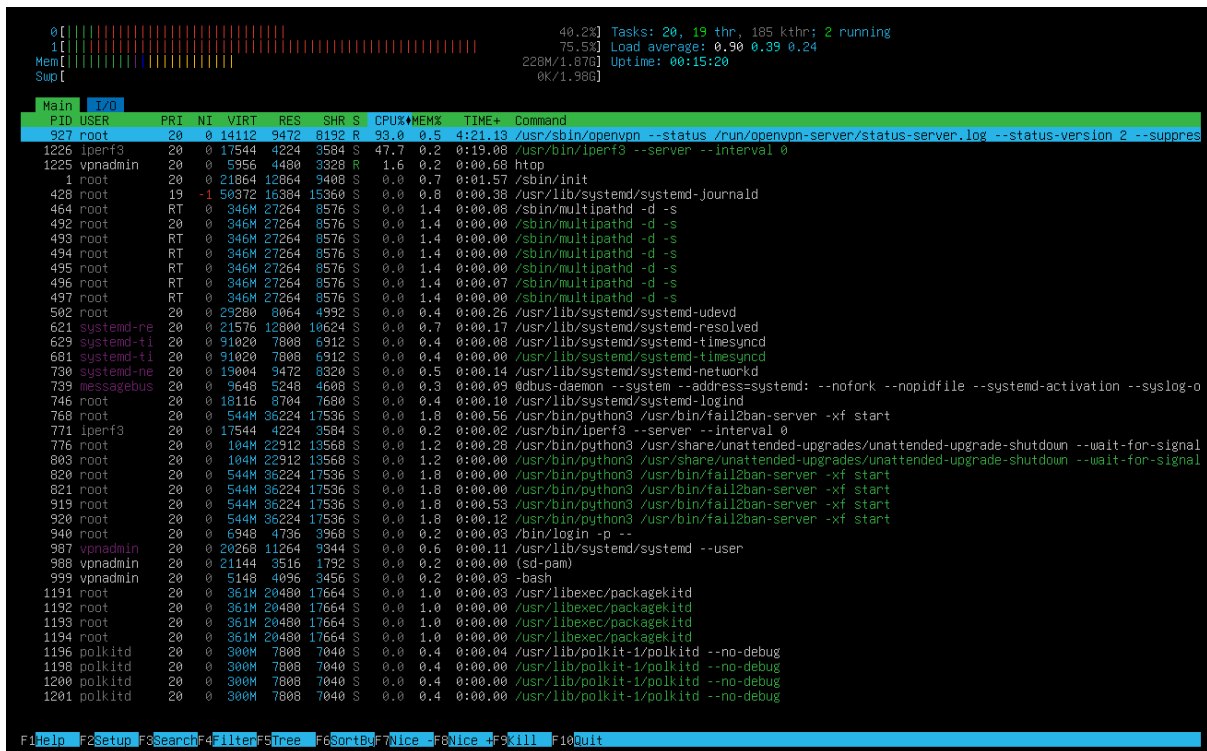


Figure 60: System resource usages during the test iperf3 -c 10.8.0.1 -t 300.

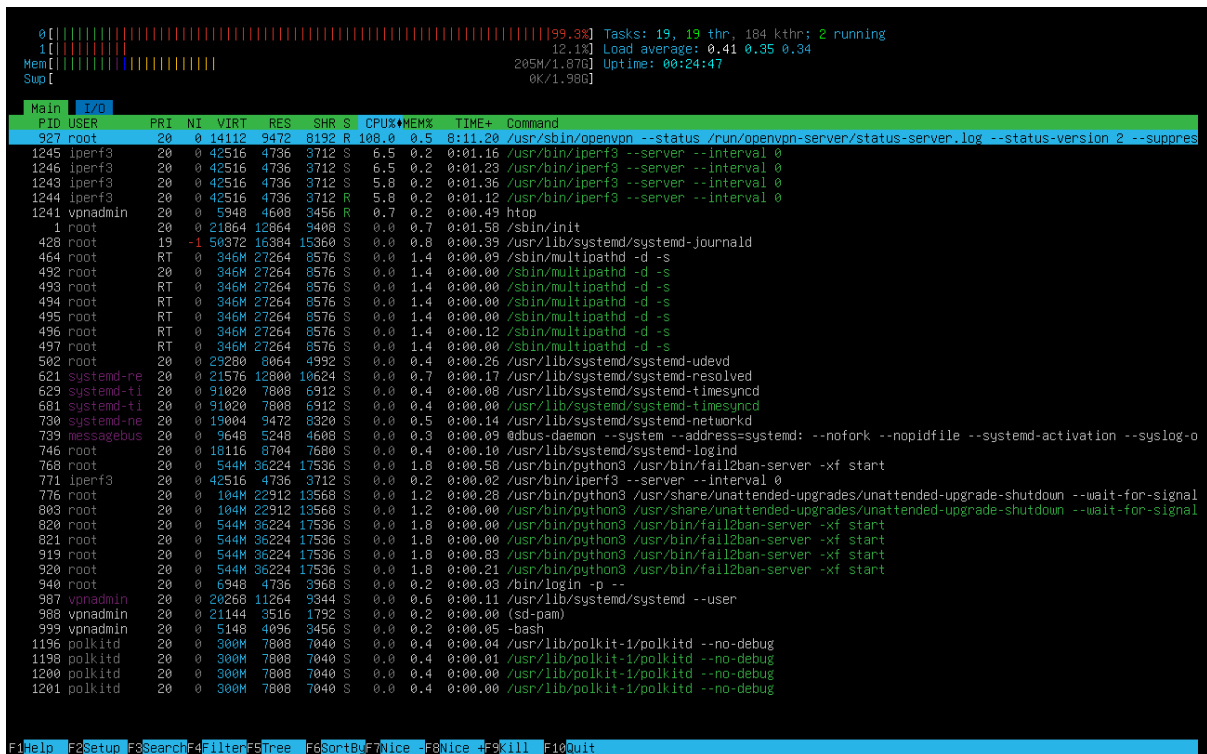


Figure 61: System resource usages during the test iperf3 -c 10.8.0.1 -t 60 -P 4.

```

0[|||||] 11.0% Tasks: 19, 16 thr, 185 kthr: 1 running
1[|||||] 8.0% Load average: 0.25 0.35 0.35
Mem[|||||] 201M/1.87G Uptime: 00:27:41
Swap[|] 0K/1.98G

Main T/O
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
927 root 20 0 14112 9472 6192 S 0.1 0.5 0:45.31 /usr/sbin/openvpon --status /run/openvpon-server/status-server.log --status-version 2 --suppres
771 iperf3 20 0 42008 4264 3712 S 7.4 0.2 0:00.33 /usr/bin/iperf3 -s -server -interval 0
1248 iperf3 20 0 42008 4264 3712 S 1.4 0.2 0:00.58 /usr/bin/iperf3 -s -server -interval 0
1 root 20 0 21864 12864 9408 S 0.0 0.7 0:01.58 /sbin/init
428 root 19 -1 50372 16384 15360 S 0.0 0.8 0:00.39 /usr/lib/systemd/systemd-journald
454 root RT 0 346M 27264 8576 S 0.0 1.4 0:00.10 /sbin/multipathd -d -s
492 root 20 0 346M 27264 8576 S 0.0 1.4 0:00.00 /sbin/multipathd -d -s
493 root RT 0 346M 27264 8576 S 0.0 1.4 0:00.00 /sbin/multipathd -d -s
494 root RT 0 346M 27264 8576 S 0.0 1.4 0:00.00 /sbin/multipathd -d -s
495 root RT 0 346M 27264 8576 S 0.0 1.4 0:00.00 /sbin/multipathd -d -s
496 root RT 0 346M 27264 8576 S 0.0 1.4 0:00.13 /sbin/multipathd -d -s
497 root RT 0 346M 27264 8576 S 0.0 1.4 0:00.00 /sbin/multipathd -d -s
502 root 20 0 29260 8064 4992 S 0.0 0.4 0:00.26 /usr/lib/systemd/systemd-udevd
621 systemd-ne 20 0 21576 12800 10624 S 0.0 0.7 0:00.17 /usr/lib/systemd/systemd-resolved
629 systemd-tl 20 0 91020 7808 6912 S 0.0 0.4 0:00.00 /usr/lib/systemd/systemd-timesyncd
681 systemd-tl 20 0 91020 7808 6912 S 0.0 0.4 0:00.00 /usr/lib/systemd/systemd-timesyncd
730 systemd-ne 20 0 19004 9472 8320 S 0.0 0.5 0:00.14 /usr/lib/systemd/systemd-networkd
739 messagebus 20 0 9548 5248 4608 S 0.0 0.3 0:00.09 dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation --syslog-o
746 root 20 0 18116 8704 7680 S 0.0 0.4 0:00.10 /usr/lib/systemd/systemd-logind
768 root 20 0 544M 36224 17536 S 0.0 1.8 0:00.59 /usr/bin/python3 /usr/bin/fail2ban-server -xf start
776 root 20 0 104M 22912 13568 S 0.0 1.2 0:00.28 /usr/bin/python3 /usr/share/unattended-upgrades/unattended-upgrade-shutdown --wait-for-signal
803 root 20 0 104M 22912 13568 S 0.0 1.2 0:00.00 /usr/bin/python3 /usr/share/unattended-upgrades/unattended-upgrade-shutdown --wait-for-signal
820 root 20 0 544M 36224 17536 S 0.0 1.8 0:00.00 /usr/bin/python3 /usr/bin/fail2ban-server -xf start
821 root 20 0 544M 36224 17536 S 0.0 1.8 0:00.00 /usr/bin/python3 /usr/bin/fail2ban-server -xf start
919 root 20 0 544M 36224 17536 S 0.0 1.8 0:00.53 /usr/bin/python3 /usr/bin/fail2ban-server -xf start
920 root 20 0 544M 36224 17536 S 0.0 1.8 0:00.23 /usr/bin/python3 /usr/bin/fail2ban-server -xf start
940 root 20 0 6948 4736 3968 S 0.0 0.2 0:00.03 /bin/login -p --
987 vpnadmin 20 0 20268 11264 9344 S 0.0 0.6 0:00.11 /usr/lib/systemd/systemd --user
988 vpnadmin 20 0 21144 3516 1792 S 0.0 0.2 0:00.00 (sd-pam)
999 vpnadmin 20 0 5148 4096 3456 S 0.0 0.2 0:00.05 -bash
1196 polkitd 20 0 300M 7808 7040 S 0.0 0.4 0:00.04 /usr/lib/polkit-1/polkitd --no-debug
1198 polkitd 20 0 300M 7808 7040 S 0.0 0.4 0:00.01 /usr/lib/polkit-1/polkitd --no-debug
1200 polkitd 20 0 300M 7808 7040 S 0.0 0.4 0:00.00 /usr/lib/polkit-1/polkitd --no-debug
1201 polkitd 20 0 300M 7808 7040 S 0.0 0.4 0:00.00 /usr/lib/polkit-1/polkitd --no-debug
1241 vpnadmin 20 0 5948 4608 3456 R 0.0 0.2 0:01.38 htop
F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice F8Nice F9Kill F10Quit

```

Figure 62: System resource usages during the test iperf3 -c 10.8.0.1 -u -b 10M -t 60.