Edge-Based Policy Caching for Low Latency Security Enforcement in Hybrid Clouds

Geonmin Kim
Dept. of Software Engineering
Chonnam National University
Gwangju, Korea
204869@jnu.ac.kr

Yejin Kim
Dept. of Software Engineering
Chonnam National University
Gwangju, Korea
ye031010@jnu.ac.kr

Eunseong Lee
Dept. of Software Engineering
Chonnam National University
Gwangju, Korea
200750@jnu.ac.kr

Hyeonji Jang
Dept. of Artificial Intelligence
Chonnam National University
Gwangju, Korea
gka1225@jnu.ac.kr

Kyungbaek Kim
Dept. of Artificial Intelligence Convergence
Chonnam National University
Gwangju, Korea
kyungbaekkim@jnu.ac.kr

***Abstract*—With the spread of big data and the development of cloud computing technology, many enterprises are switching to a hybrid-cloud environment that combines on-premises infrastructure and public and private clouds. However, this integration of heterogeneous infrastructures reveals the limitations of existing boundary-based security models and exposes policy-evaluation latency. In particular, delays in evaluating and applying security policies across infrastructures create bottlenecks in real-time detection and response systems. Recently, new security approaches based on artificial intelligence and generative AI have been proposed for modern hybrid-cloud architectures. Therefore, this study proposes a hybrid-cloud security model that integrates machine learning-based anomaly detection, real-time log analysis, automatic policy generation, and policy caching to the edge. The proposed model comprises a traffic gateway layer that centralizes and collects API request metadata, a policy generation layer that repeatedly trains log data from collected requests and generates new policies, and a policy evaluation layer that minimizes response delay by using edge and central policy agents in a dual configuration. This model introduces the concept of risk-aware security policy caching, which blocks high-frequency attacks at the edge while simultaneously improving detection rate and response speed. Through this, major performance indicators such as anomaly detection accuracy and policy application delay time were quantitatively evaluated. Over 10 iterative experimental rounds, the proposed model achieved an average attack detection rate of 89.75% outperforming the centralized base model at 85.91% while reducing the block time of attack logs by 20.07% through policy caching.**

Keywords—hybrid-cloud security, edge computing, security policy caching, machine learning, anomaly detection, security automation

I. INTRODUCTION

With the emergence of Big Data and the rapid advancement of Cloud Computing technology[1-3], many enterprises around the world are transforming the way they store, process, and operate their data. Traditionally, they stored data in an on-premises environment, which involves storing and managing data internally within their organization, offering strong advantages in terms of data security and physical control. However, building an on-premises environment requires significant initial setup costs and has inherent limitations in flexibility and scalability. Moreover, the ongoing costs for maintenance of hardware and network infrastructure are becoming increasingly burdensome for enterprises, especially as data volume grows rapidly due to big data.

Against this backdrop, Cloud-Computing technology[4] has emerged as a new paradigm of managing data, making the exclusive use of on-premises environments inefficient. Enterprises have begun transitioning to more flexible and scalable alternatives through cloud-based systems[5]. Cloud Computing technology dramatically reduces the initial investment required for large-scale storage and enables dynamic resource allocation based on demand, thereby maximizing the operational efficiency of data storage systems[6]. In response to these trends, many organizations are now adopting hybrid-cloud architectures that combine cloud systems with their on-premises environment[7].

A hybrid cloud architecture is an environment in which public clouds, private clouds, and on-premise infrastructures are interconnected via networks to enable seamless interoperability of data and applications[8]. Public cloud refers to services offered by external providers such as Amazon Web Services (AWS)[9], Microsoft Azure[10], and Google Cloud Platform (GCP)[11]. In contrast, a private cloud is an internally managed cloud infrastructure operated by a specific organization. The hybrid cloud model combines public and private cloud services with on-premise resources, allowing organizations to allocate resources flexibly and optimize their data storage and processing strategies.

However, the integration of on-premises and cloud systems introduces a new set of challenges[12-13]. The fusion of heterogeneous environments blurs traditional network boundaries and increases structural complexity, thereby exposing new security vulnerabilities. In particular, existing static-rule-based security systems or perimeter-based firewall systems are limited in their ability to detect and respond to advanced cyber threats such as polymorphic attacks, advanced persistent threats(APTs), and evasion attacks. To address these challenges, there is a growing demand for more integrated and intelligent security frameworks. Recently, security research based on machine learning(ML) and artificial intelligence(AI) has intensified with efforts to overcome existing limitations in anomaly detection, pattern-based detection, and policy automation[14-18]. However, existing studies have focused on improving detection performance and data-based threat recognition without systematically addressing the latency of policy evaluation and enforcement. Consequently, large-scale hybrid-cloud environments face serious operational delays. This affects the actual large-scale hybrid cloud operating environments heavily. Accordingly, this study proposes a new concept called Risk-Aware Security Policy Caching, which applies the

core ideas of service caching principles from edge computing and risk-based caching to the system and aims to fundamentally improve the limitations of existing studies.

II. Background

The priority in designing a hybrid cloud security architecture is to understand the structural and operational differences among heterogeneous infrastructures and to devise a consistent mechanism for applying security policies across them. This chapter compares and examines differences between design criteria and API-based operational strategies from a security perspective, focusing on the main characteristics of on-premises, public clouds, private clouds, and hybrid cloud environments.

A. Security Structure and Challenges for On-Premise Environments

The on-premise environment is a model that directly owns and operates physical assets and network infrastructure within an organization and provides high reliability based on total control over data and isolation in terms of security. In general, APIs are designed for communication between in-house systems and are built around RESTful architectures, internal DNS-based address systems, and links with in-house authentication systems.

However, because such systems assume a closed network, they remain vulnerable to insider attacks and configuration errors like omission of authentication and authorization procedures, non-application of encryption, and neglect of legacy APIs[19]. In particular, plaintext HTTP communication is often employed on an internal network, enabling attackers to harvest sensitive information via ARP spoofing or packet sniffing [20]. In addition, real-time attack detection and accident response are difficult if Access Control List(ACL) management for API calls is not automated or Security Information and Event Management(SIEM) linkage is insufficient. Therefore, although an on-premises environment offers physical isolation, it is secure only when policy maintenance and integrated log-analysis systems are implemented concurrently.

B. Security Differences Between Public and Private Clouds

Public clouds and private clouds are both cloud computing models, but they have fundamental differences in design and application of security policies. These differences are key factors that cause problems in policy consistency and real-time security applications in cloud environments.

Public clouds are operated by external cloud service providers(CSPs) such as Amazon Web Services(AWS), Microsoft Azure, and Google Cloud Platform(GCP). Resource access and security policy management are controlled through integrated APIs provided by these CSPs. These APIs provide advanced policy interfaces such as IAM-based authentication and authorization systems, OAuth2.0, OpenID Connect(OIDC), and key-based access control. However, the policy grammar and application methods differ for each platform[21]. For example, even if the access policy is the same, the configuration methods and evaluation logic of AWS IAM policies, Azure Role Assignment, and GCP IAM Binding are different, making it difficult to consistently apply security policies across multiple clouds.

In addition, public clouds provide various security components(e.g., API Gateway, CASB, SIEM API) to support automated policy evaluation, but these services are mainly operated in a centralized structure, and since each request goes through a fixed evaluation pipeline, this could cause a delay in policy application when high-frequency requests occur. In particular, if the policy update cycle is dependent on CSPs, there is a structural limitation that it is difficult to reflect the policy in real time, even after a threat occurs.

Private clouds are independently built and operated by a single organization, and security policies are also configured based on internal standards and business rules of the organization. They have the advantage of ensuring a high level of user customization and autonomous control authority, but the policy automation and real-time detection and application systems are often relatively inadequate compared to public clouds. In particular, the absence or limited scope of log collection and analysis infrastructure is a major obstacle to securing real-time and scalable attack detection. Furthermore, in private clouds, security quality depends on the operator's technical capabilities, and system configuration and automation are often limited than in public clouds.

C. Hybrid Cloud Complexity and Security Response

A hybrid cloud is a structure in which on-premises, public, and private cloud environments are connected through networks such as APIs, Virtual Private Networks(VPNs), and Multi-Protocol Label Switching(MPLS) to operate as a single system. This is due to a strategy to balance security and scalability by maintaining on-premises sensitive data and transferring non-core tasks to the cloud. However, this structural heterogeneity is a major factor in hindering security integration.

Since each public cloud service provider uses different API structures and security policy frameworks, it is difficult to consistently apply a single policy, resulting in problems such as duplicate definitions of security policies, synchronization delays, and omissions of exceptions. In particular, the delay in policy evaluation acts as a bottleneck for real-time security detection and response, which can directly lead to a decline in service quality and failure to respond to threats. Existing security systems have relied on a firewall-centered perimeter-based model, but in a hybrid cloud environment, network boundaries do not physically exist or are logically abstracted, making this traditional approach ineffective. Moreover, centralized policy evaluation methods cause real-time degradation due to evaluation delay and increased network round-trip time for geographically distributed user requests or high-frequency API traffic.

Recently, intelligent security research based on machine learning or AI has been actively conducted[14], but most of them do not sufficiently consider the policy application delay problem in hybrid cloud environments, and as a result, they reveal practical operational limitations in real-time performance. Therefore, the security policy delay problem in hybrid-cloud environments should be considered as a core task for service continuity and threat response. For this, a redesign of the policy evaluation flow is required, and we propose a new structure that combines an edge-based fast judgment and caching technique.

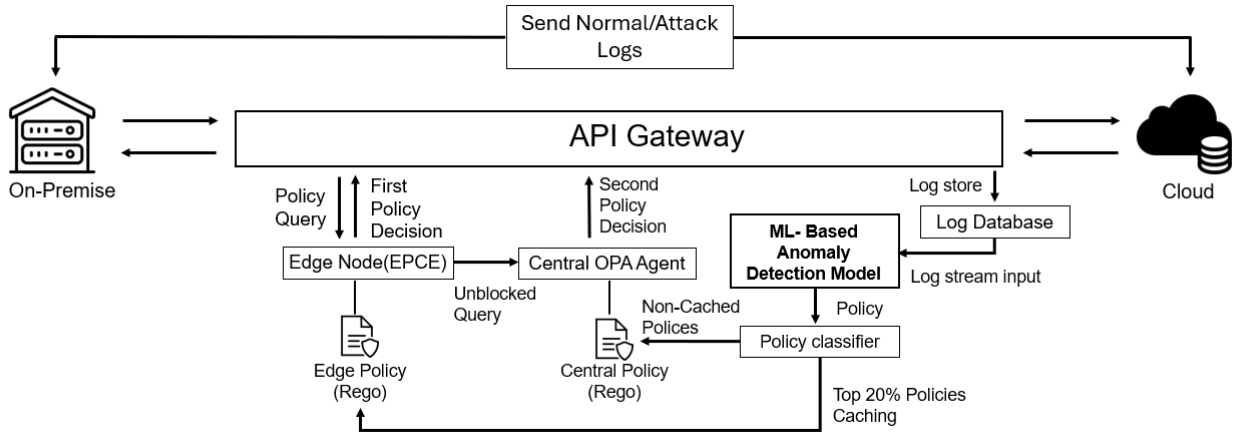

Figure 1. Proposed Model Using Edge Policy Cache Evaluation(EPCE)

III. Proposed Edge-Based Policy Caching Model

This chapter hierarchically decomposes the proposed security architecture for implementing real-time API threat detection and blocking within a hybrid cloud environment, detailing the roles of each layer, the technical implementation method, and the overall system flow in an integrated structure. The architecture forms a closed automated security system comprising machine learning-based anomaly detection, API flow aggregation, and an iterative learning mechanism; each layer is designed to maximize detection accuracy, response speed, and the efficiency of policy propagation throughout the system.

A. Traffic Gateway Layer

The traffic gateway layer is the layer located at the front end of the proposed architecture and serves as the secure entry point for all external requests to reach the system for the first time. This layer is configured based on the open-source API Gateway, Kong Gateway, and performs the function of controlling and recording all API requests flowing into cloud-based microservices or on-premises internal resources in a centralized manner.

When receiving an API request, the Kong Gateway automatically extracts key metadata from the request, such as URI, method, protocol, and DNS domain name, and converts them into JSON-style structured log and stores the log in the database. This log is then used as input data for the machine learning based anomaly detection modules and acts as a basis for policy enforcement decisions.

The Gateway functions as more than a simple router; it directly interacts with the policy evaluation layer to determine whether a request should be accepted. For each request, the gateway sends a REST-API query to the edge node or Central Open Policy Agent[22], and accordingly to the permit or deny decision, forwards or blocks the request. This structure simultaneously provides security visibility and control by collecting distributed API paths in a hybrid cloud environment at a single point and enabling tracking of all request flows.

B. Policy Generation Layer

The policy generation layer analyzes API log data stored in the database and automatically generates security policies using the machine learning model. Log data is parsed by features such as URI, DNS domain, and protocol, and converted into a character string vector that can be learned through special character removal and tokenization. After that, each log is vectorized and used to train the ML-based anomaly Detection Model.

The trained policy classifier calculates the frequency of appearance of each feature, and in this study, the top 20% feature tokens are selected and inserted into the edge node, and the rest are moved to the central OPA. This process is automatically performed for each experimental round, and when a new attack pattern that was not detected in the previous round occurs, the policy that reflects that feature is immediately updated. This allows the policy to be gradually refined so that a much more flexible response to new types of attacks is possible compared to prior static signature-based security systems.

C. Policy Evaluation Layer

The policy evaluation layer is a key layer that determines the suitability of API requests in real time, according to the security policy, and whether to allow requests. This layer is configured based on the OPA, and determines whether the request URI corresponds to a predefined blocking condition through a declarative policy language, Rego.

Policies are prioritized and stored in two separate files:

1) Edge Policy (Edge Node)

It contains only high-frequency patterns extracted from the machine learning detection module and is distributed to the edge instance of the OPA in a lightweight form. Edge OPA makes an immediate decision on a request to make a quick response.

2) Central Policy (Central OPA)

It is a comprehensive policy file including the entire pattern, performing secondary discrimination on complex requests that are not detected at the edge node. This requires relatively high processing time and resources, but the detection range is wider.

The Edge Policy Cache Evaluation(EPCE) first evaluates whether to permit/deny the request based on the edge policy for all requests. Its main function is to minimize the delay between request reception and decision making by running lightweight policy checks on incoming requests. The EPCE node loads a minimum subset of high-frequency pattern-matching rules that are derived from the policy classifier.

When an API request is received, the gateway first sends a query to the EPCE node. If a match is found within the cached edge policy set, the request is immediately blocked. Otherwise, the request is escalated to a more comprehensive central OPA. The central OPA performs more precise policy determination, and this hierarchical structure significantly reduces response time to known threats while maintaining detailed inspection capabilities.

D. Inter-Layer Linkage and Security Implementations

The three layers are not simply vertically listed independent layers but constitute a dynamic security loop that forms a circular structure for real-time detection, policy generation, and policy evaluation. This structure provides the following security advantages.

1) Real-Time Adaptability and Iterative Learning

When a new attack appears, the policy is automatically updated through log analysis and is immediately reflected in the blocking policy. And through this iterative learning, policies become increasingly precise, reducing the likelihood of omissions and more flexible to new attacks.

2) Separation of Detection and Blocking Efficiency

EPCE provides rapid judgments while central OPA offers comprehensive detection, achieving a balance between performance and accuracy.

3) System Scalability

Each layer is configured independently, making it easy to scale multiple gateways, multiple OPA instances, or the clouds.

This structure solves the rigidity and detection delay of traditional security systems that rely solely on static rules and provides a consistent security application and rapid response in dynamic and heterogeneous environments like hybrid clouds.

IV. Experimental Setup

In this chapter, we analyze how effectively the proposed security architecture can respond to various types of attacks based on quantitative indicators, and in particular, we aim to prove the effectiveness of the edge-based policy caching structure through repeated learning experiments.

A. Experimental Environment and Dataset Configuration

A test bed implements Kong Gateway as a traffic gateway, OPA as a policy evaluation layer, and Elasticsearch as a log storage. Used logs are based on the Behaviour-Centric Cybersecurity Center(BCCC) dataset, which is a form in which packet-based collected logs are reconstructed into API request units. Each request is labeled as malware, phishing, spam, or normal request.

In 889993 collected logs, 50,000 logs are randomly sampled for each round, with 10,000 attack logs and 40,000 normal logs. The experiment is repeated in 10 rounds in total, and new logs are added for each round to perform cumulative learning. This iterative structure aims to overcome the rigidity of updating static rule-based systems and increase flexibility for new types of attacks.

B. Policy Generation and Caching Mechanism

Each experimental round consists of three stages. In the first stage, attack logs and normal logs are sampled at a certain ratio and separated into training data and test data. Out of the total 50,000 logs, 40,000 normal logs and 10,000 attack logs were composed, and 80% of them are used for learning and the remaining 20% for evaluation. An average of approximately 8,731 of the total experimental logs were used as test logs for each round, and of these, an average of 6,985 normal logs and an average of 1,746 attack logs were composed. The collected logs are accumulated and used as learning data as the rounds repeat.

The second stage is feature extraction and model learning. Tokens are extracted from each log based on strings such as URI, DNS domain name, protocol, and vectorized using CountVectorizer. At this time, meaningless tokens (e.g., length 2 or less, number/hex value) are filtered out, and the result of the vectorizer is composed of a sparse vector of about 300 dimensions. Based on this vector, a Random Forest Classifier is learned, and the importance of each feature is calculated to extract the top 300 meaningful tokens.

The third stage is policy generation and caching. The extracted top tokens are converted into regular expressions and written in the Rego format. The generated policy is distributed to the EPCE node for the top 20% risks, and the rest are assigned to Central OPA. Afterwards, this policy is applied to the traffic gateway in real time, and each API request is evaluated in the order of EPCE -> Central OPA. Where each request was blocked is recorded in a separate field and used to analyze the results after the experiment ends. The ratio of attack logs blocked in EPCE or Central OPA is calculated. In addition, the average time taken for policy

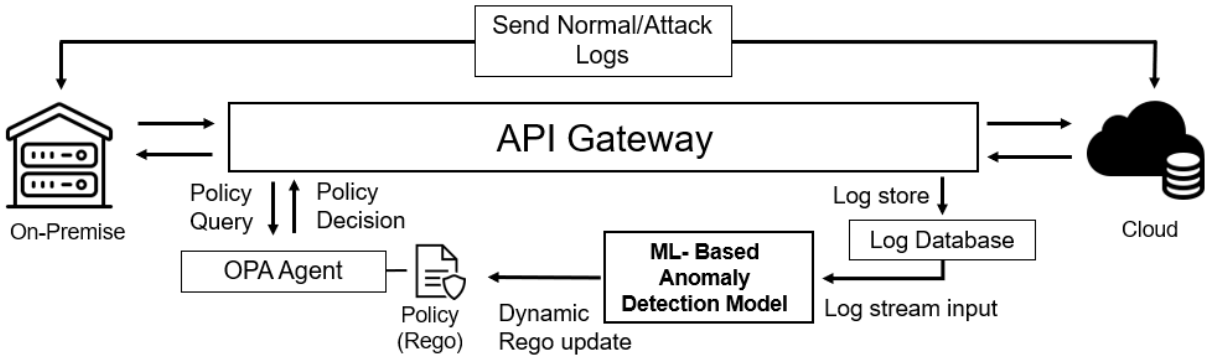

Figure 2. OPA-Only Baseline Model Used For Comparison

evaluation in each layer is recorded to measure response time performance.

V. Experimental Result

Following a total of 10 experiments, the proposed edge-based policy caching structure showed the following performance compared to the OPA-only baseline model.

TABLE I. Performance Comparison to OPA Only Baseline Methods

Features	Model Type	
	Proposed Model	***OPA Only***
Average Block Time(s)	**25.76**	32.23
Std. dev of Block Time	0.55	0.40
Average Block Rate(%)	**89.75**	85.91
Std. dev of Block Rate	2.24	2.09

The proposed model showed a clear performance advantage over the single evaluation method(hereinafter referred to as OPA Only) through a total of 10 repeated rounds. As shown in Table I, the average blocking time was 25.76 seconds on the proposed model, which is approximately 20.07% less than the total blocking time of 32.23 seconds for the OPA-only model. This result is not only a simple reduction in delay, but also quantitative evidence that the bottleneck in the policy evaluation path throughout the system was removed. Another important indicator for evaluating policy application performance is the blocking success rate for attack logs. The average blocking rate for the proposed model was 89.75% while the OPA-Only remained at 85.91%. This difference of 3.84% shows that the actual security detection performance was also improved in the proposed model.

TABLE II. Iterative Security Evaluation Results: Proposed Model versus OPA Only Baseline

Run	Block Rate(%)		Total Num of Attack Logs		Blocked Num of Attack Logs by Position		
	Proposed Model	***OPA Only***	***Proposed Model***	***OPA Only***	***Proposed Model***		***OPA Only***
					EPCE	***Central OPA***	
1	84.53	82.93	1739	1757	1226	244	1457
2	88.81	85.07	1743	1761	1285	263	1498
3	92.22	84.84	1761	1755	1352	272	1489
4	91.59	84.97	1725	1743	1277	303	1481
5	89.67	89.13	1772	1738	1252	337	1549
6	90.51	85.46	1729	1768	1257	308	1511
7	93.22	89.05	1754	1753	1291	344	1561
8	91.27	88	1753	1733	1280	320	1525
9	87.96	83.53	1744	1730	1264	270	1445
10	87.74	86.12	1745	1758	1241	290	1514
Avg	**89.75**	85.91	1746	1749	1272	295	1503

In Table II, which represents more specific figures, the edge node(EPCE) blocked 81.2% of the attack logs. The remaining 18.8% was processed by the Central OPA, which indicates that the Central OPA functions as a complementary mechanism for high-risk or atypical traffic that the edge did not detect.

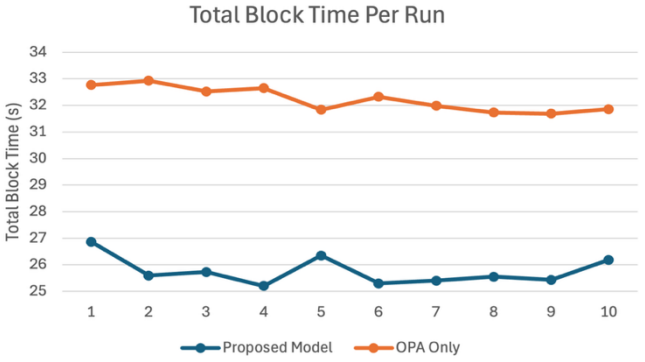

Figure 3. Total Block Time Per Run

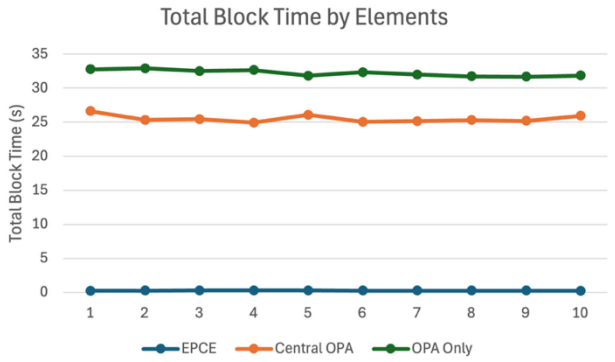

Figure 4. Total Block Time By Elements

Fig. 3 is a graph that shows the change in total blocking time for each repeated round, showing that the proposed model consistently maintains a lower blocking time than the OPA-only model in all rounds. This proves that the proposed model is not only superior in performance in certain situations but also maintains stable performance through repeated rounds.

Fig. 4 compares the average values of the response times collected through the experiment, showing the substantial delay improvement effect of the proposed model. In the case of the proposed model, the EPCE is executing blocks on attack logs very quickly. In detail, the average block time for EPCE was 0.26 seconds, Central OPA 25.50 seconds, and OPA Only model 32.23 seconds.

VI. Conclusion

In this study, we proposed an edge-based policy caching architecture for a hybrid cloud, including machine learning-based anomaly detection and automatic policy generation and distribution structure as a way to simultaneously resolve real-time policy application delay and detection accuracy in hybrid-cloud environments. The architecture consists of a traffic gateway, a policy evaluation layer, and an iterative learning-based policy generation layer, which comprehensively satisfies three key requirements: real-time detection and response, flexibility for new types of attacks, and system scalability.

The experiment was conducted 10 times using the BCCC dataset and showed a significant performance improvement over the existing single OPA-based structure, such as an average block rate of 89.75% and a 20.07% reduction in total block time for attack logs. In particular, 81.2% of all detections were blocked by the edge cache policy,

demonstrating the efficiency and practical applicability of the proposed model. This model is significant in that it provides a practical solution that can realize a dynamic and adaptive security policy system beyond the limitations of a fixed-rule-based system.

VII. Future Works

Future research can proceed in directions that further amplify the latent potential of the present framework. First, instead of selecting cache candidates solely by request frequency, a dynamic, risk-aware policy orchestration could be devised—one that optimizes caching strategies autonomously through a reinforcement-learning mechanism that weighs real-time threat likelihood, impact severity, and resource sensitivity. Building on the Random Forest's speed and interpretability, a hierarchical ensemble that blends LightGBM-Tabular, Temporal GNNs, and Transformer-based models could be introduced to capture both temporal correlations and polymorphic attack patterns, thereby boosting detection sensitivity and adaptability in tandem. Finally, by deploying a multi-region, distributed hybrid-cloud testbed and evaluating policy-version synchronization, cache consistency, and failover times under chaos-engineering scenarios, the practical deployability of the proposed EPCE architecture could be demonstrated with greater conviction. Collectively, these research avenues are expected to catalyze the evolution of policy caching into an intelligent, autonomous, and massively distributed security paradigm.

Acknowledgment

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation(IITP)-Innovative Human Resource Development for Local Intellectualization program grant funded by the Korea government(MSIT)(IITP-2025-RS-2022-00156287, 34%). This work was supported by Korea Institute of Planning and Evaluation for Technology in Food, Agriculture and Forestry(IPET) through the Agriculture and Food Convergence Technologies Program for Research Manpower development, funded by Ministry of Agriculture, Food and Rural Affairs(MAFRA)(project no. RS-2024-00397026, 33%). This research was conducted with the support of the Korea Internet & Security Agency (KISA) - Information Security Specialized University Support Project (33%).

References

[1] Z. Tang, "A Preliminary Study on Data Security Technology in Big Data Cloud Computing Environment," 2020 International Conference on Big Data & Artificial Intelligence & Software Engineering (ICBASE), Bangkok, Thailand, 2020, pp. 27-30.

[2] E. Collins, "Intersection of the Cloud and Big Data," in IEEE Cloud Computing, vol. 1, no. 1, pp. 84-85.

[3] S. Thakur and L. Verma, "Analysis of Big Data in Cloud Computing Technologies," 2024 11th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida, India, 2024, pp. 1-4.

[4] M. Gaianu, "On Premise Data Center vs CLOUD," 2023 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 2023, pp. 1068-1071.

[5] A. Balobaid and D. Debnath, "An Effective Approach to Cloud Migration for Small and Medium Enterprises (SMEs)," 2020 IEEE International Conference on Smart Cloud (SmartCloud), Washington, DC, USA, 2020, pp. 7-12

[6] A. Leff and O. Odukoya, "The Transformative Impact of Cloud Computing on Small and Medium-sized Enterprises (SMEs): A Comprehensive Analysis," 2024 International Conference on Smart Applications, Communications and Networking (SmartNets), Harrisonburg, VA, USA, 2024, pp. 1-5

[7] J. T. Rayfield, "Integrator: An Architecture for an Integrated Cloud/On-Premise Data-Service," 2015 IEEE International Conference on Web Services, New York, NY, USA, 2015, pp. 98-104.

[8] D. Sitaram et al., "Orchestration Based Hybrid or Multi Clouds and Interoperability Standardization," 2018 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM), Bangalore, India, 2018, pp. 67-71

[9] R. C. Pushpaleela, S. Sankar, K. Viswanathan and S. A. Kumar, "Application Modernization Strategies for AWS Cloud," 2022 1st International Conference on Computational Science and Technology (ICCST), CHENNAI, India, 2022, pp. 108-110

[10] A. Verma, D. Malla, A. K. Choudhary and V. Arora, "A Detailed Study of Azure Platform & Its Cognitive Services," 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), Faridabad, India, 2019, pp. 129-134

[11] R. Gohil and H. Patel, "Comparative Analysis of Cloud Platform: Amazon Web Service, Microsoft Azure, And Google Cloud Provider: A Review," 2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT), Kamand, India, 2024, pp. 1-5

[12] S. B. Mallisetty, G. A. Tripuramallu, K. Kamada, P. Devineni, S. Kavitha and A. V. P. Krishna, "A Review on Cloud Security and Its Challenges," 2023 International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT), Bengaluru, India, 2023, pp. 798-804.

[13] G. Raktate, K. Shelar, P. Parjane, S. Pangavhane, S. More and S. R. Deshmukh, "A Survey on Security Issues and Challenges in Cloud Computing," 2024 International Conference on Decision Aid Sciences and Applications (DASA), Manama, Bahrain, 2024, pp. 1-5.

[14] D. K. Seth, K. K. Ratra and A. P. Sundareswaran, "AI and Generative AI-Driven Automation for Multi-Cloud and Hybrid Cloud Architectures: Enhancing Security, Performance, and Operational Efficiency," 2025 IEEE 15th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 2025, pp. 00784-00793.

[15] C. Anjani, R. M. Balajee, G. Divya, Y. S. Sree, K. Padmanabham and S. S. Srithar, "Evolving Threats and AI Solutions for Modern Hybrid Cloud Architectures," 2023 2nd International Conference on Automation, Computing and Renewable Systems (ICACRS), Pudukkottai, India, 2023, pp. 478-484.

[16] S. J, K. Kanagasabapathi, K. Mahajan, S. Ahamad, E. Soumya and S. Barthwal, "AI-Enhanced Multi-Cloud Security Management: Ensuring Robust Cybersecurity in Hybrid Cloud Environments," 2023 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES), Chennai, India, 2023, pp. 1-6.

[17] Y. Mansouri, V. Prokhorenko, and M. A. Babar, "An Automated Implementation of Hybrid Cloud for Performance Evaluation of Distributed Databases," J.Netw. Comput. Appl., vol. 167, p. 102740, Oct. 2020

[18] I A. Mishra, P. Sarat and R. Afza, "A factual study on hybrid multi cloud cyber security threats and proposed methodologies to enable cyber resilience," 2024 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), Bangalore, India, 2024, pp. 1-6.

[19] Cadet, Emmanuel & Osundare, Olajide & Ekpobimi, Harrison & Samira, Zein & Weldegeorgise, Yodit. (2024). Comprehensive Framework for Securing Financial Transactions through API Integration in Banking Systems. 20. 662-672.

[20] M. L. Ali, S. Ismat, K. Thakur, A. Kamruzzaman, Z. Lue and H. N. Thakur, "Network Packet Sniffing and Defense," 2023 IEEE 13th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 2023, pp. 0499-0503

[21] KZ. Li, Q. Lu, L. Zhu, X. Xu, Y. Liu and W. Zhang, "An Empirical Study of Cloud API Issues," in IEEE Cloud Computing, vol. 5, no. 2, pp. 58-72.

[22] A. Paul, R. Manoj and U. S, "Amazon Web Services Cloud Compliance Automation with Open Policy Agent," 2024 International Conference on Expert Clouds and Applications (ICOECA), Bengaluru, India, 2024, pp. 313-317