



Design, Simulation and Deployment of a Snort-Based Intrusion Detection and Prevention System (IDPS) for Securing the ICT Infrastructure of Kebbi State Polytechnic, Dakingari Using Software Defined Networking (SDN)

Shamsu Sani¹, Salihu Alhassan², Sulaiman Umar S. Noma³, Sadiq Abubakar Zagga⁴

Department of Computer Science, Kebbi State Polytechnic, Dakingari, Kebbi State, Nigeria ^{1,2,3,4}

ABSTRACT

The increasing dependence of tertiary institutions on Information and Communication Technology (ICT) infrastructure has expanded the attack surface of institutional networks and increased the potential impact of network-based cyberattacks. Software Defined Networking (SDN) provides programmability, centralized control and a global view of network resources, but the separation of the control and data planes also introduces security risks, particularly against denial-of-service and distributed denial-of-service (DDoS) attacks. This study presents the design, simulation and prototype deployment of a Snort-based Intrusion Detection and Prevention System (IDPS) integrated with an SDN environment for securing the ICT infrastructure of Kebbi State Polytechnic, Dakingari. The work transforms an earlier Snort-based SDN intrusion detection study into an institution-oriented security architecture. An experimental and prototype-based research design was adopted. A virtual testbed was implemented using VirtualBox, Ubuntu Linux, OpenDaylight, Mininet, Open vSwitch, Snort, Kali Linux, Metasploitable2 and Wireshark. A programmable SDN topology was created and subjected to normal traffic and selected DDoS attack scenarios, including SYN flood, ping flood, ping of death and HTTP/SYN flooding. Snort was configured with local rules and traffic mirroring to inspect network traffic and generate alerts. The experimental observations show that the integrated environment successfully exposed abnormal traffic patterns and generated Snort alerts for the selected attack scenarios; in the documented ping flood and ping-of-death experiments, the attacked destination recorded zero packets received after the attack was stopped, corresponding to complete packet loss during those controlled observations. The study demonstrates the feasibility of combining SDN programmability with Snort-based network intrusion monitoring as a cost-effective security model for tertiary-institution networks. However, the reported experiments constitute a controlled virtual/prototype deployment rather than a live production deployment in the Polytechnic ICT Centre. Future work should therefore validate the architecture on live institutional traffic, quantify detection accuracy and false-positive rates, and extend the system with automated SDN-based flow blocking and adaptive or machine-learning-assisted detection.

Keywords: Software Defined Networking, Snort, Intrusion Detection and Prevention System, DDoS, Network Security, Mininet, OpenDaylight, Kebbi State Polytechnic.

1. Introduction

The rapid integration of ICT into tertiary education has changed how institutions deliver teaching, learning, research and administration. Academic and administrative services increasingly depend on interconnected systems, institutional networks, online platforms, digital communication and electronic records. This dependence improves operational efficiency but also increases the consequences of network disruption or unauthorized access. The ICT infrastructure of a tertiary institution therefore requires security controls capable of continuously monitoring traffic, identifying malicious activity and supporting timely mitigation.

The original MSc study on which this paper is based established a Snort-based intrusion detection environment for SDN and demonstrated selected DDoS attacks in a virtual testbed. The present paper retains the technical foundation of that work while reframing it around the security needs of Kebbi State Polytechnic, Dakingari. The institution-oriented framing is consistent with the accompanying research proposal, which identifies institutional ICT services as important assets requiring real-time monitoring, attack detection and mitigation.

Traditional security mechanisms such as firewalls and standalone signature-based tools remain important, but no single mechanism provides complete protection. Network attacks may exploit weaknesses in hosts, applications, switches, controllers or communication paths. In SDN, the centralized controller provides a global view and enables programmable responses, but it may also become an attractive target because compromise or overload of the control plane can affect a large portion of the network.

SDN separates the control plane from the data plane and enables network behaviour to be programmed through standardized interfaces. This separation creates an opportunity to integrate security functions with network control. A network intrusion detection system can inspect traffic and identify known attack patterns, while SDN can provide a programmable mechanism for enforcing mitigation policies. Snort is particularly suitable for a prototype of this type because it provides packet inspection, rule-based detection, logging and alert generation.

The central problem addressed in this paper is therefore the need for a practical and cost-conscious security architecture that can monitor an SDN-based institutional network and identify common DDoS traffic patterns. The study focuses on four attack scenarios documented in the MSc experiment: SYN flood, ping flood, ping of death and HTTP/SYN flooding.

The study aims to design, simulate and prototype-deploy a Snort-based IDPS integrated with SDN for improving the security posture of the ICT infrastructure of Kebbi State Polytechnic, Dakingari. The specific objectives are to: (i) design an SDN security architecture suitable for the institutional ICT environment; (ii) configure Snort for network traffic inspection and customized DDoS signatures; (iii) construct a controlled virtual testbed using OpenDaylight, Mininet and Open vSwitch; (iv) generate and observe selected DDoS attack scenarios; (v) evaluate the ability of the integrated environment to identify malicious traffic; and (vi) identify the requirements for moving from laboratory prototype deployment to live institutional deployment.

The contribution of the study is primarily practical. Rather than proposing only a theoretical detection model, it demonstrates an implementable open-source architecture in which SDN provides programmable network control and Snort provides packet-level security monitoring. The resulting architecture can serve as a baseline for future deployment and quantitative evaluation within the Polytechnic and comparable tertiary institutions.

2. Related Literature

Snort IDS rules improved using the association rules data mining technique for detecting network probe attacks, as was investigated in Improving Intrusion Detection System Based on Snort Rules for Network Probe Attacks Detection. where data on anomalous behavior is found using the Snort IDS. and demonstrated that the proposed Snort IDS rules—based on data mining detection of network, probe attacks—proved more effective than the original Snort IDS rules, including icmp rules and icmp-info. rules of Snort IDS and the appropriate parameters for the proposed Snort IDS rules are defined as Min sup set to 10% and Min conf set to 100%, and through the application of eight variable attributes and create a warning when unusual behavior patterns match up with the set of rules (Niemiec et al., 2021).

The study of effectiveness and performance of a new intrusion detection system (IDS) known as a signature-based multi-layer distributed intrusion detection system employing mobile agents, which was utilized to detect imminent threats with an extremely high success rate. Small, efficient, and many databases were used in a dynamic manner with plenty of opportunity for regular updates. And the results show that the suggested IDS model outperforms conventional systems with a single database for string signatures by a wide margin (Ajiya Ahmad et al., 2021).

Anomaly Detection Technique for Intrusion Detection in SDN Environment using Continuous Data Stream Machine Learning Algorithms were studied in order to overcome the main challenges that happen when developing an anomaly based system using machine learning algorithms. And discovered that, the bandwidth depletion attack targets the channel between the switches and the controller through either UDP or HTTP flooding, and also the effectiveness of the proposed technique (Ribamar et al., 2021).

A Survey on Intrusion Detection System for Software Defined Networks (SDN) was introduced and a survey of existing security solutions was designed to secure the SDN. And discussed a comparative study of various IDS approaches based on a deep learning model and machine learning methods. The researchers have started to apply DL techniques in the area of IDS as deep learning has the capability to cope with large scale data and to automatically find a correlation in the data. But more expansion of deep learning is required in the field of IDS for SDN to detect all new types of attacks with good accuracy (Hande & Muddana, 2020)

2.1 Intrusion Detection and Prevention

An intrusion detection system monitors events or network traffic and analyzes them for indicators of unauthorized or malicious activity. IDS technologies are commonly classified according to where monitoring occurs and how detection is performed. Host-based IDS monitors activities on individual hosts, whereas network-based IDS examines traffic traversing a network. Detection may also be signature-based, in which known attack patterns are matched against observed traffic, or anomaly-based, in which deviations from an established normal profile are identified.

Snort is primarily associated with signature-based network intrusion detection and is capable of protocol analysis, packet inspection, rule matching, logging and alert generation. Its flexible rule language makes it possible to define institution-specific signatures for attack patterns. This flexibility is useful in a controlled research environment because the researcher can create and validate rules for the exact traffic scenarios being tested.

An IDPS extends the IDS concept by combining detection with a response or prevention function. Detection alone informs an administrator that malicious traffic is present; prevention requires an enforcement mechanism capable of dropping, blocking, redirecting or otherwise constraining the malicious traffic. In an SDN environment, the controller and programmable switches provide a potential enforcement layer. The original MSc experiment primarily demonstrated detection and traffic disruption in the controlled testbed; full autonomous prevention through dynamic SDN flow installation should be treated as a next implementation stage unless it is separately validated.

2.2 Software Defined Networking

SDN is a networking paradigm in which the control logic is separated from packet-forwarding functions. The architecture is commonly described in terms of an application layer, control layer and infrastructure layer. The controller occupies the control layer and maintains a logically centralized view of the network, while switches in the infrastructure layer forward packets according to rules supplied by the controller.

The programmability of SDN is particularly relevant to cybersecurity. Security applications can use network-wide information to observe traffic and implement policies across multiple switches. However, centralization can introduce controller-related risks, including denial of service, unauthorized access and exploitation of controller interfaces. Consequently, an SDN deployment should treat the controller as a critical security asset and should combine network monitoring with appropriate access control, resilience and traffic-management mechanisms.

2.3 DDoS Attacks in SDN

DDoS attacks attempt to exhaust network, computing or application resources through large volumes of malicious traffic or repeated service requests. In SDN environments, DDoS traffic can affect hosts, switches, links and the controller itself. Flooding attacks are particularly important because their traffic volume can consume bandwidth and processing resources and can also increase control-plane workload.

The present study concentrates on four controlled scenarios. SYN flooding generates large numbers of TCP SYN packets and can exhaust connection-handling resources. Ping flooding generates high-rate ICMP traffic. Ping of death uses oversized or malformed ICMP packets to stress or exploit packet-processing behaviour. HTTP flooding targets an application service through repeated HTTP requests. These scenarios provide complementary examples of transport-, network- and application-oriented flooding.

2.4 Snort and SDN-Based Security

Previous research reviewed in the MSc thesis demonstrates the use of Snort and SDN for DDoS detection, while other studies investigate machine-learning-based intrusion detection in SDN. The literature indicates that signature-based systems remain useful for known attacks because rules can provide transparent and relatively lightweight detection, whereas anomaly and machine-learning approaches offer opportunities for detecting previously unseen patterns. A practical institutional architecture can therefore begin with a transparent Snort-based baseline and subsequently incorporate adaptive analytics.

The research gap addressed here is not the claim that Snort or SDN-based DDoS detection is entirely new. Rather, the study provides a context-specific prototype for a tertiary-institution ICT environment and documents the configuration and controlled attack experiments needed as a foundation for subsequent live deployment, automated SDN enforcement and quantitative benchmarking.

3. Materials and Methods

3.1 Research Design

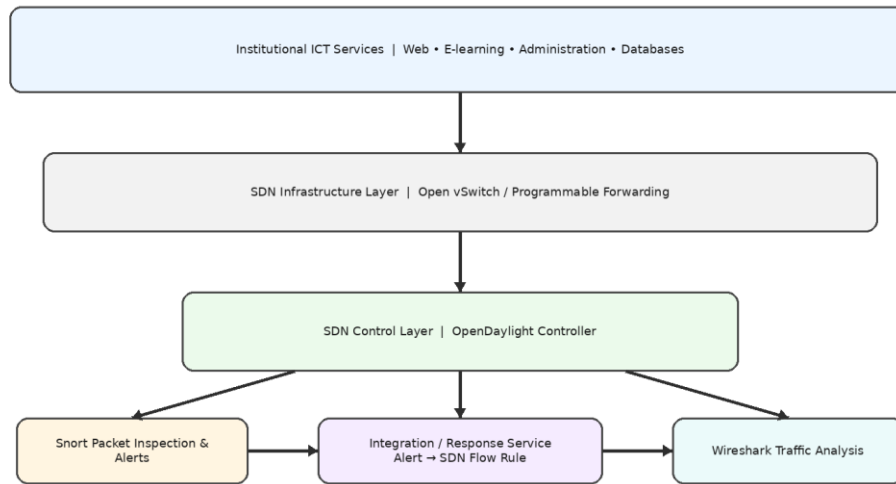
The study adopted an experimental, simulation-based and prototype-oriented research design. A deductive approach was used to translate security requirements and findings from prior studies into a testable architecture, implement the architecture in a controlled environment, generate selected attack traffic and observe the response of the security components. The use of a virtual testbed reduces the risk of disrupting production services while allowing repeatable experimentation.

The research design follows the prototype approach used in the MSc work. Open-source technologies were selected to reduce cost and facilitate reproducibility. The documented host computer used an Intel Core i5-5200U processor at 2.20 GHz with 16 GB RAM. Virtual machines were operated using Oracle VirtualBox.

3.2 Proposed Institutional Security Architecture

Figure 1. Proposed SDN-Snort Institutional Security Architecture.

Figure 1. Proposed SDN-Snort Institutional Security Architecture



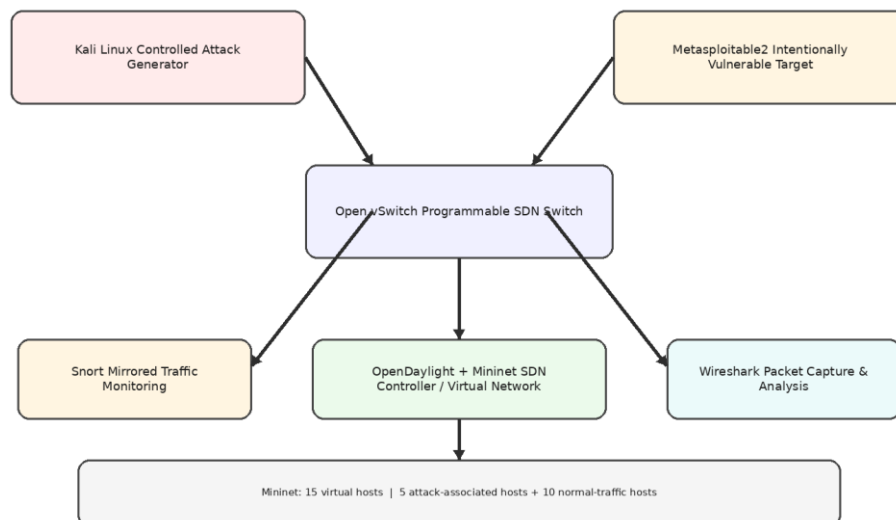
The proposed architecture places the Polytechnic ICT network behind a programmable SDN forwarding layer. OpenDaylight acts as the SDN controller, while Mininet emulates hosts and Open vSwitch provides programmable switching. Snort is positioned as a network security monitoring component connected to the switching environment through a mirrored traffic path. Kali Linux is used only as a controlled security-testing platform for generating attack traffic, while Metasploitable2 provides an intentionally vulnerable target for laboratory experiments.

The architecture has four logical functions: (1) traffic forwarding and programmable control through Open vSwitch and OpenDaylight; (2) packet-level inspection through Snort; (3) traffic observation and protocol analysis through Wireshark; and (4) controlled attack generation through Kali Linux. In a production implementation, Snort alerts would be passed to a response component capable of translating alerts into SDN flow rules that block or rate-limit malicious sources. The original experiment validates the monitoring and alerting path; the automated response path remains an important deployment extension.

3.3 Virtual Testbed Configuration

Figure 2. Virtual Testbed Topology Used for Controlled Experiments.

Figure 2. Virtual Testbed Topology Used for Controlled Experiments



The virtual environment comprised two Ubuntu Linux environments together with Kali Linux and Metasploitable2. OpenDaylight and Mininet were installed and configured in the SDN environment, while Snort was installed on a separate Ubuntu virtual machine for network monitoring. Mininet was used to construct the virtual network, and Open vSwitch was used as the programmable switch. Kali Linux generated attack traffic and Wireshark captured and analyzed traffic patterns.

The MSc topology included fifteen virtual hosts within Mininet: five hosts associated with malicious traffic generation and ten hosts used for normal traffic. The hosts were connected through an Open vSwitch switch under the control of OpenDaylight. Kali Linux and the Snort monitoring system were connected to the switching environment through additional interfaces. Traffic mirroring was configured so that traffic traversing the monitored path could be inspected by Snort.

The topology was deliberately isolated from the production network. This is important because DDoS generation must never be performed against an operational institutional network without explicit authorization and safeguards.

3.4 Attack Scenarios

Four attack categories were selected from the MSc experiment: ping flood, ping of death, SYN flood and HTTP/SYN flooding. Hping3 was used as the principal packet-generation utility in the documented experiments. Normal ICMP communication was first established to provide a baseline before attack traffic was introduced.

The experimental process consisted of: establishing normal connectivity; starting the SDN controller and Mininet topology; confirming host reachability; configuring Snort local rules; validating the Snort configuration; enabling traffic mirroring; generating an attack from the controlled Kali environment; observing traffic in Wireshark; and recording the resulting Snort alerts and destination behaviour.

3.5 Snort Rule Configuration

Snort was configured as a network-based IDS using local rules targeted at the selected DDoS patterns. The rules were tested for configuration validity before the attack experiments. The monitoring interface was connected to the mirrored traffic path so that Snort could inspect packets not necessarily destined for the Snort host itself.

For a production deployment, the local rule set should be supplemented by maintained community or vendor rules, tuned thresholds, suppression policies, baseline traffic profiles and institution-specific signatures. Rules should also be reviewed periodically to reduce false positives and accommodate legitimate high-volume services such as online registration, examination periods and e-learning traffic.

3.6 Performance Evaluation Framework

The original thesis primarily reports qualitative detection observations and traffic graphs rather than a complete numerical confusion-matrix evaluation. Accordingly, this paper does not invent accuracy, precision, recall or false-positive values that were not reported in the source experiment. For a publication-grade live or extended test, the recommended metrics are detection rate/recall, precision, false-positive rate, false-negative rate, F1-score, alert latency, throughput, packet loss, CPU utilization and memory utilization.

Where the source experiment provides an observable outcome, it is retained. In the ping flood and ping-of-death demonstrations, the destination was reported as receiving zero packets after the attack was stopped, with 100% packet loss in those controlled observations. These results should be interpreted as attack-session observations rather than as a general network performance percentage.

4. Implementation and Results

4.1 SDN Controller and Network Initialization

OpenDaylight was started on the Ubuntu SDN environment and its management interface was used to observe the controller environment. Mininet was subsequently launched with remote-controller configuration and used to create simple and custom topologies. Connectivity tests between hosts confirmed that the virtual hosts could communicate before the attack scenarios were introduced.

The custom Mininet topology was implemented using Python. The prototype code created hosts, an Open vSwitch switch and links between them, then started the Mininet command-line interface. This approach allowed the topology to be modified programmatically and provided a repeatable basis for the experiments.

4.2 Normal Traffic Baseline

Normal ICMP communication was generated before the attacks. Wireshark was used to observe the resulting traffic and establish a visual baseline. The baseline was important because it demonstrated ordinary packet exchange and provided a reference against which the sudden increase in traffic during flooding experiments could be observed.

4.3 SYN Flood Observation

The SYN flood scenario generated a high volume of TCP SYN traffic toward the target environment. Wireshark displayed the resulting increase in packet activity, distinguishing the attack period from normal communication. Snort was configured to inspect the mirrored traffic and generate an alert when the traffic pattern matched the relevant rule.

The experiment demonstrates the value of combining packet-level visibility with programmable network control. Snort identifies the traffic pattern, while the SDN architecture provides an appropriate location for a subsequent automated response such as installing a temporary drop rule or rate-limiting the offending flow.

4.4 Ping Flood Observation

The ping flood scenario was generated from the controlled Kali Linux environment toward the target. The thesis documents that Snort detected the ping flood and that the destination subsequently recorded zero packets received after the attack was stopped. This was accompanied by complete packet loss in the controlled observation. The result confirms that the test environment could expose the flooding condition and that the attack materially affected packet delivery to the target during the experiment.

4.5 Ping of Death Observation

The ping-of-death experiment used oversized ICMP traffic in the controlled testbed. Wireshark was used to observe the traffic, while Snort generated an alert indicating possible ping-of-death activity. The source thesis reports zero packets received and 100% packet loss at the destination after the attack was stopped. The observation supports the feasibility of detecting the selected abnormal ICMP pattern in the laboratory environment.

4.6 Traffic Analysis

Note: Figures in this manuscript are schematic representations prepared from the documented experimental architecture and should be interpreted as conceptual/testbed diagrams rather than photographs of a live institutional deployment.

The traffic graphs documented in the MSc work show a relatively low-volume normal ICMP baseline and a pronounced increase in captured packets during flooding. For the SYN flood, the graph shows a sudden increase in packets over time. Similar traffic increases were documented for ping-of-death and ping flood. These observations support the use of traffic-rate information as a complementary signal alongside signature matching.

The results should nevertheless be interpreted carefully. The study did not report a sufficiently large labeled dataset or repeated trials from which statistically robust accuracy and false-positive estimates could be calculated. The results therefore establish proof of concept rather than a statistically exhaustive performance benchmark.

4.7 Summary of Experimental Findings

Scenario	Traffic type	Observation	Evidence in source study
Normal traffic	ICMP	Stable baseline communication observed in Wireshark	Normal ping/ICMP figures
SYN flood	TCP	Sudden increase in packet activity; Snort detection demonstrated	Wireshark and Snort experiment
Ping flood	ICMP	Snort detection demonstrated; destination later recorded zero packets	Controlled attack experiment
Ping of Death	Oversized ICMP	Snort indicated possible detection; destination later recorded zero packets	Controlled attack experiment
HTTP/SYN flooding	Application/transport flooding	Attack traffic and Snort detection path demonstrated	Controlled experiment

5. Discussion

The experimental results indicate that a Snort-based network monitoring layer can be integrated with an SDN testbed to provide visibility into selected DDoS traffic patterns. The most important architectural advantage is the possibility of combining detection with programmable response. In a conventional network, an alert may require an administrator to locate the appropriate device and manually change a configuration. In SDN, the controller can potentially receive an alert and apply a flow policy across the relevant forwarding devices.

The study also illustrates the value of a virtual testbed for an institution such as Kebbi State Polytechnic. A live DDoS experiment against an operational ICT network would present unacceptable availability and safety risks. VirtualBox, Mininet and intentionally vulnerable hosts allow the security architecture to be developed and tested without affecting students, staff or institutional services.

At the same time, Snort's signature-based approach has a clear limitation: it is strongest when the attack pattern is known and an appropriate rule exists. Variants of known attacks, previously unseen attacks and highly adaptive adversaries may evade static signatures. This limitation motivates a layered architecture in which Snort provides transparent signature detection while statistical anomaly detection or machine learning is added as a second analytical layer.

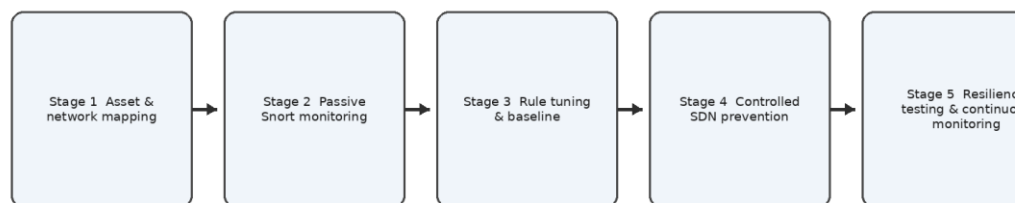
Another important issue is that the title of this paper uses the term 'deployment'. The underlying MSc work demonstrates deployment of the prototype components in a virtualized testbed, not a verified live production deployment inside the Polytechnic ICT Centre. For academic accuracy, this paper uses 'prototype deployment' when interpreting the existing results. A journal version should not claim that the solution has been operationally deployed on the Polytechnic's live network unless such a deployment is subsequently performed and documented.

The proposed institutional architecture should also address operational requirements that were outside the scope of the MSc experiment. These include controller hardening, authentication, secure management interfaces, redundancy, logging and retention, time synchronization, network segmentation, rule lifecycle management, incident-response procedures, privacy protection, backup and recovery, and controlled change management. These are necessary before any production deployment.

6. Proposed Deployment Model for Kebbi State Polytechnic ICT Infrastructure

Figure 3. Staged Deployment Model for Polytechnic ICT Infrastructure.

Figure 3. Staged Deployment Model for Polytechnic ICT Infrastructure



Operational principle: begin in monitor-only mode; enable blocking only after baseline data, tuning, authorization and rollback procedures are established.

Based on the experimental architecture, the following staged deployment model is recommended for the Polytechnic. Stage 1 is asset and network mapping, identifying servers, user segments, wireless networks, administrative systems, e-learning services and internet gateways. Stage 2 is passive monitoring, in which Snort observes mirrored traffic without blocking it. Stage 3 is rule tuning, using legitimate institutional traffic to reduce false alarms. Stage 4 is controlled prevention, in which selected high-confidence alerts are translated into temporary SDN flow rules for blocking or rate limiting. Stage 5 is resilience testing and continuous monitoring.

In the proposed operational architecture, OpenDaylight would maintain the SDN control view; Open vSwitch or compatible switches would enforce flow rules; Snort would inspect mirrored traffic; and an integration service would translate validated Snort alerts into controller actions. High-confidence DDoS alerts could trigger temporary source blocking or rate limiting, with automatic expiration of rules to avoid permanent disruption caused by false positives.

A production deployment should begin in monitor-only mode. Blocking should only be enabled after adequate baseline data, rule tuning, authorization and rollback procedures are established. The security system should also maintain audit logs showing the alert, source, destination, rule identifier, action taken, time and duration of mitigation.

7. Conclusion

This paper presented a Snort-based Intrusion Detection and Prevention System architecture for securing the ICT infrastructure of Kebbi State Polytechnic, Dakingari through Software Defined Networking. The work was derived from an MSc virtual SDN security experiment and extended by explicitly framing the architecture around institutional ICT protection and a staged path toward deployment.

The experimental environment combined OpenDaylight, Mininet, Open vSwitch, Snort, Wireshark, Kali Linux, Metasploitable2 and VirtualBox. The documented experiments demonstrated the generation and observation of normal traffic, SYN flood, ping flood, ping-of-death and HTTP/SYN flooding scenarios. Snort local rules and traffic mirroring enabled detection of the selected attack patterns. The reported ping flood and ping-of-death sessions produced zero packets received at the destination after the attack was stopped, corresponding to 100% packet loss in those specific controlled observations.

The main conclusion is that the combined SDN-Snort architecture is technically feasible as a laboratory prototype and provides a useful foundation for institutional network security. However, the current evidence does not establish live-network deployment, comprehensive detection accuracy, low false-positive rates or autonomous prevention performance. Those claims require further experiments and controlled operational validation.

8. Limitations and Future Research

The principal limitation is the controlled virtual environment. It does not reproduce the full diversity, scale, encrypted traffic, user behaviour and workload variability of a live tertiary-institution network. The study also concentrates on selected DDoS scenarios and does not provide a comprehensive evaluation of other threats such as lateral movement, credential attacks, malware, ARP spoofing, DNS abuse or application-layer exploitation.

A second limitation is the absence of a complete numerical confusion-matrix evaluation. Future experiments should use repeated trials with labeled benign and malicious traffic and report precision, recall, F1-score, false-positive rate, false-negative rate, detection latency and throughput. Measurements should be taken at low, medium and high traffic intensity to determine how the system behaves under increasing load.

A third limitation is that the current work does not fully automate the prevention loop. Future work should implement a secure Snort-to-SDN integration layer capable of installing and expiring flow rules automatically. Additional research should investigate anomaly detection and machine-learning models, explainable security analytics, encrypted-traffic monitoring, controller redundancy and adversarial robustness.

Finally, before live deployment at Kebbi State Polytechnic, a pilot should be conducted on a segregated network segment under institutional authorization. The pilot should begin with passive monitoring, followed by controlled prevention after baseline behaviour and false-positive rates have been established.

9. Declarations

Ethical and operational safety: All attack traffic described in this study was generated within a controlled virtual laboratory environment. The methodology should not be reproduced against production systems without explicit authorization.

Data availability: The study is based on the experimental configuration, observations and materials contained in the author's MSc research. A reproducible dataset should be prepared in future work by exporting timestamped packet/alert records from the testbed.

Conflict of interest: No conflict of interest is declared in relation to the technical work presented in this manuscript.

References

- Abubakar, A., & Pranggono, B. (2017). Machine learning based intrusion detection system for Software Defined Networks.
- Ahmad, S., & Hussain Mir, A. (2022). SDN interfaces: Protocols, taxonomy and challenges. *International Journal of Wireless and Microwave Technologies*, 12(2), 11–32. <https://doi.org/10.5815/ijwmt.2022.02.02>
- Ajaeiy, G. A., & Chehab, N. A. I. H. E. A. K. A. (2017). Flow-based intrusion detection system for SDN. *Journal of Basic and Applied Scientific Research*, 3(6), 850–859.
- Ajiya Ahmad, A., Boukari, S., Musa Bello, A., & Aliyu Muhammad, M. (2021). A survey of intrusion detection techniques on Software Defined Networking (SDN). *International Journal of Innovative Science and Research Technology*, 6(8).
- Alashhab, A. A., Zahid, M. S. M., Azim, M. A., Dahi, M. Y., Isyaku, B., & Ali, S. (2022). A survey of low rate DDoS detection techniques based on machine learning in Software-Defined Networks. *Symmetry*, 14(8). <https://doi.org/10.3390/sym14081563>
- Alkhalil, Z., Hewage, C., Nawaf, L., & Khan, I. (2021). Phishing attacks: A recent comprehensive study and a new anatomy. *Frontiers in Computer Science*, 3. <https://doi.org/10.3389/fcomp.2021.563060>
- Badotra, S., & Panda, S. N. (2020). SNORT based early DDoS detection system using OpenDaylight and Open Networking Operating System in software defined networking. *Cluster Computing*. <https://doi.org/10.1007/s10586-020-03133-y>

- Bhosale, K. S., Nenova, M., & Iliev, G. (2020). Intrusion detection in communication networks using different classifiers. In *Techno-Societal 2018*, 19–28. https://doi.org/10.1007/978-3-030-16962-6_3
- Casula, M., Rangarajan, N., & Shields, P. (2021). The potential of working hypotheses for deductive exploratory research. *Quality & Quantity*, 55(5), 1703–1725. <https://doi.org/10.1007/s11135-020-01072-9>
- Dantas Silva, F. S., Silva, E., Neto, E. P., Lemos, M., Venancio Neto, A. J., & Esposito, F. (2020). A taxonomy of DDoS attack mitigation approaches featured by SDN technologies in IoT scenarios. *Sensors*, 20(11). <https://doi.org/10.3390/s20113078>
- Dong, S., Abbas, K., & Jain, R. (2019). A survey on distributed denial of service (DDoS) attacks in SDN and cloud computing environments. *IEEE Access*, 7, 80813–80828. <https://doi.org/10.1109/ACCESS.2019.2922196>
- Farooq, M. S., Riaz, S., & Alvi, A. (2023). Security and privacy issues in Software-Defined Networking (SDN): A systematic literature review. *Electronics*, 12(14). <https://doi.org/10.3390/electronics12143077>
- Hande, Y., & Muddana, A. (2020). A survey on intrusion detection system for software defined networks (SDN). *International Journal of Business Data Communications and Networking*, 16(1), 28–47. <https://doi.org/10.4018/IJBDCN.2020010103>
- Iriananda, S. W., et al. (2019). Intelligent distributed privacy-preserving DDoS attack detection using software defined networks.
- Jonker, J., & Pennink, B. W. (2010). The essence of research methodology.
- Kaur Chahal, J., Bhandari, A., & Behal, S. (2019). Distributed denial of service attacks: A threat or challenge. *New Review of Information Networking*, 24(1), 31–103. <https://doi.org/10.1080/13614576.2019.1611468>
- Mustafa, A. S., Mkpnam, D., & Abdullahi, A. (2018). Security in Software Defined Networks (SDN): Challenges and research opportunities for Nigeria. *International Journal of Computer Applications Technology and Research*, 7(8), 297–300. <https://doi.org/10.7753/IJCATR0708.1002>
- Olanrewaju, R. F., Islam Khan, B. U., Najeeb, A. R., Ku Zahir, K. N. A., & Hussain, S. (2018). Snort-based smart and swift intrusion detection system. *Indian Journal of Science and Technology*, 11(4), 1–9. <https://doi.org/10.17485/ijst/2018/v11i4/120917>
- Pradhan, A., & Mathew, R. (2020). Solutions to vulnerabilities and threats in Software Defined Networking (SDN). *Procedia Computer Science*, 171, 2581–2589. <https://doi.org/10.1016/j.procs.2020.04.280>
- Ribamar, A. de, Ribeiro, L., Yves, R., Santos, C., Clayton, A., & Nascimento, A. (2021). Anomaly detection technique for intrusion detection in SDN environment using continuous data stream machine learning algorithms.
- Riskhan, B., Safuan, H. A. J., Hussain, K., Elnour, A. A. H., Abdelmaboud, A., Khan, F., & Kundi, M. (2023). An adaptive distributed denial of service attack prevention technique in a distributed environment. *Sensors*, 23(14). <https://doi.org/10.3390/s23146574>
- Singh. (2020). UC Berkeley electronic theses and dissertations.
- Sudar, K. M., & Deepalakshmi, P. (2020). Comparative study on IDS using machine learning approaches for software defined networks. *International Journal of Intelligent Enterprise*, 7(1–3), 15–27. <https://doi.org/10.1504/IJIE.2020.104642>
- Sultana, N., Chilamkurti, N., Peng, W., & Alhadad, R. (2018). Survey on SDN based network intrusion detection system using machine learning approaches.
- Yin, D., Zhang, L., & Yang, K. (2018). A DDoS attack detection and mitigation with Software-Defined Internet of Things framework. *IEEE Access*, 6, 24694–24705. <https://doi.org/10.1109/ACCESS.2018.2831284>