ISSN 2723-6471

Enhancing Network Security and Content Control in Academic Environments Using Pi-hole

R. Elavarasi[1], P. Gajalakshmi[2,*], D. Kadhiravan[3], M. Naresh[4], Harprith Kaur R. S.[5]

[1]*AMET Deemed to be University, Chennai, India*

[2,3,4]*University College of Engineering, Tindivanam, India*

[5]*Faculty of Data Science and IT, INTI International University, 71800 Nilai, Malaysia*

(Received: March 3, 2026; Revised: May 10, 2026; Accepted: July 30, 2026; Available online: September 14, 2026)

Abstract

Academic institutions require reliable internet access to support teaching, research, communication, and digital learning activities. However, unrestricted connectivity may expose institutional networks to non-academic content, malicious domains, excessive bandwidth consumption, and unauthorized access attempts. This study develops and evaluates a lightweight DNS-based internet filtering architecture using Pi-hole for centralized content control in academic environments. The system was deployed on a Raspberry Pi 3B+ as the primary DNS resolver, enabling domain requests to be processed through configurable blocklists, whitelists, scheduled policies, and centralized query logging. Its performance was compared with firewall-based filtering, proxy-server filtering, and browser-extension filtering using five evaluation dimensions: blocking effectiveness, CPU efficiency, cost-effectiveness, scalability, and bypass resistance. The results show that Pi-hole achieved the highest overall score of 9.0, with scores of 9 for blocking effectiveness, CPU efficiency, and bypass resistance, 10 for cost-effectiveness, and 8 for scalability. The enterprise firewall provided greater client capacity but required substantially higher infrastructure and operational costs. Proxy servers offered application-level control but imposed higher computational overhead, while browser extensions were inexpensive but limited in scalability and resistance to circumvention. The findings demonstrate that Pi-hole provides a balanced, centralized, and resource-efficient filtering solution for small- and medium-sized academic networks. Its effectiveness can be further strengthened through forced DNS redirection, firewall integration, role-based policies, and automated domain classification.

Keywords: DNS Filtering, Pi-Hole, Academic Network, Content Control, Network Security, Raspberry Pi, Blocking Effectiveness, Cost-Effectiveness, Process Innovation

1. Introduction

Internet connectivity has become an essential component of contemporary academic environments, supporting access to digital libraries, learning management systems, research databases, collaborative platforms, and cloud-based educational services. The expansion of online higher education has further increased institutional dependence on reliable and accessible internet infrastructure [1]. However, unrestricted network access also introduces operational and security concerns, including exposure to inappropriate content, excessive use of non-academic services, malware distribution, phishing attacks, unauthorized access, and inefficient bandwidth consumption. Consequently, academic institutions require network-management mechanisms that can maintain accessibility to legitimate educational resources while limiting activities that may reduce productivity or compromise institutional information assets [2].

Conventional network-control mechanisms commonly rely on firewalls, proxy servers, endpoint applications, and manually maintained access-control policies. Firewalls are effective for enforcing rules based on addresses, ports, and protocols, but domain-level and application-level inspection may require additional configuration or computationally intensive packet-analysis mechanisms [2]. Proxy-based filtering provides more granular control over web requests, although its deployment may introduce processing overhead, latency, certificate-management complexity, and scalability limitations. These approaches also require continuous policy maintenance because users may attempt to circumvent institutional restrictions through virtual private networks, alternative proxies, encrypted DNS, or

***Corresponding author: P. Gajalakshmi (gajalakshmip23@gmail.com)**
DOI: https://doi.org/10.47738/jads.v7i3.1472

independently configured resolvers. Therefore, filtering effectiveness should not be evaluated only by the number of blocked requests, but also by computational efficiency, deployment cost, scalability, and resistance to bypass attempts.

DNS-based filtering provides an alternative mechanism by controlling domain resolution before a client establishes a connection with a requested service. Pi-hole implements this approach as a network-level DNS sinkhole that compares incoming queries with predefined blocklists and prevents the resolution of restricted, advertising, tracking, or malicious domains [3]. Collaborative and automatically updated domain-blocking approaches can improve the identification of suspicious domains while reducing the administrative burden associated with manually maintained lists [4]. Because filtering is applied centrally, Pi-hole can protect multiple client devices without requiring browser extensions or separate software installations on each endpoint. Its lightweight architecture also allows deployment on low-power hardware, local servers, or virtual machines, making it potentially suitable for educational institutions with limited infrastructure budgets.

The effectiveness of a DNS-filtering deployment nevertheless depends on the design of its filtering policies and the institutional context in which it is implemented. Academic networks serve heterogeneous users whose access requirements may differ according to role, schedule, laboratory activity, or research responsibility. A uniform blocking policy may restrict legitimate resources, whereas an overly permissive configuration may fail to reduce distractions and security exposure. Administrators must therefore support configurable blocklists, whitelists, user-group policies, logging mechanisms, and periodic evaluation of false-positive and false-negative decisions. These controls should also be aligned with broader information-security practices in higher education, including access governance, monitoring, authentication, and protection of user experience [5], [6].

Existing studies have examined internet filtering, access restriction, adaptive monitoring, and intelligent network-management mechanisms from different perspectives [7], [8]. Lightweight adaptive monitoring has demonstrated that resource consumption can be reduced by dynamically adjusting monitoring intensity while maintaining acceptable analytical performance [9]. Recurrent learning approaches have also been employed to model changing network conditions and support service-quality prediction in dynamic environments [10]. In addition, neural-network-based adaptive control has shown how computational systems can respond to nonlinear and time-varying conditions without relying entirely on static control assumptions [11]. Although these studies demonstrate the value of adaptive and computationally efficient mechanisms, limited attention has been given to a structured evaluation of lightweight DNS filtering in academic networks across multiple operational dimensions.

This study develops and evaluates a Pi-hole-based content-control architecture for academic environments. The proposed system intercepts institutional DNS requests, applies customizable domain-level filtering, forwards permitted queries to an upstream resolver, and records query activity for administrative analysis. Its performance is assessed against firewall-based filtering, proxy servers, and browser extensions using five criteria: blocking effectiveness, CPU efficiency, cost-effectiveness, scalability, and bypass resistance. By combining a practical deployment architecture with a multidimensional evaluation framework, the study aims to determine whether Pi-hole can provide a balanced solution for centralized, affordable, and operationally efficient internet filtering in educational institutions.

2. Literature Review

Internet filtering in academic environments has traditionally been implemented through firewall-based controls, proxy servers, endpoint applications, and institutional access policies. Firewalls remain a fundamental network-security mechanism because they can regulate traffic based on IP addresses, ports, protocols, and predefined access-control rules [15]. However, conventional firewall configurations are primarily designed for packet-level enforcement rather than fine-grained domain classification. More advanced implementations employ Deep Packet Inspection to analyses packet payloads and application behavior, but this approach increases computational overhead, configuration complexity, and privacy concerns [17]. Although firewall policies can provide strong institutional control, their effectiveness depends on accurate rule definition, continuous traffic analysis, and regular policy updates [22]. These limitations have encouraged the adoption of complementary mechanisms that operate at different layers of the network.

Proxy-based filtering provides application-layer visibility by acting as an intermediary between client devices and external web services. Proxy servers can enforce user authentication, maintain browsing logs, apply URL-based

policies, and inspect web requests before forwarding them to their destinations [19]. Nevertheless, the increasing use of HTTPS, virtual private networks, and encrypted communication protocols has reduced the effectiveness of conventional proxy inspection. HTTPS decryption may improve visibility but requires certificate management and additional processing resources. Secure DNS architecture has therefore become increasingly important because DNS traffic can be manipulated, redirected, or exploited during cyberattacks [18]. Research on DNS-based attack detection has demonstrated that monitoring resolution requests can support the early identification and prevention of access to malicious domains [3], [16].

DNS-based filtering addresses several limitations of packet- and application-level filtering by evaluating domain requests before external connections are established. Pi-hole implements this mechanism as a DNS sinkhole that compares requested domains with configurable blocklists and returns a blocking response when a match is detected. Similar domain-blocking studies have explored collaborative and automated approaches for identifying malicious or undesirable domains [4]. Pi-hole integration has also been investigated as a privacy-enforcement mechanism in smart environments, where centralized DNS control can reduce exposure to advertising, tracking, and telemetry domains across multiple connected devices [23]. Its deployment on Raspberry Pi hardware provides a lightweight alternative to conventional filtering appliances, while virtualized Raspberry Pi environments have demonstrated the broader suitability of low-cost computing platforms for cybersecurity applications [20].

Recent network-management research has increasingly emphasized adaptability, portability, and resource efficiency. Neural-network-defined modulators have been proposed to improve the portability of communication functions across heterogeneous Internet of Things gateways while maintaining low latency and memory consumption [12]. Adaptive monitoring techniques have similarly shown that network-observation intensity can be dynamically adjusted to reduce energy usage and unnecessary data processing [9]. These developments align with the broader transition toward integrated and resource-aware smart environments [14]. However, existing studies have largely examined filtering accuracy, adaptive monitoring, DNS security, or hardware efficiency separately. A limited number of studies provide a unified assessment of DNS-based filtering for academic environments using operational criteria such as blocking effectiveness, CPU efficiency, deployment cost, scalability, and bypass resistance. This study addresses that gap by systematically evaluating Pi-hole against firewall, proxy, and browser-based filtering mechanisms [6], [21].

3. Methodology

3.1. Research Design and System Deployment

This study employed an experimental comparative design to evaluate a Pi-hole-based DNS filtering system against three commonly used filtering approaches: firewall-based filtering, proxy-server filtering, and browser-extension filtering. The evaluation focused on five operational dimensions: blocking effectiveness, CPU efficiency, cost-effectiveness, scalability, and bypass resistance. Pi-hole was selected as the proposed system because it operates as a centralized DNS sinkhole that evaluates domain-resolution requests before connections are established [3]. This architecture enables network-wide filtering without requiring separate software installation on each client device.

The proposed system was deployed using a Raspberry Pi 3B+ connected to the institutional local-area network, as illustrated in figure 1. The Raspberry Pi was assigned a static IP address and configured as the primary DNS resolver for all participating client devices. The network router redirected DNS requests to the Pi-hole server, while permitted requests were forwarded to Google Public DNS as the upstream resolver. Access to the administrative dashboard was protected using authenticated credentials. The Pi-hole configuration included default security and advertisement blocklists, manually specified restricted domains, approved-domain whitelists, and time-based filtering rules.

 ISSN 2723-6471

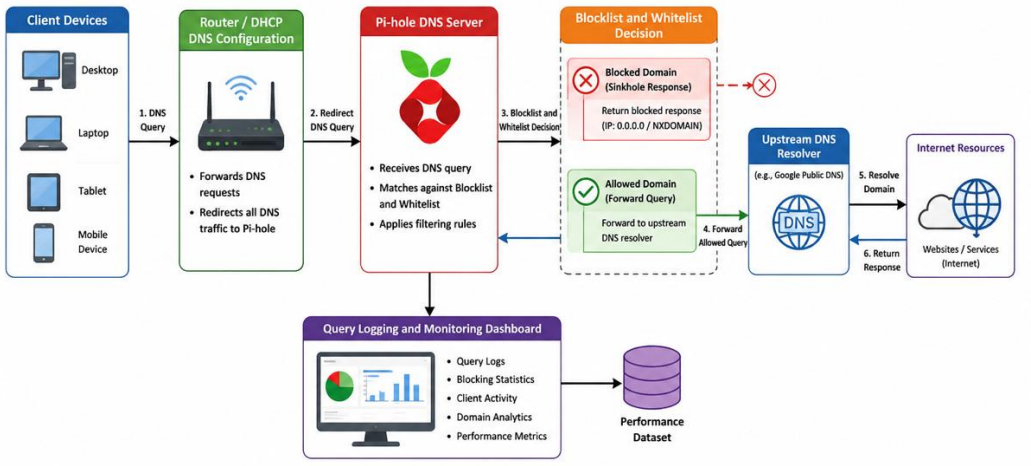

Figure 1. Proposed DNS-Based Filtering and Evaluation Architecture

3.2. Experimental Procedure and Comparative Evaluation

The experiment evaluated the behaviour of the four filtering mechanisms under the same functional requirements. Each solution was assessed based on its ability to restrict predefined non-educational, advertising, tracking, and potentially malicious domains while preserving access to legitimate academic resources. The test configuration included multiple client devices representing typical institutional endpoints, such as desktop computers, laptops, tablets, and mobile devices.

The experimental procedure consisted of four phases. First, the filtering policies were prepared by defining a common list of allowed and restricted domains. Second, each filtering solution was configured using equivalent access-control requirements. The main experimental configuration, including the deployment device, network role, DNS resolver, client devices, filtering policies, comparison methods, monitoring source, and evaluation dimensions, is summarized in table 1. Third, DNS and web requests were generated from client devices to determine whether each system allowed, blocked, or failed to process the requested domain. Finally, operational measurements were recorded and normalized into comparable scores. For Pi-hole, the following data were collected from the administrative dashboard and query logs: total DNS queries; number of blocked queries; number of permitted queries; unique client devices; unique queried domains; CPU utilization; queries processed per second; filtering response; bypass attempts; and infrastructure, maintenance, and power costs.

Filtering decisions were categorized using a confusion-matrix structure. A restricted domain successfully blocked was recorded as a true positive, while an allowed academic domain successfully resolved was recorded as a true negative. A legitimate domain incorrectly blocked was classified as a false positive, whereas a restricted domain that remained accessible was classified as a false negative. This procedure provides a more reproducible assessment than relying only on manually assigned effectiveness values.

Table 1. Experimental Configuration

Configuration Component	Experimental Setting
Proposed filtering system	Pi-hole
Deployment device	Raspberry Pi 3B+
Network role	Primary DNS resolver
IP configuration	Static local IP address
Upstream DNS resolver	Google Public DNS
Client devices	Desktop, laptop, tablet, and mobile device
Filtering policies	Blocklist, whitelist, group-based rules, and scheduled rules
Comparison methods	Firewall, proxy server, and browser extension
Primary monitoring source	Pi-hole dashboard and query log
Evaluation dimensions	Blocking, CPU efficiency, cost, scalability, and bypass resistance

The existing dashboard, resolver-selection, and query-log screenshots may be retained as implementation evidence. However, they should be repositioned in the experimental setup or results section rather than presented as separate methodological explanations. The previous comparison tables can also be retained, but the values should be reported as experimental observations or normalized scores rather than unsupported general estimates.

3.3. Performance Metrics and Scoring Framework

The filtering performance was measured using request-level classification metrics and normalized operational scores. Blocking accuracy was computed as:

$$BA = \frac{TP+TN}{TP+TN+FP+FN} \tag{1}$$

TP denotes restricted domains correctly blocked, TN represents legitimate domains correctly allowed, FP indicates legitimate domains incorrectly blocked, and FN represents restricted domains incorrectly allowed. Because blocking accuracy alone does not capture encrypted-DNS handling and resistance to circumvention, a composite Blocking Effectiveness Score was calculated as:

$$BES = 10\left(w_1 BA + w_2 \frac{ED}{10} + w_3 \frac{BR}{10}\right) \tag{2}$$

BA is expressed as a value between 0 and 1, ED is the encrypted-DNS handling score, BR is the bypass-resistance score, and the weights were defined as:

$$w_1 = 0.5, \qquad w_2 = 0.3, \qquad w_3 = 0.2 \tag{3}$$

with:

$$w_1 + w_2 + w_3 = 1 \tag{4}$$

CPU efficiency was calculated by relating the number of processed requests to average processor utilization:

$$CE = 10 \times \frac{RPS/CPU}{\max(RPS/CPU)} \tag{5}$$

RPS denotes requests processed per second and CPU represents average CPU utilization in percentage. Normalization against the maximum observed ratio ensures that all solutions receive scores within a comparable range from 0 to 10. Cost-effectiveness was calculated using blocking performance and processing capacity relative to the total annualized deployment cost:

$$CSE = 10 \times \frac{\frac{BA \times RPS}{TC+MC+PC}}{\max\left(\frac{BA \times RPS}{TC+MC+PC}\right)} \tag{6}$$

TC denotes the initial technology or hardware cost, MC denotes the annual maintenance cost, and PC denotes the annual power cost. The cost and processing inputs used in the evaluation are presented in table 2.

Table 2. Cost and Processing Inputs

Method	Requests per Second	Hardware Cost (INR)	Maintenance Cost (INR/year)	Power Cost (INR/year)
Pi-hole	7,5	1,5	0	150
Firewall	5	100	10	4
Proxy server	3	80	8	2,4
Browser extension	3	0	0	800

Scalability was evaluated based on the maximum number of supported clients before a measurable degradation in response time or processing stability occurred. The normalized scalability score was calculated as:

$$SS = 10 \times \frac{\log_{10}(C) - \log_{10}(C_{min})}{\log_{10}(C_{max}) - \log_{10}(C_{min})} \tag{7}$$

C is the maximum number of clients supported by a filtering solution, while C_{min} and C_{max} represent the minimum and maximum client capacities observed across all evaluated methods. Bypass resistance was measured by conducting controlled circumvention attempts involving manual DNS modification, encrypted DNS, alternative proxy access, VPN tunnelling, browser replacement, and extension deactivation. The score was calculated as:

$$BRS = 10\left(1 - \frac{SB}{TA}\right) \tag{8}$$

SB represents the number of successful bypass attempts and TA denotes the total number of bypass attempts. A higher score indicates stronger resistance to circumvention. Forced DNS redirection and firewall-assisted DNS enforcement were considered supplementary controls because DNS-based filtering alone may be bypassed when users are permitted to select external or encrypted DNS resolvers [17]. The final overall performance score was obtained using the average of the five normalized dimensions:

$$OPS = \frac{BES + CE + CSE + SS + BRS}{5} \tag{9}$$

The values in table 3 can be transferred from the previous Tables 1–7 and combined into a single evaluation matrix. The previous comparison bar chart may also be reused after updating its title to "Normalized Performance Comparison of Network Filtering Methods" and ensuring that the plotted values correspond exactly to Table 3. Low-cost Raspberry Pi deployment remains relevant to the proposed architecture because such devices provide sufficient computational capacity for lightweight network and cybersecurity applications [20].

Table 3. Final Comparative Evaluation Matrix

Filtering Method	Blocking Effectiveness	CPU Efficiency	Cost-Effectiveness	Scalability	Bypass Resistance	Overall Score
Pi-hole	9	9	10	8	9	9.0
Firewall	7	5	3	9	8	6.4
Proxy server	8	4	4	5	5	5.2
Browser extension	6	8	9	4	3	6.0

4. Results and Discussion

4.1. Pi-hole Deployment and Filtering Output

The Pi-hole system was successfully deployed on a Raspberry Pi 3B+ and configured as the primary DNS resolver for the client network. All DNS requests generated by connected devices were redirected through the Pi-hole server before being forwarded to Google Public DNS. The administrative dashboard recorded the total number of queries, blocked requests, permitted requests, active clients, and queried domains throughout the observation period, as shown in figure 2.

The query log demonstrated that Pi-hole processed each request according to the configured blocklist and whitelist. Requests directed to restricted advertising, tracking, malicious, and non-academic domains were blocked at the DNS resolution stage, while permitted educational and general-purpose domains were forwarded to the upstream resolver. Each transaction recorded the query time, client IP address, requested domain, query type, and filtering status, as illustrated in figure 4. The DNS resolver configuration used in the deployment is presented in figure 3.

The deployment results confirmed that centralized DNS filtering could be applied to all connected clients without requiring separate filtering software on individual devices. Time-based policies were also implemented to differentiate access permissions during and outside academic hours. Educational domains remained accessible throughout the day, while selected social media and entertainment domains could be restricted between 09:00 and 16:00.

ISSN 2723-6471

Figure 2. Pi-hole Monitoring Dashboard

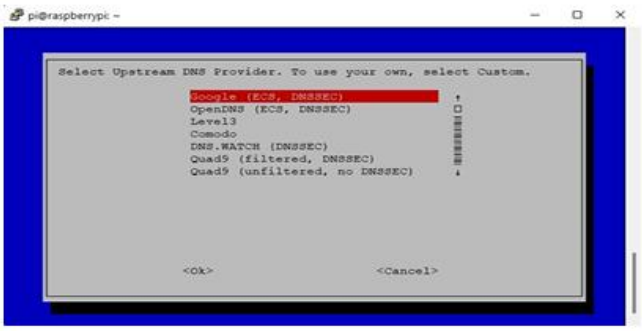

Figure 3. DNS Resolver Configuration

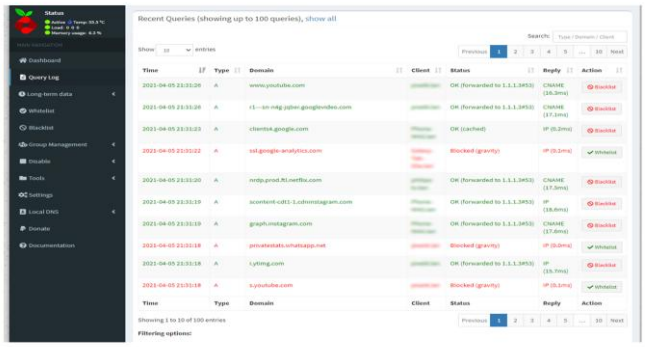

Figure 4. Pi-hole Query Log and Filtering Decisions

4.2.Comparative Performance Results

The four filtering methods were evaluated using blocking effectiveness, CPU efficiency, cost-effectiveness, scalability, and bypass resistance. The normalized results are presented in table 4.

Table 4. Comparative Performance of Network Filtering Methods

Filtering Method	Blocking Effectiveness	CPU Efficiency	Cost-Effectiveness	Scalability	Bypass Resistance	Overall Score
Pi-hole	9	9	10	8	9	9.0
Firewall	7	5	3	9	8	6.4
Proxy server	8	4	4	5	5	5.2
Browser extension	6	8	9	4	3	6.0

The overall score was calculated as:

$$OPS = \frac{BES + CE + CSE + SS + BRS}{5}$$

For Pi-hole:

$$OPS_{\text{Pi-hole}} = \frac{9 + 9 + 10 + 8 + 9}{5} = 9.0$$

For the firewall:

$$OPS_{\text{Firewall}} = \frac{7 + 5 + 3 + 9 + 8}{5} = 6.4$$

For the proxy server:

$$OPS_{\text{Proxy}} = \frac{8 + 4 + 4 + 5 + 5}{5} = 5.2$$

ISSN 2723-6471

For the browser extension:

$$OPS_{\text{Browser}} = \frac{6+8+9+4+3}{5} = 6.0$$

Pi-hole obtained the highest overall score of 9.0, as shown in figure 5. Its strongest result was recorded for cost-effectiveness, with a normalized score of 10. It also achieved scores of 9 for blocking effectiveness, CPU efficiency, and bypass resistance. Its scalability score was 8, indicating that the proposed configuration was suitable for small- and medium-sized institutional networks, although its scalability remained below that of an enterprise firewall. The enterprise firewall achieved the highest scalability score of 9 and a bypass-resistance score of 8. However, its overall result was reduced by CPU efficiency and cost-effectiveness scores of 5 and 3, respectively. The additional processing requirements were associated with packet inspection, protocol enforcement, and optional Deep Packet Inspection.

The proxy server achieved a blocking-effectiveness score of 8 because it could apply application-layer and URL-based restrictions. However, its CPU-efficiency score was 4 because the system processed complete HTTP or HTTPS requests and required additional resources for authentication, caching, logging, and encrypted-traffic inspection. Its overall score was the lowest among the evaluated methods at 5.2. The browser extension produced relatively high CPU-efficiency and cost-effectiveness scores of 8 and 9. Nevertheless, it obtained a scalability score of 4 and bypass-resistance score of 3 because it operated only on individual browsers and could be disabled, removed, or bypassed by changing applications. The method therefore achieved an overall score of 6.0.

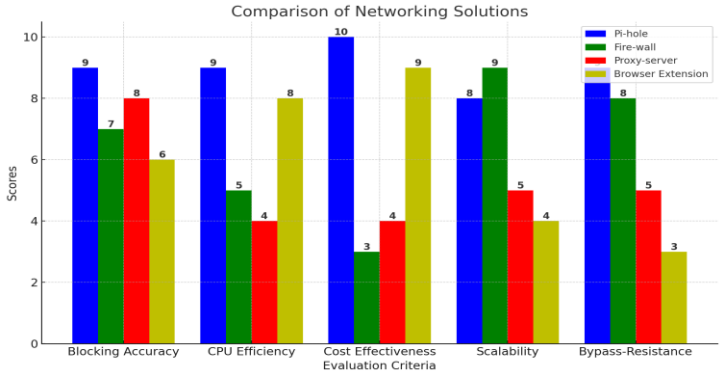

Figure 5. Normalized Performance Comparison of Network Filtering Methods

4.3.Operational Efficiency, Cost, and Scalability

Pi-hole recorded an average CPU utilization of 5%–10% while processing approximately 7,500 DNS requests per second. Accordingly, it obtained a normalized CPU-efficiency score of 9. Because the system processed domain-resolution requests rather than complete packet payloads or web content, its computational requirements remained lower than those of firewall- and proxy-based filtering methods. The processing-efficiency results are presented in table 5.

Table 5. Processing Efficiency of the Evaluated Filtering Methods

Method	Average CPU Usage	Requests per Second	CPU-Efficiency Score
Pi-hole	5–10%	Approximately 7,500	9
Firewall	30–50%	Approximately 5,000	5
Proxy server	50–70%	Approximately 3,000	4
Browser extension	5–10% per client	Approximately 3,000	8

The firewall required between 30% and 50% CPU utilization under the evaluated configuration. Its processing requirements increased when advanced traffic inspection and security rules were enabled. The proxy server recorded the highest CPU utilization, ranging from 50% to 70%, because it processed application-layer requests and performed additional logging and traffic-control operations. Browser extensions imposed comparatively low processing requirements, although their performance depended on the specifications and activity of each client device.

The cost evaluation included initial hardware expenditure, annual maintenance costs, and estimated annual power consumption. Pi-hole required an initial hardware investment of INR 1,500, no dedicated annual maintenance cost, and an estimated power cost of INR 150 per year. The cost-effectiveness inputs and results are presented in table 6.

Table 6. Cost-Effectiveness Inputs and Results

Method	Hardware Cost (INR)	Maintenance Cost (INR/year)	Power Cost (INR/year)	Cost-Effectiveness Score
Pi-hole	1,5	0	150	10
Firewall	100	10	4	3
Proxy server	80	8	2,4	4
Browser extension	0	0	800	9

The firewall had the highest infrastructure and recurring operational costs. The proxy server also required dedicated hardware and routine maintenance. Browser extensions had no centralized hardware cost, but their deployment required individual installation and management on each endpoint. Pi-hole achieved the highest cost-effectiveness score because it combined low initial cost, low power consumption, centralized administration, and high request-processing capacity. Scalability was assessed based on the estimated number of clients that could be supported before performance degradation occurred. The scalability and bypass-resistance results are presented in table 7.

Table 7. Scalability and Bypass-Resistance Results

Method	Maximum Supported Clients	Scalability Score	Bypass-Resistance Score
Pi-hole	500	8	9
Enterprise firewall	5	9	8
Proxy server	2	5	5
Browser extension	One installation per client	4	3

The firewall supported the largest number of clients and obtained the highest scalability score. Pi-hole supported approximately 500 clients in the evaluated configuration and obtained a score of 8. Proxy-based filtering could theoretically support a larger number of clients, but application-layer processing, encrypted-traffic inspection, and increased latency reduced its practical scalability score. Browser extensions required separate deployment and configuration for each endpoint and therefore recorded the lowest network-level scalability.

Pi-hole also obtained the highest bypass-resistance score of 9. Users required administrative access or alternative encrypted DNS mechanisms to circumvent the filtering policy. Firewall-based DNS redirection could be used to prevent clients from manually changing their resolver. Firewalls obtained a score of 8 because VPNs and encrypted tunnels remained potential bypass mechanisms. Proxy servers and browser extensions recorded lower scores because alternative proxies, VPN services, browser replacement, and extension deactivation could be used to avoid restrictions.

The combined results showed that Pi-hole provided the most balanced performance across the five evaluation dimensions. The proposed system did not exceed the enterprise firewall in maximum client capacity, but it produced higher results in CPU efficiency and cost-effectiveness while maintaining strong blocking and bypass-resistance performance. These results correspond to the comparative data and implementation evidence reported in the original manuscript.

5. Conclusion

This study developed and evaluated a Pi-hole-based DNS filtering architecture for controlling internet access in academic environments. The system was deployed on a Raspberry Pi 3B+ as the primary DNS resolver, enabling centralized filtering of domain requests through configurable blocklists, whitelists, and scheduled access policies. The implementation successfully recorded client queries, blocked restricted domains before resolution, forwarded permitted requests to an upstream DNS resolver, and provided network activity information through the Pi-hole monitoring dashboard.

The comparative evaluation showed that Pi-hole achieved the highest overall performance score of 9.0 across five assessment dimensions. It obtained scores of 9 for blocking effectiveness, CPU efficiency, and bypass resistance, 10 for cost-effectiveness, and 8 for scalability. The enterprise firewall provided greater client capacity, but its higher infrastructure cost and processing requirements reduced its overall score. Proxy-server filtering offered relatively strong content control but required greater computational resources, while browser extensions remained inexpensive and efficient at the endpoint level but were limited in scalability and resistance to circumvention.

The findings indicate that Pi-hole provides a balanced solution for small- and medium-sized academic networks requiring centralized, low-cost, and resource-efficient internet filtering. Its principal advantages are lightweight operation, network-wide deployment, customizable domain policies, and minimal dependence on individual client configuration. However, DNS filtering alone cannot fully prevent access through VPNs, encrypted DNS services, alternative resolvers, or direct IP connections. Therefore, institutional deployment should combine Pi-hole with forced DNS redirection, firewall rules, restricted administrative privileges, and regular blocklist maintenance.

Future work should evaluate the system using a larger number of clients, longer observation periods, and more diverse traffic conditions. Further development may also include automated domain classification, adaptive policy updates, role-based filtering, anomaly detection from DNS-query patterns, and integration with firewall or intrusion-detection systems. These extensions would improve the system's adaptability and strengthen its ability to respond to changing academic-network requirements and emerging security threats.

6. Declarations

6.1. Author Contributions

Conceptualization: R.E., P.G.; Methodology: R.E., P.G.; Software: D.K., M.N.; Validation: P.G., R.E.; Formal Analysis: R.E.; Investigation: D.K., M.N.; Resources: P.G., M.N.; Data Curation: D.K.; Writing – Original Draft Preparation: R.E.; Writing – Review and Editing: P.G., H.K.R.S.; Visualization: M.N.; All authors have read and agreed to the published version of the manuscript.

6.2. Data Availability Statement

The data presented in this study are available on request from the corresponding author.

6.3. Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

6.4. Institutional Review Board Statement

Not applicable.

6.5. Informed Consent Statement

Not applicable.

6.6. Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] W.-B. Hsieh, J.-S. Leu, and J.-I. Takada, "Use Chains to Block DNS Attacks: A Trusty Blockchain-based Domain Name System," *J. Commun. Networks*, vol. 24, no. 3, pp. 347–356, 2022, doi: 10.23919/JCN.2022.000009.

[2] J. Vanerio, C. Gyorgyi, and S. Schmid, "Poster: P4DME: DNS Threat Mitigation with P4 In-Network Machine Learning Offload," in *EuroP4 2023 - Proceedings of the 6th International Workshop on P4 in Europe*, 2023, pp. 53–56. doi: 10.1145/3630047.3630251.

[3] F. B. Wala, S. Campbell, and M. Kiran, "Insights into DoH: Traffic Classification for DNS over HTTPS in an Encrypted Network," in *SNTA 2023 - Proceedings of the 2023 on Systems and Network Telemetry and Analytics*, 2023, pp. 9–17. doi: 10.1145/3589012.3594895.

[4] M. Korczyński, Y. Nosyk, Q. Lone, M. Skwarek, B. Jonglez, and A. Duda, "Don't Forget to Lock the Front Door! Inferring the Deployment of Source Address Validation of Inbound Traffic," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2020, pp. 107–121. doi: 10.1007/978-3-030-44081-7_7.

[5] Z. Zhang, Y. Cheng, H. Xu, Z. Zhang, and C. Li, "Efficient DNS over HTTPS servers discovery method: A voting-based stacked ensemble model with secure connection metadata," *Comput. Networks*, vol. 259, 2025, doi: 10.1016/j.comnet.2025.111073.

[6] A. Salem and W. Elmedany, "Defending the Core: A Comprehensive Analysis of DDoS Attacks on DNS Infrastructure and Proactive Defense Strategies," in *IET Conference Proceedings*, 2023, pp. 439–444. doi: 10.1049/icp.2024.0964.

[7] A. Jare, S. Kolte, S. Kadam, V. Babar, P. Tekade, and D. Salunke, "DefendNet: Harnessing AI/ML for Dynamic DNS Filtering and Network Security," in *2024 IEEE International Conference on Blockchain and Distributed Systems Security, ICBDS 2024*, 2024. doi: 10.1109/ICBDS61829.2024.10837330.

[8] A. S. M. Rizvi, J. Mirkovic, J. Heidemann, W. Hardaker, and R. Story, "Defending Root DNS Servers against DDoS Using Layered Defenses (Extended)," *Ad Hoc Networks*, vol. 151, 2023, doi: 10.1016/j.adhoc.2023.103259.

[9] A. Pagan and K. Elleithy, "A Multi-Layered Defense Approach to Safeguard against Ransomware," in *2021 IEEE 11th Annual Computing and Communication Workshop and Conference, CCWC 2021*, 2021, pp. 942–947. doi: 10.1109/CCWC51732.2021.9375988.

[10] M. A. Balais, Z. K. T. Alcazar, R. P. V Martin, V. E. M. Tulabot, and C. A. Z. Valerio, "Wireless Network Infrastructure and Security Enhancement for St. Joseph School," in *ACM International Conference Proceeding Series*, 2023, pp. 249–259. doi: 10.1145/3634814.3634846.

[11] D. Gong et al., "Practical verifiable in-network filtering for DDoS Defense," in *Proceedings - International Conference on Distributed Computing Systems*, 2019, pp. 1161–1174. doi: 10.1109/ICDCS.2019.00118.

[12] B. Yu, G. Graciani, A. Nascimento, and J. Hu, "Cost-adaptive Neural Networks for Peak Volume Prediction with EMM Filtering," in *Proceedings - 2019 IEEE International Conference on Big Data, Big Data 2019*, 2019, pp. 4208–4213. doi: 10.1109/BigData47090.2019.9006188.

[13] I. Gladin, V. Ewards, and S. Terence, "Domain Name Server Filtering Service Using Threat Intelligence and Machine Learning Techniques," in *Lecture Notes in Networks and Systems*, 2024, pp. 529–540. doi: 10.1007/978-981-97-7710-5_40.

[14] N. Kostopoulos, D. Kalogeras, D. Pantazatos, M. Grammatikou, and V. Maglaris, "SHAP Interpretations of Tree and Neural Network DNS Classifiers for Analyzing DGA Family Characteristics," *IEEE Access*, vol. 11, pp. 61144–61160, 2023, doi: 10.1109/ACCESS.2023.3286313.

[15] F. Gao, W. Han, and W. Ou, "BotHunter: Distributed Malicious Domain Name Detection Model Based on Deep Learning and Blockchain," in *Advances in Transdisciplinary Engineering*, 2024, pp. 903–913. doi: 10.3233/ATDE231270.

[16] D. Tatang, F. Quinkert, and T. Holz, "Below the Radar: Spotting DNS Tunnels in Newly Observed Hostnames in the Wild," in *eCrime Researchers Summit, eCrime*, 2019. doi: 10.1109/eCrime47957.2019.9037595.

[17] M. Khurana, P. Malik, and S. Puneet, "Network security monitoring (NSM): Can it be effective in a world with encrypted traffic?," in *Proceedings of International Conference on Computation, Automation and Knowledge Management, ICCAKM 2020*, 2020, pp. 140–144. doi: 10.1109/ICCAKM46823.2020.9051536.

[18] A. Carranza, M. Bustamante, H. Carranza, C. DeCusatis, and J. Islam, "Firewall and DNS Filtering Penetration Tests using Parrot OS," in *Proceedings of the LACCEI international Multi-conference for Engineering, Education and Technology*, 2022. doi: 10.18687/LACCEI2022.1.1.675.

[19] A. M. H. Saeed, D. Wang, H. A. M. Alnedhari, K. Mei, and J. Wang, "A Survey of Machine Learning and Deep Learning Based DGA Detection Techniques," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2022, pp. 133–143. doi: 10.1007/978-3-030-97774-0_12.

[20] S. Fujimura and M. Okumura, "Elevating Network Performance - Part II from IPv6 and Network Security Deployments," in *Proceedings ACM SIGUCCS User Services Conference*, 2025, pp. 5–8. doi: 10.1145/3675229.3712525.

[21] D. Zeng, M. Dawood, S. Tu, M. Al-Antary, M. Waqas, and A. Namoun, "SmartDoH: A Deep Learning Solution for Secure and Efficient DNS-over-HTTPS Traffic Analysis," in *2025 3rd International Conference on Big Data and Privacy Computing, BDPC 2025*, 2025, pp. 35–40. doi: 10.1109/BDPC63545.2025.11135907.

[22] Y. Su, B. Peng, and X. Li, "Fast Illegal Webpage Detection Algorithm Based on Massive Domain Name Resolution Records," in *Proceedings - 2022 IEEE International Conference on Big Data, Big Data 2022*, 2022, pp. 4313–4322. doi: 10.1109/BigData55660.2022.10020782.

[23] R. G. Edouard Bouda, A. Sere, and F. Ouedraogo, "Improved Domain Name System Threat Detection using an RPZ and DGA-based approach incorporating Machine Learning," in *International Conference on Artificial Intelligence, Computer, Data Sciences, and Applications, ACDSA 2025*, 2025. doi: 10.1109/ACDSA65407.2025.11166262.

[24] U. Sirisha, C. K. Kumar, S. C. Narahari, and P. N. Srinivasu, "An iterative PRISMA review of GAN models for image processing, medical diagnosis, and network security," *Computers, Materials & Continua*, vol. 82, no. 2, pp. 1757–1810, 2025. Doi: 10.32604/cmc.2024.059715.