Integrated Random Forest over Network-Based Firewall Implementation on Mikrotik Network for Phishing Filtering

Agustinus Noertjahyana, Stephanus Antonius Ananda, Juven Axel Wedianto

Tenggilis Utara II / 7., Surabaya, Indonesia

DOI: https://doi.org/10.51244/IJRSI.2026.13010117

Received: 22 January 2026; Accepted: 27 January 2026; Published: 06 February 2026

ABSTRACT

Phishing attacks have escalated significantly, necessitating robust yet cost-effective network security solutions. Addressing the limitations of static blocking in Mikrotik and the prohibitive costs of dedicated hardware firewalls, this paper proposes an automated filtering system that integrates the Random Forest Machine Learning algorithm with Mikrotik architecture via a Python-based RouterOS API. The proposed system enables dynamic monitoring of DNS caches to automatically identify and block phishing domains through firewall drop rules. Experimental evaluation involved feature selection, offline validation, and real-world deployment. Results demonstrate that the 10-feature model delivers the optimal balance between accuracy and latency, achieving 90% accuracy with an average classification time of 11.5 seconds. In live network testing, the system successfully detected and mitigated phishing threats within 7 to 21 seconds. While CPU utilization increased by 7-40% during active detection, memory efficiency remained stable. This study validates that integrating Random Forest with Mikrotik offers an adaptive, scalable, and economical solution for network-based phishing prevention.

Keywords: Cognitive Security, Firewall Automation, Mikrotik, Phishing Filtering, Random Forest, Network Security

INTRODUCTION

The rapid expansion of the internet has revolutionized business and education sectors but has concurrently introduced complex security vulnerabilities. Among these, phishing remains a predominant threat, where malicious entities impersonate trusted organizations to illicitly acquire sensitive data [1]. Reports from the Anti-Phishing Working Group (APWG) indicate a dramatic surge in phishing sites, escalating from approximately 150,000 in 2013 to over 1.5 million in early 2023 [2]. These attacks are typically propagated through deceptive emails and messaging platforms, bypassing traditional security perimeters.

While conventional firewalls provide fundamental protection via IP-based blocking, they often lack the intelligence to detect sophisticated, content-based threats. Next-Generation Firewalls (NGFW) offer advanced capabilities such as Deep Packet Inspection (DPI) and AI-driven intrusion detection [3]; however, the high acquisition and deployment costs render them inaccessible for many small-to-medium scale institutions. Consequently, there is a pressing need for a cost-effective alternative that leverages existing infrastructure without compromising security efficacy.

Mikrotik routers present a viable solution, functioning as versatile network gateways capable of firewall operations [4]. Previous studies have explored manual firewall configurations for DNS blocking [5]. To enhance this approach, the RouterOS Application Programming Interface (API) can be exploited to automate rule management, transforming static routers into dynamic defense systems [6]. Furthermore, the integration of Machine Learning (ML) algorithms facilitates the automated classification of phishing patterns. Comparative research indicates that the Random Forest algorithm outperforms Support Vector Machine (SVM) in domain classification, offering superior accuracy (96.2%) and processing speed [7].

In this paper, it proposes an automated Network-based Phishing Filtering system by integrating the Random Forest algorithm into the Mikrotik environment. Unlike static blacklisting, this system utilizes Python to interface with RouterOS-API for real-time dynamic firewall management. The primary objective is to design a low-cost, adaptive security mechanism capable of mitigating phishing attacks with minimal impact on network performance.

System Model and Architecture

The proposed system operates on a detection-and-mitigation framework designed to intercept client access to malicious domains. The architectural workflow is illustrated in Fig. 1.

A. Operational Workflow The process initiates when a client device attempts to resolve a domain name (e.g., "blackhat.com"). This request is cached within the Mikrotik DNS service. An external control unit, executing a Python-based automation script, periodically polls the DNS cache via the RouterOS-API. The system extracts domain attributes and classifies them as either "phishing" or "legitimate" using a pre-trained Random Forest model. Upon identifying a phishing domain, the system triggers the Mikrotik firewall to append the domain to a specific "drop" address list. Consequently, the firewall enforces a blocking rule, terminating the connection and preventing data exfiltration.

B. Data Acquisition and Caching Strategy To ensure computational efficiency, the operator filters DNS cache entries specifically for 'A' records with a Time-to-Live (TTL) of one hour. This constraint ensures that the system focuses on active, recently accessed domains while discarding stale data. Following classification, processed domains are archived in a local detection log to prevent redundant processing and to maintain a historical record of identified threats. It is assumed that this protection layer applies to standard network configurations and does not inspect traffic encrypted via VPNs or DNS over HTTPS (DoH).

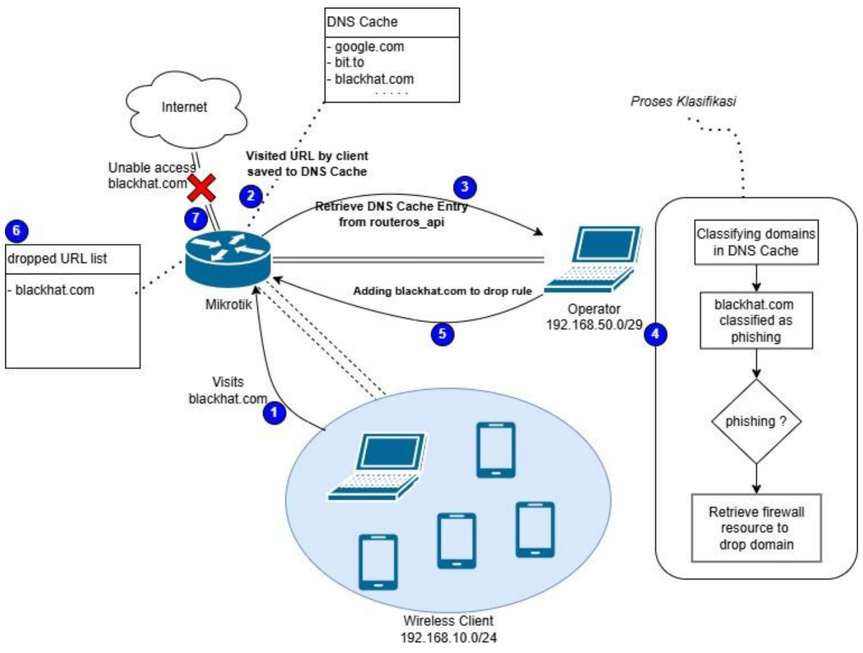

DNS Cache
- google.com
- bit.to
- blackhat.com
Proses Klasifikasi
Internet
Unable access blackhat.com
Visited URL by client saved to DNS Cache
2
3
Retrieve DNS Cache Entry from routeros_api
6
7
dropped URL list
- blackhat.com
Mikrotik
Adding blackhat.com to drop rule
5
Operator 192.168.50.0/29
4
Classifying domains in DNS Cache
blackhat.com classified as phishing
phishing ?
Retrieve firewall resource to drop domain
Visits blackhat.com
1
Wireless Client 192.168.10.0/24

Fig. 1 Operational workflow of the proposed filtering system

Dataset Preparation and Feature Extraction

To ensure the machine learning model generalizes well to real-world network traffic, a comprehensive dataset comprising both malicious and legitimate domains was constructed. The data preparation pipeline involves acquisition, feature engineering, and rigorous preprocessing.

A. Data Acquisition The primary dataset was aggregated from multiple sources to create a balanced representation of network threats. The base phishing data was sourced from the Kaggle repository, which was restructured from a URL-based format into a domain-based format to align with DNS-level filtering requirements. To ensure the inclusion of emerging threats, this dataset was augmented with recent phishing feeds obtained from OpenPhish.

Conversely, samples of legitimate domains were compiled from supplementary Kaggle datasets and enriched with local DNS cache logs recorded directly from the Mikrotik router. This inclusion of local traffic data ensures that the model is attuned to the specific browsing patterns of the target network environment.

B. Feature Engineering and External Intelligence Feature extraction focused on identifying attributes that differentiate phishing domains from legitimate ones. The features are categorized into two groups:

1. **Lexical Features:** These attributes are derived directly from the domain string, such as domain length and the specific Top-Level Domain (TLD) used.

2. **External Intelligence:** To enhance detection accuracy, the system leverages external APIs to gather contextual metadata:

- **Domain Age:** Retrieved via the **Whois API** to identify newly registered domains, which are often correlated with phishing campaigns.

- **Reputation Score:** Obtained from **OpenPageRank** to assess the domain's authority and traffic ranking.

- **Index Status:** Search engine queries (via **Google Custom Search** or **DuckDuckGo**) are executed to verify if the domain is indexed, as legitimate sites are typically indexed while short-lived phishing sites are not.

It is acknowledged that relying on free-tier APIs introduces constraints regarding latency and service dependency, which were mitigated by caching strategies but remain a factor in real-time performance.

C. Data Preprocessing Following data collection, a rigorous preprocessing phase was implemented to ensure data quality. This involved integrating new domain samples and encoding categorical features into numerical formats suitable for the Random Forest algorithm.

To address data inconsistencies, noise reduction techniques were applied, including the imputation of missing values, correction of erroneous encodings, and the elimination of duplicate entries. The final processed dataset was exported as a CSV file, serving as the foundational input for model training.

Algorithm Implementation and System Integration

This section delineates the technical implementation of the proposed phishing detection system, focusing on the configuration of the machine learning classifier and its operational integration with the network infrastructure. The implementation framework is divided into three critical stages: the optimization of the Random Forest algorithm via hyperparameter tuning, the strategic selection of features to balance computational efficiency with accuracy, and the development of an automated mitigation mechanism using the RouterOS API to enforce firewall policies in real-time.

A. Model Configuration and Training Strategy

The core classification engine employs the Random Forest algorithm, selected for its ensemble architecture which aggregates decisions from multiple decision trees. This approach significantly mitigates the risk of overfitting and ensures prediction stability, particularly when dealing with high-dimensional domain data. For the training phase, the prepared dataset was partitioned using an 80:20 ratio, allocating 16,000 domains for training and 4,000 domains for testing. This split ensures sufficient data volume for the model to learn complex patterns while reserving a substantial portion for unbiased validation.

B. Hyperparameter Optimization

To maximize detection accuracy, the model underwent rigorous tuning using Randomized Search Cross-Validation (CV). Unlike manual tuning, this method systematically explores a predefined grid of hyperparameters to identify the optimal configuration. As detailed in Table 1, the optimization process determined that a configuration of 300 estimators (n_estimators), a maximum depth of 50, and a minimum sample split of 2 yielded the highest estimated accuracy of 97%. These parameters were subsequently locked for the final deployment to ensure consistent performance.

Table 1. Parameter generated from random search

Factor	**Value**
n_estimator	300
min_samples_split	2
min_samples_leaf	1
max_features	sqrt
max_depth	50
bootstrap	TRUE

C. Feature Selection for Efficiency

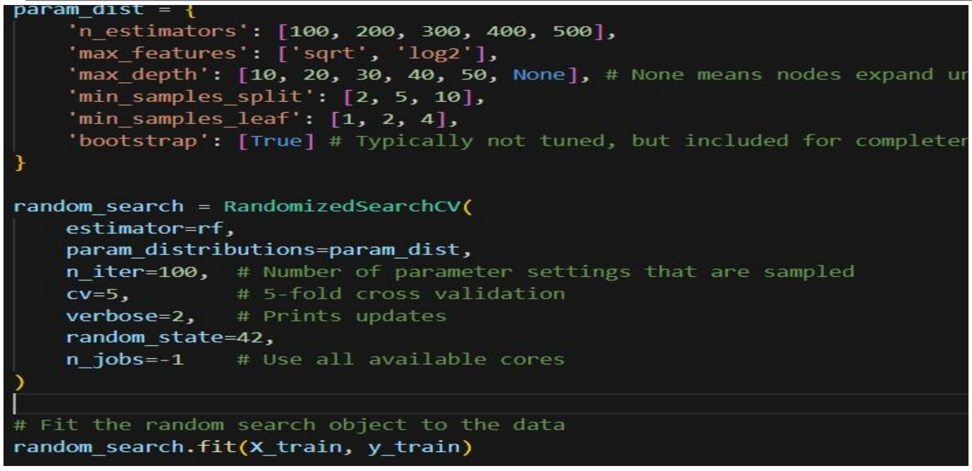
Based on the feature importance analysis (which ranked page score as the most critical attribute), three distinct model variants were developed to evaluate the trade-off between computational speed and classification accuracy:

1. **Full-Feature Model:** Incorporates all 19 extracted attributes to capture maximum detail.
2. **10-Feature Model:** Retains only the top 10 most discriminative features based on importance ranking.
3. **5-Feature Model:** Utilizes the top 5 features, prioritized for maximum processing speed. All variants utilized the identical hyperparameters established in the previous step to ensure a controlled comparative analysis.

D. Automated Mitigation Mechanism

The system features a real-time integration module connecting the Python backend with the Mikrotik RouterOS via API. Upon classifying a domain as "phishing," the system executes an automated mitigation sequence:

1. **Tagging:** The malicious domain is identified by the classifier.
2. **API Execution:** A command is sent to the router to append the domain to a specific Address List designated as *"phishing-sites"*.
3. **Enforcement:** The Mikrotik firewall, pre-configured with a "drop" rule for this address list, immediately terminates any connection attempts to the tagged IP addresses. This automated loop ensures that protection is applied at the network layer within seconds of detection, effectively blocking the threat before the client's browser can load the malicious content.



```
param_dist = {
    'n_estimators': [100, 200, 300, 400, 500],
    'max_features': ['sqrt', 'log2'],
    'max_depth': [10, 20, 30, 40, 50, None], # None means nodes expand ur
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True] # Typically not tuned, but included for completer
}

random_search = RandomizedSearchCV(
    estimator=rf,
    param_distributions=param_dist,
    n_iter=100,  # Number of parameter settings that are sampled
    cv=5,        # 5-fold cross validation
    verbose=2,   # Prints updates
    random_state=42,
    n_jobs=-1    # Use all available cores
)

# Fit the random search object to the data
random_search.fit(X_train, y_train)
```

Fig. 2 Used Random Search combinations

In Table 1, the Randomized Search algorithm successfully identified the optimal parameters from Fig. 2, achieving an estimated accuracy of 97%. These parameters were subsequently applied as the fixed configuration for the initial model, serving as the full-feature variant.

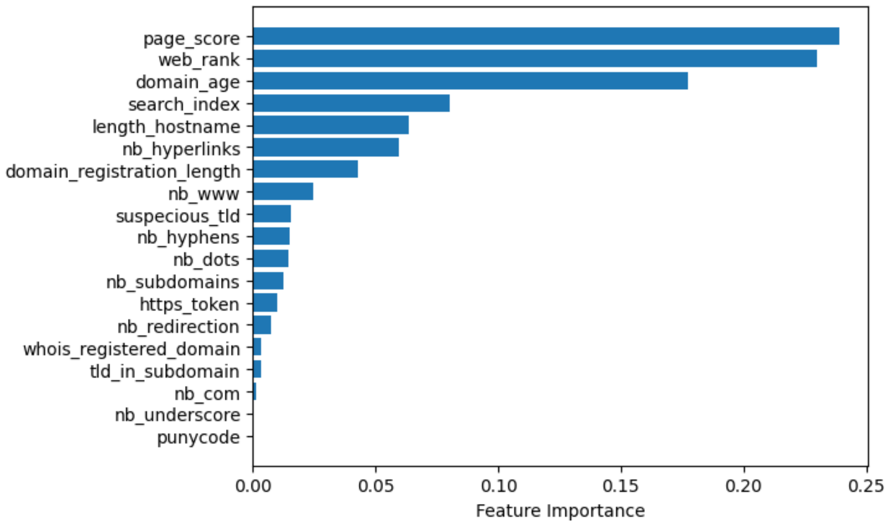


Fig. 3 Feature rankings from feature importance results

In Fig. 3, the feature importance results identified page_score as the most significant attribute. In accordance with the study's methodology, a comparative analysis was conducted using models restricted to the top 5 and top 10 features, selected based on their descending order in the importance hierarchy. To ensure consistency, both reduced-feature models utilized the identical hyperparameters established for the full-feature model.

RESULT AND DISCUSSION

This section presents a comprehensive evaluation of the proposed system, analyzing the impact of feature selection on classification accuracy, computational efficiency, and real-time operational stability.

A. Feature Importance Analysis

To determine the most effective attributes for phishing detection, the Random Forest algorithm's feature importance metric was utilized. As illustrated in Fig. 3, reputation-based features proved significantly more discriminative than purely lexical attributes. Specifically, page_score emerged as the dominant predictor, followed by web_rank and domain_age. This indicates that a domain's lack of historical reputation and search engine indexing is a stronger indicator of malicious intent than the length or structure of the domain name itself.

B. Model Performance Evaluation

The classification performance was assessed using three model variations (Full-Feature, 10-Feature, and 5-Feature) across two distinct datasets: an internal test set and an external dataset derived from real-world DNS cache logs.

1. Internal Dataset Performance: As detailed in Table 2, the Full-Feature model achieved the highest performance with an accuracy of 96.5% and an F1-score of 0.965. However, the 10-Feature model demonstrated comparable proficiency, achieving 96.2% accuracy. The 5-Feature model, while decent, exhibited a slight drop in precision, indicating that removing too many features leads to information loss.

Table 2. Evaluation of test data in dataset

Category	Accuracy	Precision	Recall	F1
Full Feature (19)	0.965	0.967	0.964	0.965
5 Feature	0.956	0.955	0.956	0.955
10 Feature	0.962	0.962	0.962	0.962

When tested against external data (Table 3), all models experienced a slight performance degradation (approximately 5%), attributed to the diverse characteristics of live network traffic. However, the relative performance ranking remained consistent, with the 10-Feature model maintaining a robust accuracy of 91% and an F1-score of 0.92, demonstrating its generalization capability.

Table 3. Evaluation of test data outside dataset

Category	Accuracy	Precision	Recall	F1
Full Feature (19)	0.91	0.87	0.97	0.92
5 Feature	0.873	0.81	0.97	0.88
10 Feature	0.9	0.86	0.97	0.91

C. Computational Efficiency and Trade-offs

In a network-based filtering system, processing latency is as critical as detection accuracy. Table 4 presents the offline processing time and error rates for processing 200 domains.

- **The 5-Feature Model** was the fastest, averaging 9.97 seconds per URL. However, this speed came at the cost of a high False Positive Rate (FPR) of 0.36, which poses a risk of blocking legitimate traffic.
- **The Full-Feature Model** minimized the FPR to 0.13 but was the least efficient, requiring 14.3 seconds per URL.
- **The 10-Feature Model** offered the optimal trade-off, achieving a balanced processing time of 11.5 seconds while maintaining a controlled FPR of 0.2. Based on this analysis, the 10-Feature model was selected for the final deployment.

Table 4. Evaluation of another test data for offline testing

Category	Total Time	Average time/url	FPR	FNR
Full Feature (19)	47.49 min	14.3 sec	0.13	0.4

5 Feature	33.15 min	9.97 sec	0.36	0.8
10 Feature	38.03 min	11.5 sec	0.2	0.5

D. Qualitative Error Analysis

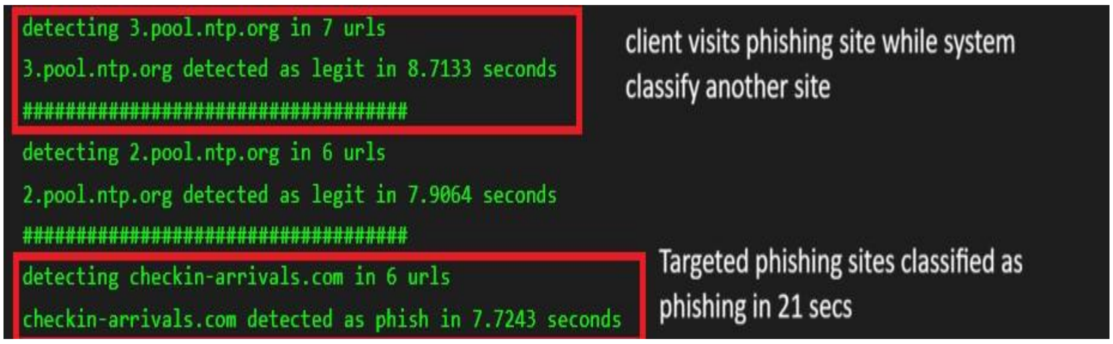
An in-depth analysis of misclassification cases reveals specific limitations in the detection logic. As observed in the experimental scenarios:

- **False Positives:** Legitimate domains were occasionally misclassified as phishing because they were absent from the OpenPageRank database. The model incorrectly inferred that a lack of reputation data equated to malicious intent.

- **False Negatives:** Some sophisticated phishing domains evaded detection because they possessed legitimate attributes, such as a long registration age (domain_age). In these cases, the phishing content was likely hosted on a specific URL path rather than the root domain, which the current domain-based filtering approach may overlook.

E. Real-Time Deployment and Resource Utilization

The system's responsiveness was evaluated in a live network environment. As shown in the detection timeline (Fig. 4), the system successfully identified and blocked a target phishing domain (checkin-arrivals.com) within 7 seconds of processing. However, due to the sequential processing queue, the total latency from the client's request to the final block action was approximately 21 seconds.

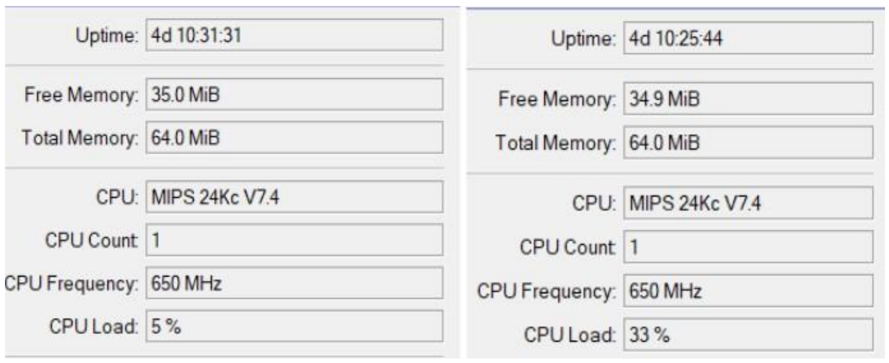


```
detecting 3.pool.ntp.org in 7 urls
3.pool.ntp.org detected as legit in 8.7133 seconds
####################################
detecting 2.pool.ntp.org in 6 urls
2.pool.ntp.org detected as legit in 7.9064 seconds
####################################
detecting checkin-arrivals.com in 6 urls
checkin-arrivals.com detected as phish in 7.7243 seconds
```

client visits phishing site while system classify another site
Targeted phishing sites classified as phishing in 21 secs

Fig. 4 Detecting domain timeline

Regarding hardware resources (Fig. 5), the integration of the Python-based polling mechanism caused the CPU load on the Mikrotik router to fluctuate between 7% and 56%, compared to a baseline of 3-12%. This increase is primarily driven by the continuous DNS cache retrieval process. Despite the CPU spike, memory utilization remained efficient, with the router retaining approximately 35 MiB of free memory, ensuring that core network functions were not compromised.



Uptime: 4d 10:31:31
Free Memory: 35.0 MiB
Total Memory: 64.0 MiB
CPU: MIPS 24Kc V7.4
CPU Count: 1
CPU Frequency: 650 MHz
CPU Load: 5 %

Uptime: 4d 10:25:44
Free Memory: 34.9 MiB
Total Memory: 64.0 MiB
CPU: MIPS 24Kc V7.4
CPU Count: 1
CPU Frequency: 650 MHz
CPU Load: 33 %

Fig. 5 Mikrotik CPU prior & while system running

F. Limitations and Scalability Analysis

While the system demonstrates effective detection, several limitations are acknowledged. First, the dependency on external free-tier APIs (Whois, OpenPageRank) introduces variable latency that cannot be fully controlled. In a high-traffic production environment, this could lead to bottlenecks. Second, the observed CPU peak of 56% on the test device suggests that lower-end MikroTik models (e.g., hAP lite series) might experience performance degradation under similar loads.

Furthermore, the current evaluation metrics focused on Accuracy, Precision, and F1-Score to prioritize the balance between False Positives and False Negatives. Although ROC/AUC analysis was not included in this study, the high F1-score (0.91) sufficiently indicates the model's robustness in handling the imbalanced nature of phishing datasets. Lastly, the system is designed for standard DNS traffic and does not currently inspect encrypted queries via DNS over HTTPS (DoH) or VPNs, which remains a challenge for network-level filtering.

CONCLUSION

This study successfully designed and implemented an automated phishing filtering system by integrating the Random Forest algorithm with Mikrotik RouterOS. The experimental results highlight the critical role of feature selection in optimizing network security tools. While the Full-Feature model achieved the highest theoretical accuracy, its processing latency made it less ideal for real-time deployment. Conversely, the 5-Feature model, despite being the fastest, exhibited a high False Positive Rate that could disrupt legitimate user activity.

The 10-Feature model emerged as the optimal configuration, striking a strategic balance between classification accuracy (approximately 90%) and processing speed. In offline testing, this model maintained an average classification time of 11.5 seconds. During real-world deployment, the system demonstrated capable responsiveness, detecting and blocking threats within a range of 7 to 20 seconds once sufficient log data was accumulated.

Regarding resource utilization, the integration of the Python-based polling mechanism did result in a notable increase in CPU load (spiking up to 56%) due to continuous DNS cache retrieval. However, memory usage remained highly efficient, retaining substantial free space (34.9 MiB) and ensuring that the router’s core routing functions were not compromised.

In conclusion, this research validates that integrating lightweight Machine Learning models with existing network infrastructure is a feasible solution for small-to-medium institutions. Future work will focus on optimizing scalability by implementing asynchronous DNS monitoring to reduce CPU overhead. Additionally, offloading the inference process to a dedicated edge server while keeping enforcement on the router could solve hardware limitation issues. Further research should also explore mechanisms to detect encrypted DNS traffic (DoH/DoT) to enhance resilience against modern evasion techniques.

REFERENCES

1. Cloudflare, "What is phishing? | Phishing attack prevention," Cloudflare Learning Center. [Online]. Available: https://www.cloudflare.com/learning/access-management/phishing-attack/. [Accessed: Jan. 23, 2026].
2. Petrosyan, "Number of phishing domain names worldwide from 1st quarter 2013 to 4th quarter 2024," Statista, Sep. 1, 2025. [Online]. Available: https://www.statista.com/statistics/266155/number-of-phishing-domain-names-worldwide/.
3. RocketMeUpNetworking, "Next-Generation Firewalls (NGFW) — Beyond Traditional Security Measures," Medium, Oct. 28, 2024. [Online]. Available: https://medium.com/@RocketMeUpNetworking/next-generation-firewalls-ngfw-beyond-traditional-security-measures-d2ebe6dcab8b.
4. RevoU, "Apa itu Mikrotik? Pengertian, Fungsi, dan Jenisnya," RevoU.co. [Online]. Available: https://www.revou.co/kosakata/Mikrotik.

5. H. P. Fitrian, F. Dani, I. Fadilah, R. D. Fauzan, and M. R. Ardhyansyah, "Implementasi Mikrotik firewall sebagai solusi filtering situs judi online dalam jaringan," JATI (Jurnal Mahasiswa Teknik Informatika), vol. 9, no. 1, pp. 1685–1691, 2025. DOI: 10.36040/jati.v9i1.12781.
6. B. Māris, "API Implementation Details," Mikrotik Documentation, Feb. 27, 2025. [Online]. Available: https://help.Mikrotik.com/docs/spaces/ROS/pages/47579160/API.
7. T. Jason, "Prediksi phishing menggunakan machine learning support vector, dan random forest," Undergraduate Thesis, Informatics Dept., Petra Christian University, Surabaya, Indonesia, 2024. [Online]. Available: https://dewey.petra.ac.id/digital/view/59221.
8. Mostafavi, "Farah_phishing Dataset, Version 1," Kaggle Datasets, Nov. 2024. [Online]. Available: https://www.kaggle.com/datasets/akrammostafavi/farah-phishing.
9. D. Kurniawan, Pengenalan Machine Learning dengan Python, 6th ed. Jakarta, Indonesia: Elex Media Komputindo, 2020.