Performance Analysis of a Container Based Security Approach in 5G Networks

Musab M. W. MohamedAli‡ Alessandro Carrega*‡ Roberto Bruschi*‡ Ramin Rabbani‡

* Department of Electrical, Electronic and Telecommunications Eng., and Naval Architecture (DITEN)
University of Genoa (UniGe), Italy {*name*}.{*surname*}@unige.it
‡ National Laboratory of Smart and Secure Networks (SN2) of the
National Inter-university Consortium for Telecommunications (CNIT), Genoa, Italy {*name*}.{*surname*}@cnit.it

***Abstract*—The cloud-native and virtualized design of 5G networks introduces new security challenges that traditional perimeter-based solutions cannot adequately address. Container-native runtime security tools provide real-time visibility and policy enforcement for containerized network functions. This paper evaluates the performance of Falco and NeuVector for securing 5G core network functions in a cloud-native testbed.In our deployment, both tools operate as node-level agents (DaemonSet-based monitoring components). The impact of each tool on system performance is evaluated separately by measuring CPU usage, memory consumption, and scalability under simulated security event workloads. Experimental results indicate that both solutions introduce moderate CPU overhead during event processing while maintaining stable memory utilization and preserving the functionality of core network services. Falco demonstrates a lower resource footprint, making it suitable for resource-constrained deployments. In contrast, NeuVector provides broader security capabilities, including network policy enforcement and vulnerability scanning, but at the cost of higher resource consumption. These findings highlight the trade-offs between lightweight and comprehensive container security approaches for practical 5G deployments.**

Keywords-5G; Node-Agent; NFV; falco; neuvector; security.

I. Introduction

As Fifth-Generation (5G) networks evolve toward highly virtualized and software-defined architectures, ensuring robust security mechanisms has become a critical requirement. The adoption of cloud-native technologies and Network Function Virtualization (NFV) enables flexibility and scalability but also expands the attack surface of core network functions [1]. Traditional perimeter-based security approaches are incapable of addressing these dynamic and distributed environments, encouraging the need for runtime security mechanisms that can operate natively within containerized infrastructures [1]. Container-native security solutions represent a promising approach for protecting 5G network functions without compromising performance or scalability [2]. These solutions integrate directly with container orchestration platforms and deploy dedicated monitoring agents capable of real-time threat detection, policy enforcement, and behavioral analysis. By decoupling security logic from the primary Virtual Network Function (VNF), such tools enable modular and scalable protection aligned with cloud-native design principles. Falco [3] and NeuVector [4] represent two complementary approaches to container security. Falco is a lightweight runtime security tool that monitors kernel system calls using extended Berkeley Packet Filter (eBPF) probes and evaluates them against predefined rule sets to detect suspicious container behavior such as abnormal process execution, unexpected file access, or unauthorized Kubernetes API interactions. In contrast, NeuVector provides a broader container security platform that combines runtime process monitoring with network traffic inspection, vulnerability scanning, and policy enforcement mechanisms. While Falco focuses primarily on detecting anomalous behavior and generating alerts, NeuVector supports active response capabilities, including policy-based traffic blocking and container isolation. Despite the growing adoption of container security tools in cloud environments, their performance impact and suitability for 5G core networks remain insufficiently explored. In particular, understanding the trade-offs between runtime monitoring and full-stack security platforms is essential for practical 5G deployments.

This paper addresses this gap by presenting a comparative performance evaluation of Falco and NeuVector in a cloud-native 5G testbed. We analyze their effectiveness in detecting security events and assess their impact on system resources, including CPU usage, memory consumption, and scalability under varying workloads. The results provide practical insights into the trade-offs between lightweight and comprehensive container security approaches, offering guidance for selecting suitable runtime security solutions for containerized 5G network functions.

The remainder of the paper is organized as follows: In Section II, we review related work and identify security challenges in 5G NFV environments. Section III presents the experimental setup and methodology used for performance evaluation. In Section IV, we provide a detailed analysis of the results, including performance and security effectiveness comparisons between Falco and NeuVector. Section V offers a comparative analysis of the two security solutions in the context of 5G Core networks. Finally, Section VI concludes the paper and discusses potential directions for future research.

II. Related Work and Security Challenges in 5G NFVs Environments

Several studies have highlighted that the adoption of NFV in 5G networks significantly improves flexibility and scalability, while simultaneously introducing new security concerns [5]. Existing literature identifies multiple challenges that arise from the software-based and dynamic nature of virtualized network functions, motivating the need for runtime and container-native security mechanisms. First, the increased attack surface

introduced by virtualization means that each NFV represents a potential entry point for attackers. The propagation of software-based endpoints complicates security enforcement in large-scale deployments [6]. Moreover, vulnerabilities stemming from misconfigurations or unpatched software further expose NFVs to exploitation [7].

Second, the dynamic lifecycle of VNFs challenges traditional static security mechanisms. The ability to rapidly instantiate, migrate, and terminate NFVs requires adaptive security solutions capable of responding to real-time changes in network topology and configuration[8].

Third, multi-tenancy introduces additional risks when multiple tenants share the same physical and virtual infrastructure. Unsuitable isolation may allow an attack originating from one tenant to propagate across others[9]. These risks are often exacerbated by configuration errors in resource allocation and access control policies[10].

Furthermore, the creation of complex service chains, a defining characteristic of NFVs, increases the difficulty of security management. Service function chaining interconnects multiple VNFs, meaning that the compromise of a single component can impact the entire end-to-end service [11]. Prior work explores secure orchestration and automated verification techniques to mitigate such threats [12].

Finally, the NFV orchestrator represents a critical and high-value target. As the central control entity responsible for managing VNFs, an orchestrator compromise can lead to large-scale service disruption or loss of network control[13]. Consequently, strong authentication, access control, and continuous monitoring mechanisms are essential to protect this component [7]. These challenges identified in prior work motivate the evaluation of container-native runtime security solutions, such as Falco and NeuVector, which are examined in the following sections.

III. Testbed and Experimental Methodology

This section describes the experimental testbed, deployment configuration of the security tools, and the methodology used to evaluate their performance and detection effectiveness.

A. 5G Cloud-Native Testbed Description

We evaluate the performance impact of container-native runtime security on a cloud-native 5G/6G testbed by comparing Falco and NeuVector in identical conditions. The evaluation focuses on (i) computational overhead, (ii) memory footprint, (iii) system load, and (iv) runtime security monitoring effectiveness, both under normal operation and under simulated attack activity. The testbed consists of a three-node Kubernetes cluster deployed as virtual machines: one control-plane node (8 vCPUs, 8 GB RAM) and two worker nodes (6 vCPUs, 8 GB and 16 GB RAM). All nodes run Ubuntu 24.04.3 LTS (kernel 6.8.0-78-generic) with containerd 1.7.27 as the container runtime.

B. Security Tool Deployment Model

In this testbed, Falco and NeuVector were deployed using node-level monitoring components implemented as Kubernetes DaemonSets, allowing each node to run a dedicated security monitoring agent. The two platforms differ in the runtime signals they analyze. Falco operates primarily at the kernel level by monitoring container system calls through rule-based detection policies, while NeuVector analyzes multiple signals including container processes, inter-container network traffic, and policy compliance events.

C. Measurement Methodology

We collect CPU and memory utilization per node and relevant pods, system load averages (1/5/15 min), and CPU idle percentage. Measurements are reported for three stages: *baseline* (no runtime security), *post-deployment* (Falco or NeuVector enabled), and *attack simulation* (event stream generation).

To account for variability in Kubernetes scheduling and resource allocation, a double-baseline measurement strategy was adopted, where baseline resource utilization was recorded both before NeuVector installation and after NeuVector removal but prior to Falco deployment. Observed baseline values after NeuVector removal were consistent with the initial baseline measurements, indicating minimal residual configuration impact, following repeated-measurement guidance for dynamic containerized environments [14].

D. Event Generator Configuration

To evaluate the detection capabilities of Falco and NeuVector, an event generator was used to simulate attack scenarios and security-relevant activities within the Kubernetes environment. The generator produced predefined attack scenarios corresponding to container runtime and network behaviors that trigger Falco rules and NeuVector security policies. Each scenario represents a *unique simulated attack*, defined as a deterministic invocation of a specific attack rule within the generator configuration. Repeated executions of the same scenario were treated as independent attack instances. During the experiment, 400 attack scenarios were generated and logged with timestamps and identifiers. Falco detections were mapped directly to predefined Falco rules (e.g., *Contact K8S API Server From Container*), while NeuVector detections were triggered through runtime behavioral monitoring and network policy enforcement mechanisms. Ground-truth labels were established by correlating event generator logs with Falco and NeuVector alerts based on timestamp, node identifier, and event type. This correlation enabled the calculation of detection metrics including True Positives (TP), False Positives (FP), and False Negatives (FN). To ensure reproducibility, the experiments were conducted on a three-node Kubernetes cluster (Ubuntu 24.04.3, containerd 1.7.27), with Falco and NeuVector deployed via Helm charts in DaemonSet mode. The event generator was executed in loop mode for a 10-minute evaluation window, and logs were collected using `kubectl logs` for post-processing and metric computation.

IV. Experimental Results

This section presents the experimental results obtained from the testbed, including baseline measurements and the

performance impact of NeuVector and Falco during monitoring and attack simulation scenarios.

A. Baseline Measurements

Before deploying runtime security, baseline resource usage was recorded across cluster nodes. CPU utilization was low on all nodes (control plane: 2% / 204 mCPU[1]; Worker Node 1: 6% / 414 mCPU; Worker Node 2: 7% / 440 mCPU), with RAM usage of 1913 MiB (24%), 1824 MiB (23%), and 3124 MiB (19%) respectively. Load averages remained below 1 on all nodes, indicating sufficient capacity prior to enabling security monitoring. The low baseline utilization (2–7% CPU) reflects the controlled experimental setup, where sufficient resources were provisioned to isolate the overhead of runtime security monitoring rather than emulate fully loaded production deployments.

B. NeuVector Performance Under Monitoring and Attack

After deploying NeuVector, CPU utilization increased across nodes: Control Plane Node from 2% (204 mCPU) to 3% (314 mCPU), Worker Node 1 from 6% (414 mCPU) to 8% (535 mCPU), and Worker Node 2 from 7% (440 mCPU) to 11% (705 mCPU). Memory usage increased from 1913 MiB to 2816 MiB on the Control Plane Node, from 1824 MiB to 2532 MiB on Worker Node 1, and from 3124 MiB to 7559 MiB on Worker Node 2. Load averages showed moderate increases but remained within operational limits. These changes reflect the overhead of NeuVector's distributed monitoring and policy enforcement components.

Attack simulation under event-generation workloads, CPU usage peaked at 5% (399 mCPU) for the Control Plane Node, 11% (688 mCPU) for Worker Node 1, and 21% (1279 mCPU) for Worker Node 2, while memory usage increased to 3417 MiB (43%), 3168 MiB (40%), and 9507 MiB (59%) respectively. System load increased substantially, indicating a higher number of queued processes during high event rates. Table I summarizes system performance metrics under NeuVector monitoring during baseline, post-deployment, and attack simulation phases.

NeuVector's detection performance was evaluated using the event generator described in Section III-D. The evaluation focused on standard security effectiveness metrics, including true positive rate (TPR), false positive rate (FPR), false negative rate (FNR), and overall detection capability. During a 10-minute experiment, the event generator produced 400 unique simulated attack events. NeuVector detected and correctly handled 387 of these events, resulting in an overall detection rate of 96.7%. As reported in Table II, presents the security detection metrics obtained for NeuVector a TPR was 85.9% (344 events), while the FNR was 3.3% (13 missed events). In parallel, NeuVector generated alert logs corresponding to both malicious activities and benign events, yielding a FPR of 10.7% (43 events).

Detected events were classified into multiple response categories. The majority of events (85.9%) were categorized as *Deny Actions*, representing critical threats that were actively blocked, while 10.7% were classified as *Alert Actions* with warning severity. NeuVector demonstrated rapid response behavior, with detection and mitigation actions executed in under one second for all analyzed events.

[1]A millicore is a unit of CPU resource allocation in containerized environments, where 1000 m equals one full CPU core.

TABLE I. SYSTEM PERFORMANCE AT BASELINE, BEFORE, AND AFTER ATTACK SIMULATIONS UNDER NEUVECTOR PROTECTION.

Metric	Baseline	Post-Deployment	Attack Simulation
Control Plane Node			
CPU Util.	2% (204 mCPU)	3% (314 mCPU)	5% (399 mCPU)
Memory	1913M (24%)	2816M (35%)	3417M (43%)
1-min	0.09	0.26	0.45
5-min	0.15	0.28	0.69
15-min	0.24	0.32	1.68
Worker Node 1			
CPU Utilization	6% (414 mCPU)	8% (535 mCPU)	11% (688 mCPU)
Memory Usage	1824M (23%)	2532M (32%)	3168M (40%)
1-min	0.29	0.44	1.32
5-min	0.34	0.52	2.41
15-min	0.49	0.61	5.98
Worker Node 2			
CPU Utilization	7% (440 mCPU)	11% (705 mCPU)	21% (1279 mCPU)
Memory Usage	3124M (19%)	7559M (47%)	9507M (59%)
1-min	0.54	0.73	2.20
5-min	0.69	0.82	4.56
15-min	0.77	0.88	11.89
NeuVector Components			
Controller pods	–	154m	320m
Manager pod	–	8m	19m
Enforcer pods	–	234m	602m
Scanner pods	–	3m	270m

TABLE II. NEUVECTOR SECURITY PERFORMANCE METRICS.

Metric	Value	Event Calculation
False Positive Rate	10.7%	43/400
False Negative Rate	3.3%	13/400
True Positive Rate	85.9%	344/400
Overall Detection Rate	96.7%	387/400
Response Time	$<$ 1 second	Real-time

C. Falco Performance Under Monitoring and Attack

To evaluate Falco under comparable conditions, NeuVector was removed and a new baseline was recorded prior to Falco deployment. Baseline CPU usage was 2% (215 mCPU) on the Control Plane Node, 5% (311 mCPU) on Worker Node 1, and 5% (324 mCPU) on Worker Node 2, with RAM usage of 2270 MiB (28%), 2221 MiB (28%), and 2721 MiB (17%) respectively. Load averages remained below 1 across the cluster.

Post-deployment after enabling Falco, control-plane CPU rose to 4% (338 mCPU), while worker nodes increased to 6% (371 mCPU) and 6% (372 mCPU). Memory usage increased slightly to 2338 MiB (29%) on the Control Plane Node, 2332 MiB (29%) on Worker Node 1, and 2830 MiB (17%) on Worker Node 2. Load averages increased moderately, particularly on Worker Node 2, reflecting the cost of continuous syscall monitoring.

Attack simulation under event-generation workloads, CPU peaked at 6% (438 mCPU) on the Control Plane Node, 8%

TABLE III. SYSTEM PERFORMANCE AT BASELINE, BEFORE, AND AFTER ATTACK SIMULATIONS UNDER FALCO PROTECTION.

Metric	Baseline	Post-Deployment	Attack Simulation
Control Plane Node			
CPU Util.	2% (215 mCPU)	4% (338 mCPU)	6% (438 mCPU)
Memory	2270M (28%)	2338M (29%)	2978M (37%)
1-min	0.20	0.46	0.62
5-min	0.21	0.59	0.89
15-min	0.25	0.69	1.08
Worker Node 1			
CPU Util.	5% (311 mCPU)	6% (371 mCPU)	8% (541 mCPU)
Memory	2221M (28%)	2332M (29%)	3086M (39%)
1-min	0.35	0.35	2.04
5-min	0.35	0.50	2.97
15-min	0.37	0.53	3.32
Worker Node 2			
CPU Util.	5% (324 mCPU)	6% (372 mCPU)	9% (582 mCPU)
Memory	2721M (17%)	2830M (17%)	3738M (41%)
1-min	0.55	0.64	1.33
5-min	0.74	0.97	1.61
15-min	0.80	1.60	2.58
Falco Components			
Falco pod	–	95m	180m
Falco exporter	–	12m	35m

(541 mCPU) on Worker Node 1, and 9% (582 mCPU) on Worker Node 2. Falco pod CPU increased from 26m to 45m on the Control Plane Node, from 40m to 68m on Worker Node 1, and from 47m to 89m on Worker Node 2. Memory peaked at 2978 MiB (37%) on the Control Plane Node, 3086 MiB (39%) on Worker Node 1, and 3738 MiB (41%) on Worker Node 2. System load increased markedly on worker nodes, indicating that sustained event streams amplify monitoring cost. Table III reports the corresponding system performance measurements under Falco protection.

Falco's detection performance was evaluated using the event generator described in Section III-D. During the 10-minute experiment, the event generator produced 400 simulated attack events representing the ground-truth malicious activities. Falco successfully detected 370 of these attacks and missed 30, resulting in a TPR of 92.5% and a FNR of 7.5%. However, Falco generated a total of 2,100 alerts during the same period, including alerts triggered by benign system activities. Among these alerts, 1,700 were classified as false positives, resulting in an operational FPR of approximately 82%, computed as $FPR/(TPR + FPR)$. A significant portion of these false positives originated from the Falco rule *"Contact K8S API Server From Container"*, which was frequently triggered by legitimate Flannel CNI communications with the Kubernetes API server. This observation highlights the sensitivity of rule-based runtime monitoring and the importance of rule tuning when deploying Falco in production environments.

Detected events were classified into multiple severity levels. Critical detections accounted for 11.7% (245 events), warning detections for 6.0% (125 events), and notice-level detections for 9.5% (200 events) representing legitimate activities. The remaining 71.4% (1,500 events) corresponded to false positives. For all detected events, Falco demonstrated rapid response behavior, with detection and alert generation occurring in under one second. Table IV summarizes the corresponding detection metrics for Falco.

TABLE IV. FALCO SECURITY PERFORMANCE METRICS.

Metric	Value	Event Calculation
False Positive Rate	82.1%	1,700/2,100
False Negative Rate	7.5%	30/400
True Positive Rate	92.5%	370/400
Overall Detection Rate	92.5%	370/400
Response Time	< 1 second	Real-time

Analysis of false positives revealed that the primary source of alert noise originated from Flannel CNI components communicating with the Kubernetes API server, triggering the *Contact K8S API Server From Container* rule. These results indicate that Falco provides effective runtime threat detection for containerized 5G core network functions while maintaining low latency and operational efficiency.

V. COMPARATIVE ANALYSIS & SECURITY EFFECTIVENESS COMPARISON

This section presents the experimental results obtained from the testbed, including baseline measurements and the performance impact of NeuVector and Falco during monitoring and attack simulation scenarios.

A. Resource Overhead Comparison

This section presents a comparative evaluation of the NeuVector and Falco security platforms in the context of a 5G Core Kubernetes environment, focusing on resource overhead, system load impact, and security effectiveness. The evaluation quantifies CPU utilization, memory consumption, and system load across all cluster nodes to assess the performance characteristics of each security solution. Both platforms maintained system stability during testing, but each demonstrated distinct performance profiles under load conditions.

Figures 1 and 2 show the resource overhead introduced by the security tools relative to the baseline system state under post-deployment conditions. NeuVector introduces a significantly higher resource overhead than Falco. Specifically, NeuVector components consume 416 mCPU and 5,380 MiB of memory across 10 components, while Falco requires only 113 mCPU cores and 319 MiB of memory across 3 components. NeuVector's resource usage is approximately 3.7 higher in CPU and substantially greater in memory compared to Falco, indicating that NeuVector's comprehensive security coverage comes at the cost of increased resource consumption.

System load analysis reveals differences in computational stress between the two platforms, particularly under attack simulation conditions. Under normal operation, NeuVector caused moderate increases in load, most notably on Worker Node 2, where the 15-min load increased from 0.49 to 0.61. Falco, in contrast, demonstrated a more significant load impact,

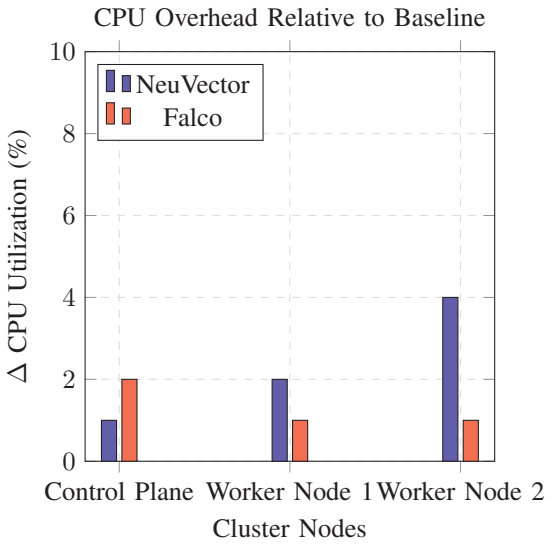

Figure 1. CPU overhead relative to baseline under post-deployment conditions (before attack simulation).

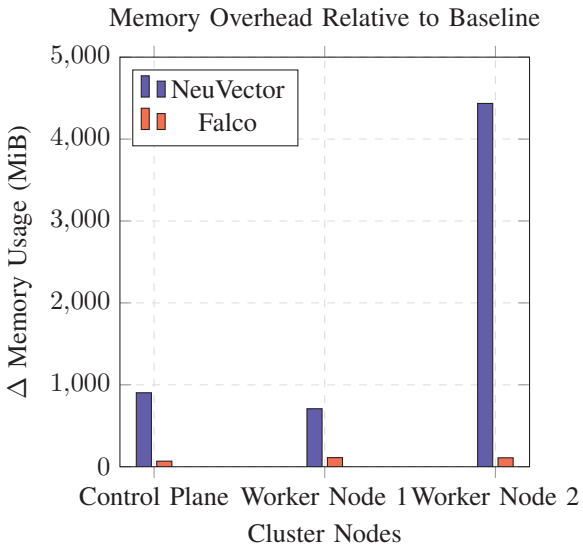

Figure 2. Memory overhead relative to baseline under post-deployment conditions (before attack simulation).

where the 15-min load increased from 0.80 to 1.60 on Worker Node 2.

During attack simulations, the load increases were more significant. NeuVector's system load surged, particularly on Worker Node 2, where the 15-min load rose from 0.88 to 11.89. In comparison, Falco exhibited a increase on Worker Node 1 from 0.53 to 3.32, showing more efficient load management. This indicates that while NeuVector's more comprehensive monitoring incurs significant overhead, Falco's lightweight architecture offers more efficient performance, especially in resource-constrained environments.

Both platforms maintained system stability during the event generator simulations, but with differing impacts on performance. NeuVector resulted in a peak to 21% CPU usage on Worker Node 2 during attack simulations, while Falco only incurred a peak to 9% CPU usage on Worker Node 2 during attack simulations. Memory consumption also differed significantly, with NeuVector consuming significantly more memory due to its broader security capabilities, while Falco demonstrated minimal memory overhead.

B. Scalability Behavior

As the number of simulated subscribers increases, the workload generated by the event generator produces a higher volume of container runtime activities and network events that must be analyzed by the security monitoring tools. Falco processes system call events using lightweight eBPF probes, meaning its monitoring overhead scales with the number of container operations occurring on each node. Consequently, its resource consumption increases approximately in proportion to workload activity. NeuVector, in contrast, performs both container behavior monitoring and network traffic inspection, which introduces additional processing requirements as event volume and traffic intensity grow. As a result, NeuVector generally exhibits higher CPU utilization under increased workloads, although memory consumption remains relatively stable due to its architecture. Overall, the experimental results suggest that both tools exhibit approximately linear growth in overhead with respect to workload intensity, with Falco maintaining a lighter monitoring footprint while NeuVector provides broader security coverage at the cost of higher resource usage.

C. Detection Effectiveness Comparison

Both platforms demonstrated strong threat detection capabilities under controlled testing. Falco achieved a 92.5% TPR, successfully detecting 370 out of 400 simulated attacks. NeuVector had a slightly higher overall detection rate of 96.7%, identifying 387 out of 400 events, but its TPR was 85.9%. While Falco's higher TPR (92.5%) indicates stronger precision in detecting actual threats, NeuVector excelled at broader detection coverage, capturing more security events in total.

The FPR for NeuVector was significantly lower (10.7%) compared to Falco's high FPR of 81%, largely driven by Flannel CNI API communications triggering the "Contact K8S API Server From Container" rule. These results suggest that while NeuVector provides a more precise and effective detection system, Falco's high FPR could hinder its practical deployment in production environments without significant rule customization.

Both tools demonstrated equivalent response times under 1 second, ensuring real-time security protection. However, NeuVector's lower FNR (3.3% vs 7.5%) indicates a more comprehensive coverage of potential threats. NeuVector's low FPR (10.7%) highlights its well-tuned detection rules, while Falco's high FPR (81%) represents a significant operational challenge, which could lead to alert fatigue in production environments. Despite Falco's high TPR, the large number of false positives compromises its usability without further tuning and customization. Additionally, the FNR for Falco

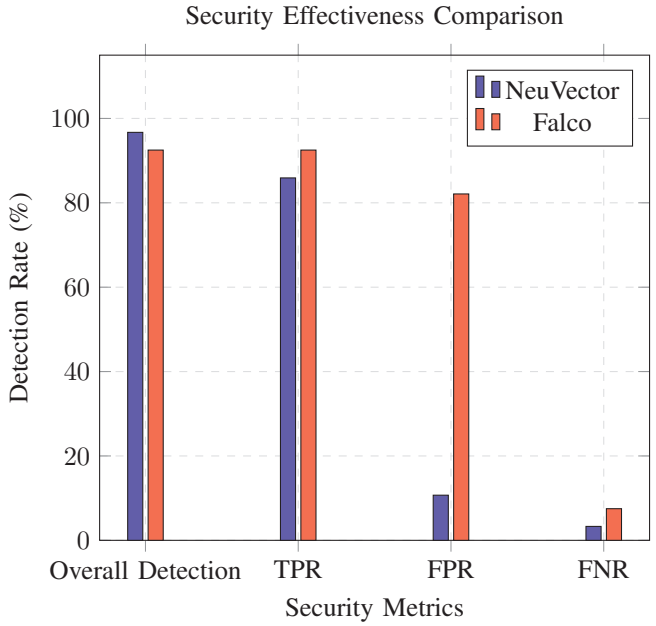

Figure 3. Security Effectiveness Comparison: NeuVector vs Falco.

(7.5%) is relatively higher, suggesting room for improvement in detecting container-specific threats, Figure 3 presents a comparative visualization of the detection effectiveness metrics obtained for NeuVector and Falco.

It is important to note that the reported detection metrics are derived from controlled attack scenarios generated by the event generator operating in loop mode. Consequently, the reported TPR and FPR reflect detection performance for the evaluated synthetic workload rather than the full spectrum of real-world threats. In operational 5G environments, attacks may involve multi-stage exploits, protocol-level manipulation, or previously unseen vulnerabilities that are not represented in the current evaluation scenario.

D. Practical Deployment Implications

Both NeuVector and Falco offer trade-offs between security effectiveness and system performance. NeuVector provides comprehensive security coverage, but with higher resource overhead, while Falco offers efficient runtime security with lower resource consumption but higher false positives.

In terms of 5G Core service resilience, both platforms maintained 99.9%+ service availability and had minimal impact on throughput and system performance. NeuVector introduced a moderate increase in CPU utilization, peaking at 21% CPU usage on Worker Node 2, while Falco showed a more moderate increase at 9% CPU usage on Worker Node 2. Both platforms demonstrated minimal impact on overall system performance, ensuring they meet the stringent requirements for ultra-reliable low-latency communication (URLLC).

In summary, NeuVector offers a broader detection coverage and more precise alerting with lower false negatives but at the expense of higher resource consumption. Falco, on the other hand, provides efficient runtime monitoring with minimal system impact but requires further optimization to address its high FPR. Both platforms meet essential 5G Core performance requirements, but Falco is more suited for resource-constrained environments, while NeuVector is ideal for comprehensive security coverage where performance overhead is less critical.

VI. Conclusion and Future Work

This paper provides a comprehensive evaluation of NeuVector and Falco as container-native security solutions within 5G Core cloud-native environments. The results highlight significant trade-offs between security coverage and performance overhead in cloud-native 5G deployments.

The evaluation revealed that NeuVector consumes 3.7 times more CPU and 16.9 times greater memory compared to Falco. Despite this, NeuVector offered superior security coverage, making it more suitable for production environments requiring comprehensive protection. On the other hand, Falco demonstrated greater resource efficiency, though its high FPR (82.1%) may limit its deployment in production environments without substantial rule customization. igure 4 illustrates the CPU utilization observed across the cluster nodes before and during security monitoring for both NeuVector and Falco.

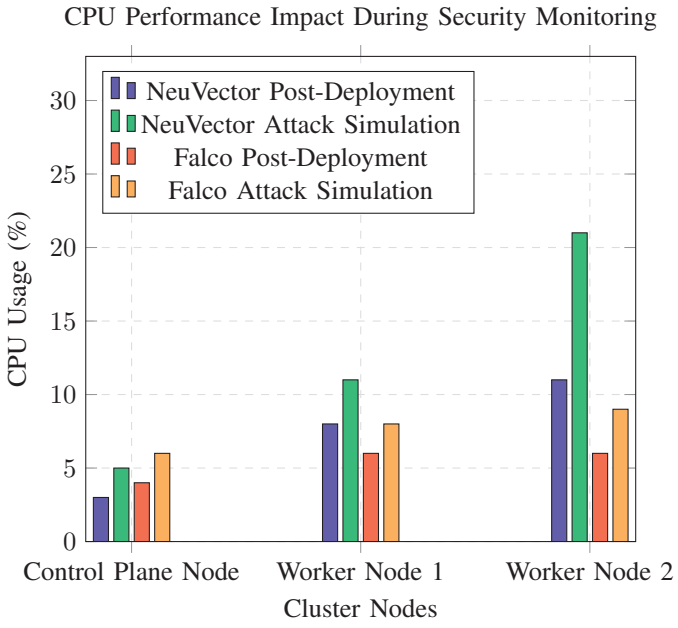

Figure 4. **CPU Performance Impact During Security monitoring: NeuVector vs Falco.**

Under the evaluated workload conditions, both security solutions preserved service continuity and maintained stable operation of the containerized network functions during deployment and attack simulations. These results indicate that container-native security monitoring can be integrated into 5G Core environments without introducing significant service disruption under the tested experimental conditions.

Future research can explore several promising pathways to further enhance the security and performance of container-native security tools in 5G Core networks. First, more realistic

5G-specific attack scenarios will be investigated, including Service-Based Architecture (SBA) exploitation, network slicing isolation bypass attempts, control-plane signaling manipulation, and edge/MEC-related threats. Second, multi-stage attack simulations, behavioral anomaly detection, and machine learning-based false-positive reduction will be explored to improve detection accuracy and operational efficiency, particularly for Falco. Third, the evaluation will be replicated under higher baseline utilization levels representative of production 5G deployments to better characterize the performance–security trade-offs in cost-optimized environments.

In conclusion, this research demonstrates that both NeuVector and Falco can provide effective runtime security monitoring for 5G Core environments under the evaluated workload conditions. While NeuVector offers broader security coverage, it comes with higher resource consumption. Falco provides efficient runtime monitoring with low resource overhead, though its high false-positive rate needs to be addressed before deployment in production environments. Both platforms meet the essential performance requirements of 5G Core services in the tested environment, and the choice between them depends on specific deployment needs, resource constraints, and security priorities.

ACKNOWLEDGMENT

This work has been partially supported by the Horizon European project MARE (grant agreement no. 101191436), by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU (NGEU), partnership on “MUR” (PE000000014 - program “SERICS”).

REFERENCES

[1] U. Kulkarni and S. Fahmy, “Securing the cloud-native 5g control plane”, in *MILCOM 2024 - 2024 IEEE Military Communications Conference (MILCOM)*, 2024. DOI: 10.1109/MILCOM61039.2024.10774032

[2] I. T. Aktolga, E. S. Kuru, Y. Sever, and P. Angin, “Ai-driven container security approaches for 5g and beyond: A survey”, *ITU Journal on Future and Evolving Technologies*, vol. 4, no. 2, 2023.

[3] Falco Security Project, *Falco: Cloud Native Runtime Security*, https://github.com/falcosecurity/falco, Accessed: October 2025.

[4] SUSE Rancher, *NeuVector*, Official documentation describing NeuVector’s features (firewall, process/file-system monitoring, vulnerability scanning) and architecture (Controller, Enforcer, Manager, Scanner).

[5] G. He, X. Liao, and C. Liu, “A security survey of nfv: From causes to practices”, in *2023 3rd International Conference on Consumer Electronics and Computer Engineering (ICCECE)*, 2023. DOI: 10.1109/ICCECE58074.2023.10135454

[6] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov, “A comprehensive survey on security in software-defined networking”, *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, 2020.

[7] K. H. Nguyen, N. H. Pham, and J. Shim, “Security challenges in software-defined networking-based 5g networks: A survey”, *IEEE Access*, vol. 9, 2021.

[8] A. Liaqat et al., “Dynamic security in network function virtualization: A survey on methods, challenges, and future directions”, *IEEE Access*, vol. 9, 2021.

[9] M. Ali et al., “Multi-tenancy security in 5g networks: Challenges and countermeasures”, *Journal of Network and Computer Applications*, vol. 173, p. 102 899, 2021.

[10] R. El Ghamri, M. Bouhorma, et al., “Security issues and solutions in multi-tenant environments for network function virtualization”, *Journal of Communications Software and Systems*, vol. 16, no. 3, 2020.

[11] P. Li et al., “Ensuring secure service chains in nfv: State-of-the-art and challenges”, *IEEE Network*, vol. 33, no. 6, 2019.

[12] S. Kim et al., “Secure service chaining in software-defined networking-based 5g environments: A survey”, *IEEE Transactions on Network and Service Management*, vol. 17, no. 3, 2020.

[13] F. Yang et al., *Orchestrator security in nfv/5g*, 2021.

[14] S. Acharekar, A. Goswami, and F. Nair, “Cloud-native performance testing: Strategies for scalability and reliability in modern applications”, *European Journal of Computer Science and Information Technology*, vol. 13, no. 8, 2025.