Securing the Cloud-Native 5G Control Plane

Umakant Kulkarni, Sonia Fahmy
Purdue University
e-mail: {ukulkarn,fahmy}@purdue.edu}

***Abstract*—The shift to cloud-native architectures in the 5G control plane introduces significant challenges in securing complex network functions deployed as microservices. Traditional security mechanisms, such as service mesh architectures and kernel-level solutions, often suffer from protocol limitations, high overhead, or incomplete inter-service encryption. Therefore, we present Zero Trust X Security Enforcement Microservice (ZTX-SEM), a protocol-agnostic zero-trust security solution designed specifically for cloud-native deployments. We propose a lightweight packet interception mechanism that operates across all protocols, a proactive authentication strategy that reduces latency and ensures continuous readiness, and an optimized secret key lookup that accelerates encrypted communication processing. Our experiments on a Kubernetes-deployed 5G control plane (core) demonstrate that ZTX-SEM outperforms existing solutions such as Istio, achieving 75% and 28% reductions in resource utilization and session setup times, respectively.**

I. Introduction

Operators are increasingly adopting a cloud-native approach to deploying 5G control plane network functions (NFs). Unlike traditional monolithic architectures, 5G NFs are now realized as microservices, typically deployed using containers [2]. This shift increases flexibility and scalability, allowing operators to deploy NFs closer to end-users, and leveraging public cloud infrastructure to reduce cost and latency, and streamline operations. However, this new paradigm also introduces security challenges, given that most NFs are delivered as black-box entities by software vendors, often without integrated security mechanisms.

Current security approaches, such as service mesh-based architectures and kernel-level solutions, attempt to address these challenges by leveraging sidecar transparent proxies or eBPF [1]. These approaches, however, often fall short due to lack of support for non-TCP traffic, high overhead, incomplete encryption of inter-service communication within the same node, or privilege escalation risks [18].

To address these limitations, we propose a comprehensive security solution specifically designed for cloud-native microservice-based architectures. We introduce Zero Trust X Security Enforcement Microservice (ZTX-SEM), which provides end-to-end encryption, mutual authentication, and low-overhead packet processing across all interfaces of the 5G control plane (core). ZTX-SEM is stateless, meaning it processes packets independently, thereby minimizing resource consumption and latency.

ZTX-SEM follows *zero trust* [21], a cybersecurity model that *eliminates implicit trust based on network location or asset ownership*. Zero trust implies verification of each request to network resources. In ZTX-SEM, we apply this approach to secure the cloud-native 5G control plane by ensuring that all communication is authenticated and encrypted. The novelty of our work lies in two aspects. First, ZTX-SEM is protocol-agnostic and preserves the original transport protocol by encrypting the payload directly, avoiding the need for secure tunnels and minimizing disruption to the 5G core. Second, ZTX-SEM enforces a zero-trust model by not trusting the underlying orchestrator or nodes. This makes it ideal for military 5G core deployments in hostile environments where the infrastructure itself may be compromised or untrusted.

The key contributions of this paper include the design and implementation of (1) a protocol-agnostic lightweight packet interception mechanism, (2) a proactive authentication to minimize latency and ensure continuous readiness, and (3) an optimized secret key lookup method for rapid encrypted communication processing. We validate ZTX-SEM on a cloud-native 5G core deployed on a Kubernetes platform and assess the overhead introduced in terms of both time and resource utilization. We benchmark ZTX-SEM against an Istio service mesh [6] and demonstrate that ZTX-SEM reduces resource utilization by 75%, with a 28% decrease in session setup times.

II. Threat Model and Requirements

We consider a cloud-native deployment of the 5G core. Our threat model assumes that an attacker can inject, modify, and sniff network traffic.

Signaling disruption and user impersonation: We consider attacks targeting signaling messages and 3GPP-TS-23.502 procedures between 5G core NFs as well as between NFs and the User Plane or Radio Access Network (RAN). 5G networks are known to be susceptible to such man-in-the-middle attacks on unencrypted connections, allowing attackers to intercept and modify critical messages, e.g., Attach Request/Response, Service Request, or Authentication Request/Response [22]. By exploiting intercepted or previously captured traffic, attackers can replay sessions and reuse authentication credentials. This allows them to impersonate legitimate users, initiate unauthorized state changes, and deny services [20].

Network function hijacking: The complexity of the 5G architecture creates opportunities for unauthorized access and configuration, where attackers exploit vulnerabilities to take control of critical 5G NFs [12]. Specifically, during service registration, a legitimate NF updates the Network Repository Function (NRF) with its details, but if this communication

This work has been supported in part by NSF grants 2226447 and 2326898.

is unencrypted, attackers can intercept and alter the message, redirecting it to a fake NRF under their control. This allows the attackers to reroute NFs to their malicious 5G core.

Exploiting cloud vulnerabilities: An attacker with access to the underlying host in a 5G cloud-native deployment can run packet capture utilities directly on the node. If the traffic between microservices is unencrypted, the attacker can easily capture sensitive information such as authentication tokens, session data, or payload content, thereby compromising confidentiality. Since microservices share the same kernel, any vulnerability in the host operating system or its kernel can allow an attacker to escalate privileges and access all inter-service communication. If one microservice is compromised, the attacker may have access to all data exchanged between microservices on the same node, which undermines the security of the entire system.

Under this threat model, four key requirements can be derived for securing the cloud-native 5G control plane:

(1) Confidentiality, Integrity, and Availability (CIA): Data exchanged between 5G microservices must be kept confidential through encryption. Unlike the one-way authentication common in Internet-based services, where only the client verifies the server's identity, cloud-native microservices demand mutual authentication.

(2) Ease of deployment: A security solution must be transparent to 5G microservices, requiring no code changes. Ideally, deployment should be accomplished via a webhook, labels, or a proxy injection mechanism. Additionally, the security tool should not require changes to the underlying host and should not need additional privileges such as root access, additional network interfaces, or the installation of custom kernel modules.

(3) Low overhead: Commonly deployed security solutions, such as VPNs and IPSec, provide blanket protection to the underlying network, but they create an IP-in-IP tunnel, adding per-packet overhead and creating fragmentation issues. A security solution should be lightweight.

(4) Customizability: A security solution should allow flexibility in the selection of policies, encryption algorithms, authentication schemes, and key sizes, and adapt to external (e.g., RAN) security threats.

III. Challenges

Three primary challenges make securing the cloud-native 5G control plane microservices difficult:

(1) Lack of integrated security in microservices: 5G NFs are often provided as black-box microservices by software vendors, without built-in security components. This requires operators to implement security externally.

(2) Operational and performance impact: Security solutions must maintain high performance. A security tool that degrades performance or alters the functionality of microservices negates the modularity and rapid deployment advantages of a cloud-native deployment. Further, requiring software vendors to modify their codebases to accommodate specific security tools is often impractical and costly.

(3) Public cloud infrastructure constraints: When deploying 5G microservices on public cloud platforms, operators must contend with the fixed nature of the underlying Infrastructure as a Service and Platform as a Service components, which are managed by the cloud provider. Security tools must function independently of container engines, CPU architectures, storage solutions, hypervisors, and kernel versions in use.

IV. Limitations of Current Approaches

Cloud-native microservices use virtual IP addresses assigned by an orchestrator, so transparent operation of the security mechanism is essential. Current industry standards for cloud-native security propose several architectures for different network stacks and threat models as described below.

a) Service Mesh-Based Security: This architecture, implemented in Istio, Linkerd, and Consul, introduces a transparent proxy adjacent to each microservice. The proxy intercepts incoming and outgoing packets, enforcing security policies without being visible to the microservice. Each microservice is typically deployed in its own network namespace, and the service mesh-based tools ensure that the transparent proxy shares this namespace.

Transparent packet interception is achieved using `iptables` redirect rules that modify the outgoing packets so that their destination IP address is set to localhost and the destination port is set to the proxy port. As a result, the transparent proxy receives the outgoing packet, establishes a TLS tunnel with another proxy, and forwards the packet to the destination proxy.

This approach, however, presents a challenge. When the `iptables` rules redirect the packet, the original destination IP address and port are altered. The transparent proxy must recover the original destination IP address and port to correctly route the packet. The proxy utilizes a socket option known as `SO_ORIGINAL_DST` to recover the original values. This option enables applications within the same network namespace to retrieve the original destination IP address and port from the kernel, even after they have been overwritten by the `iptables` rules. Once the transparent proxy has recovered this information, it applies the necessary security policies and transmits the packet over the TLS tunnel. The socket option described above, however, only functions with sockets that are "accepted," i.e., that support connection-oriented transport protocols. Therefore, it supports TCP but not UDP or SCTP.

b) Kernel or eBPF-Based Security: An example cloud-native security tool that operates at the kernel level using eBPF is Cilium [1]. Cilium has visibility into all packets regardless of the transport protocol, allowing it access to all relevant data structures. Since it processes packets at the kernel level, it does not need to reroute them, avoiding the need for a proxy and preserving transport and application protocol data structures.

Cilium utilizes IPSec or WireGuard encryption mechanisms to secure packets. However, since microservices share the host kernel to make system calls, this security implementation

operates at the host level. Consequently, communication between microservices on the same node remains unencrypted.

c) Node-Level Security Solutions: Istio's ambient mode introduces a proxy at the node level rather than using a sidecar container. This node-level proxy intercepts all incoming and outgoing traffic. However, it faces the same issue as Cilium's eBPF-based solution: traffic between microservices on the same host is unencrypted, violating zero-trust security.

d) Envoy Proxy for UDP: Most industry standard tools use Envoy [4] as a proxy for its packet interception mechanism. Envoy provides limited UDP support in the form of a UDP proxy, but it is a non-transparent proxy. As a result, the communicating party will be aware that the packet has been intercepted and rerouted by the proxy. Additionally, while there is another socket option available to retrieve the original IP address of a UDP packet, it does not address the issue of routing to the correct process.

In summary, in cloud-native 5G core deployments, especially in public cloud environments where access and privileges can be restricted, there is no comprehensive solution that fully supports all 5G core protocol stacks.

V. The ZTX-SEM Approach

In the context of a cloud-native 5G control plane (core), the ZTX-SEM solution follows a zero-trust security model that effectively addresses the vulnerabilities outlined in Section II. This section describes the components and workflow of the ZTX-SEM solution based on Figure 1.

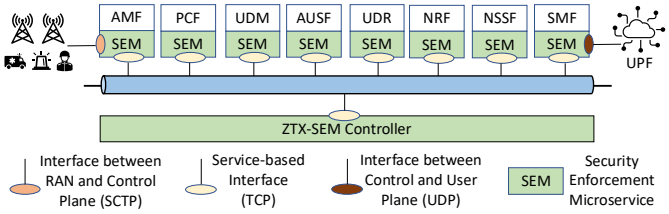

Fig. 1: ZTX-SEM solution for the 5G control plane.

A. Components of ZTX-SEM

The ZTX-Security Enforcement Microservice (SEM) is depicted in Figure 1. ZTX-SEM functions in a manner analogous to a transparent proxy, such as those found in the service mesh-based security architectures Istio, Linkerd [8], and Consul [3]. ZTX-SEM is deployed as a sidecar within the same network namespace as the microservice, where it intercepts all incoming and outgoing packets. In addition to the ZTX-SEM sidecar, we develop a "ZTX-SEM-controller." The ZTX-SEM-controller orchestrates security across a cluster.

1) Authentication: 5G microservices and the ZTX-SEM controller mutually authenticate using a public key infrastructure (PKI). The certificates and keys for this authentication are distributed through a secure bootstrapping process, utilizing external secret management systems to ensure zero trust. The distribution process follows standard protocols to ensure that only the correct entities receive them.

2) Shared Secret Derivation: Upon successful authentication, each microservice establishes a mutual TLS (mTLS) tunnel with the ZTX-SEM controller, and they collectively generate a shared secret. This shared secret serves as the encryption and decryption key for secure data transfer. The mTLS authentication channel acts as a control channel to update the key or generate the shared secret used for data channel encryption and decryption. The shared secret is unique to each pair of microservices.

3) ZTX-SEM-controller: The ZTX-SEM-controller maintains two critical tables in its memory. The first table contains a list of authenticated microservices and their identities, such as IP addresses or Fully Qualified Domain Names (FQDNs). The second table consists of the mapping between service or virtual IPs and their corresponding endpoint IPs, also known as pod or microservice IPs. This table is dynamically updated whenever a new microservice is spun up and successfully authenticates with the ZTX-SEM-controller. To keep these entries current, the controller queries the upstream Kubernetes API and updates the table accordingly. The table is essential because a microservice may receive packets from either a service IP or an endpoint IP, and must ensure the consistent use of the same shared secret across the same process.

4) ZTX-SEM Workflow: An authenticated microservice periodically receives data from the two tables maintained by the ZTX-SEM-controller. Upon receiving the list of authenticated microservices, ZTX-SEM initiates a mutual authentication and shared secret derivation process with each authenticated microservice, mirroring the process used between an individual microservice and the ZTX-SEM-controller.

ZTX-SEM employs an event-based mechanism to trigger actions upon receiving the updates from the controller. Updates may pertain to newly deployed microservices or updated service-to-endpoint mappings. The controller sends this information in two scenarios: first, when a new microservice is deployed and authenticated, the information is sent to that particular instance; second, when any information is updated, it is broadcast to all authenticated microservices.

5) Proactive Authentication: ZTX-SEM drops all packets until authentication with the destination peer is complete. Other transparent sidecar proxies, such as Istio or Consul, typically initiate peer authentication only upon the exchange of the first packet. In contrast, each microservice in our case proactively authenticates with all other microservices as soon as it receives the list from the ZTX-SEM-controller, ensuring readiness for data traffic exchange. While a particular microservice may unnecessarily authenticate with another, this only occurs once at startup, not per packet, so the performance impact is negligible. Proactive authentication is critical for 5G use cases with millisecond-level latency requirements.

6) Accelerated Secret Key Lookup: Each microservice maintains a unique shared secret key for communication with every other microservice. Consequently, the table containing IP addresses to shared secret key mappings can grow to several hundred entries. ZTX-SEM is stateless and therefore must perform a lookup for the appropriate key from memory

for every incoming or outgoing packet. ZTX-SEM utilizes hash tables to reduce lookup latency. As soon as the IP-to-shared-secret mapping table is updated, a corresponding hashed entry is added or updated in a separate hash table maintained in memory. This ensures O(1) lookup.

B. The ZTX-SEM Packet Handler

We use the `libnetfilter_queue` user space library [7] to intercept incoming and outgoing packets from a 5G NF container within a pod such that they can be forwarded to ZTX-SEM where security policies are applied. We also introduce an `init container` as a part of the Kubernetes deployment workflow to install `iptables` rules in the pod network namespace and set the packet filter for interception.

An API exposed by the OS is used to read packets that have been queued by the kernel packet filter. Security policies are defined in a callback function that is applied on every packet. We use this callback function to read packets from kernel space into the user space ZTX-SEM. Packets are sent out using raw sockets as shown in Figure 2. This ensures that the IP layer and port numbers remain intact and ZTX-SEM can deliver packets to the correct destination service and process without the need for NAT, rerouting, or other socket options. This packet-interception approach reduces exposure to network-based attacks by allowing programmatic verification and filtering of each packet based on security policies. In contrast, the traditional `iptables` redirection method introduces risks such as service hijacking and IP spoofing.

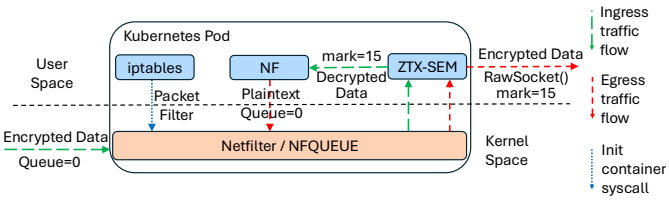

Fig. 2: ZTX-SEM packet flow.

If a packet matches the set filter, ZTX-SEM encrypts the outgoing packet or decrypts the incoming packet and forwards it to the intended receiver. If either fails, it drops the packet. Packets processed by ZTX-SEM are internally marked, enabling the kernel to distinguish them from unprocessed packets. This marking prevents an infinite loop in the packet interception mechanism by ensuring that processed packets are not repeatedly intercepted and processed. For 5G core deployment, we set the packet filter to capture all TCP, UDP and SCTP packets except for DNS packets.

We keep the TCP header intact and encrypt all the payload data that is present after the options header. Similarly, for UDP traffic, the header field is left unencrypted since it is required by the network stack to deliver packets, but we encrypt the rest of the data. Since SCTP adds multiple chunks in a packet, we combine all chunks and then encrypt under a single "Data" chunk. On the receiver side, the decryption module removes this additional chunk header and delivers the original chunk to the 5G NF.

Adding encryption can potentially increase the total packet size, causing it to exceed the Maximum Transmission Unit (MTU). We thus adjust the TCP Maximum Segment Size (MSS) in TCP-SYN packets on the fly, to be a few bytes less than the MTU. This ensures that the packet size remains within the MTU limit even after encryption. SCTP is a message-oriented protocol, so message-level fragmentation is supported. If the payload size exceeds 85% of the MTU, we first fragment the unencrypted packet, then encrypt each fragment individually. This approach preserves the message boundaries while accommodating the encryption overhead.

VI. Evaluation

The objective of our evaluation is threefold: (1) to compare the ZTX-SEM session setup time and per-transaction completion time to the Istio case, (2) to micro-benchmark the cost of adding security at each interface of the 5G core, and evaluate the time added to the total session setup time, compared to unsecured and Istio-based deployments, and (3) to compare the overhead of ZTX-SEM in terms of CPU and memory utilization to a deployment without security and to a deployment with Istio-based security. While these experiments focus on the performance impact, we have verified the security of ZTX-SEM by mimicking several of the attacks discussed in Section II.

A. Setup

We conduct experiments on CloudLab [14] where we create a Kubernetes cluster with four nodes: one serves as the manager and three serve as workers. Another node simulates the RAN interfaces between gNodeB and the core, specifically the N2 interface. All nodes are of type RS630, equipped with an Intel Xeon E5-2660 processor (x86-64 architecture), consisting of 40 CPUs with a maximum speed of 2.6 GHz. The nodes run kernel version `5.15.0-86-generic` with Ubuntu 22.04 operating system and Kubernetes version 1.30.3.

We deploy Open5GS v2.7.0 (an open-source 5G core implementation in C) [9] on the cluster and use `gnbsim v1.4.3` (an open-source tool in Go) [5] for simulating gNodeB. We use Helm charts from [16] to deploy, manage, and upgrade cloud-native Open5GS functions.

The ZTX-SEM security solution is implemented in C and utilizes OpenSSL for cryptographic operations, including encryption, decryption, and authentication. ZTX-SEM employs general socket libraries and kernel-exposed APIs for packet interception. The entire code is compiled and packaged as a docker container, which is then pulled and installed via Helm charts, transparently at runtime next to each microservice. The security features are enabled based on flags selected for a particular microservice and a specific interface of the 5G core as shown in Table I.

We implemented ZTX-SEM independently of Istio because modifying Istio to include ZTX-SEM's security mechanisms

would require a fundamental redesign of Istio's packet interception and routing. We compare ZTX-SEM with a customized deployment of Istio (v1.22.1) where non-essential functionality (i.e., rate limiting and observability) is disabled to ensure a fair comparison.

TABLE I: ZTX-SEM across 5G interfaces.

Interface	Network Functions (NFs)	Transport Protocol	Application Protocol
SBI	Between Core NFs	TCP	HTTP
N2	RAN and Core NFs	SCTP	NGAP
N4	Control and User Plane	UDP	PFCP
N18	Any NF and Database	TCP	MongoDB Wire

B. Preliminary Results

We experiment with the User Equipment (UE) registration procedure, which involves several transactions, including UE registration, PDU session establishment, and other core network processing tasks. We record three distinct time intervals: the time to complete UE registration, the time to complete the PDU session establishment procedure, and the time to complete the entire session setup. These times are collected using the `gnbsim` tool and represent the interval between the first request being sent and the last response being received.

1) Comparing Latency of ZTX-SEM and Istio: We compare individual transaction times and the time to complete the entire procedure across two deployments: one with ZTX-SEM security implemented and the other with Istio-based security implemented. We show the mean values and 95% confidence intervals, across three runs, for scenarios ranging from 100 to 1000 simultaneous requests. As shown in Figures 3, 4, and 6, the individual transaction times for UE registration, PDU session establishment, and the total session setup procedure completion time with ZTX-SEM are 28%, 26% and 32% lower than with the Istio deployment, respectively. Proactive authentication and the simple packet processing pipeline in ZTX-SEM are part of the reason. ZTX-SEM, being stateless in nature, operates on a per-packet basis and does not maintain flow information as Istio does.

2) Micro-Benchmarking 5G Core Interfaces: We enable security independently at different interfaces of the 5G core. We compare the overhead of this security in terms of the total session setup procedure completion time with a deployment without any security, and a deployment with Istio-based security. We use "unsecured" to refer to a deployment without any security, "RAN secured" to indicate that ZTX-SEM security is implemented between the RAN and the 5G core across the N2 interface, "PFCP secured" to mean that ZTX-SEM security is enabled across the interface between the control and user plane of the 5G core, specifically the N4 interface that runs over the UDP/PFCP protocol, "Core secured" to mean ZTX-SEM security is enabled across the SBI interfaces of the 5G core that runs over TCP, and "All secured" refers to security being enabled across all interfaces of the 5G core. "Istio" denotes that security is enabled using Istio across only the SBI interface of the 5G core.

As illustrated in Figure 6, when ZTX-SEM security is independently enabled across different interfaces, its effect on total session setup time is negligible compared to the deployment without any security. Further, when ZTX-SEM security is enabled across all interfaces of the 5G core, the session setup completion time is only 9% more than the deployment without any security. In contrast, Istio incurs around 34% higher session setup time than the deployment without any security. The increased latency of Istio may stem from its complex packet interception pipeline.

3) Resource Utilization: We analyze the resource utilization of ZTX-SEM in Figure 5. For Istio, we measure the CPU and memory utilization of the Istio transparent proxy. We observe that the CPU utilization for ZTX-SEM is only 7% higher than the deployment without any security. We find that the memory utilization remains the same. This is again due to the stateless nature of ZTX-SEM, as it discards all data structures as soon as the packet is processed and does not maintain any information across packets, resulting in constant memory utilization. In comparison, Istio exhibits 13% higher CPU utilization and non-zero memory utilization due to its flow-based packet interception and security policy application mechanism. When comparing the core security functionality between ZTX-SEM and Istio, ZTX-SEM yields 75% lower average CPU utilization.

VII. Discussion and Future Work

Secure 5G network slicing: Operators are creating different 5G network slices based on specific use cases. Similarly, ZTX-SEM can be deployed as a secure slice, particularly for military, emergency, and critical service use cases, co-existing with other slices.

Non-Cloud-Native deployments: Unlike industry solutions discussed in Section IV, ZTX-SEM can be easily configured to work with non-cloud-native deployments, such as monolithic hardware. By passing an additional deployment type flag, specific functionalities can be enabled or disabled to support monolithic architectures.

Adaptive security control: We will extend ZTM-SEM with dynamic and adaptive selection of security parameters. This adaptation can consider the level of security threat.

Modular ZTX-SEM security framework: ZTX-SEM is highly customizable, allowing packets received to traverse multiple pipelines as needed. Currently, ZTX-SEM processes packets through encryption, decryption, and authentication modules. Additional modules, such as rate limiting, can be added to mitigate DDoS attacks.

Post-Quantum cryptography: ZTX-SEM utilizes industry-standard AES-GCM-256 encryption, which ensures integrity through its tag header. An important direction for future work is enhancing cryptographic operations to be quantum-resistant.

VIII. Related Work

Several approaches have been proposed to address 5G network security, but many fall short in supporting cloud-native

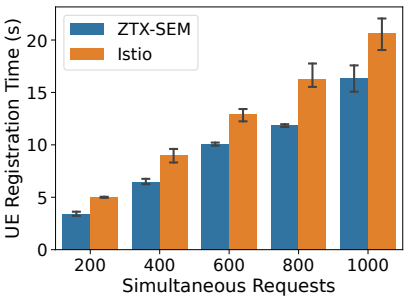

Fig. 3: UE registration time

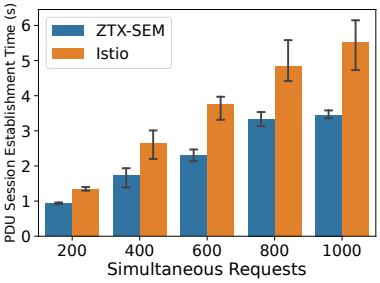

Fig. 4: PDU session establishment time

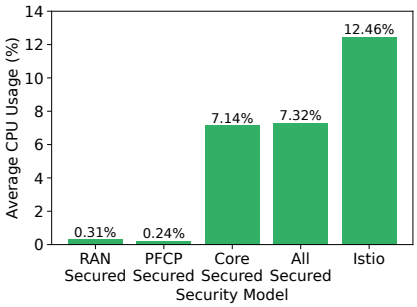

Fig. 5: CPU usage

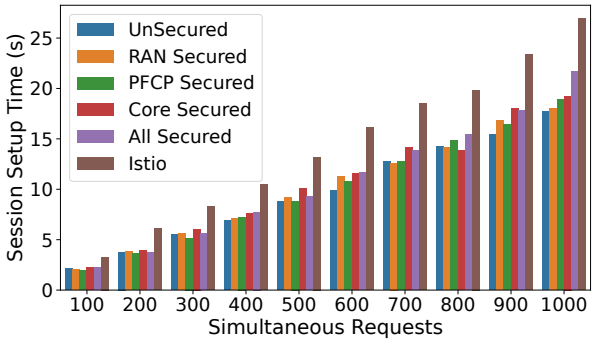

Fig. 6: Security across different interfaces

deployments or zero-trust models. For example, Loureiro et al. [17] explore the use of eBPF/XDP for rate-limiting control-plane traffic within the 5G core. Similarly, the vEPC-vSDP framework [13] enhances secure communications within the mobile core through an authentication-based approach. OpenZiti [10], an open-source zero-trust security framework, delivers a range of network-layer solutions but lacks support for 5G-specific transport protocols like SCTP due to TProxy limitations. Zscaler, a closed-source enterprise security platform, does not support service-to-service data security for microservices deployed on the same node.

Additionally, a number of studies develop threat detection mechanisms but do not incorporate robust encryption or authentication strategies. For instance, Hu et al. [15] explore security vulnerabilities in cellular emergency services. 5GCVerif [11] conducts a formal analysis of the 5G core access control mechanism. PROV5GC [19] detects attacks using provenance graphs.

IX. Conclusions

This paper presented ZTX-SEM, a versatile and practical zero-trust security solution designed for cloud-native deployments. ZTX-SEM introduces a protocol-agnostic packet interception mechanism, proactive authentication to ensure minimal latency, and an optimized secret key lookup process for efficient encrypted communication. Our preliminary results confirm that ZTX-SEM significantly reduces resource utilization and improves session setup times compared to traditional solutions such as Istio. These findings demonstrate the potential of ZTX-SEM to enhance security, while maintaining high performance, in cloud-native environments.

References

[1] Cilium. https://cilium.io/.

[2] Cloud-native service framework for 5G. https://www.3gpp.org/news-events/partner-news/openapis-for-the-service-based-architecture.

[3] Consul. https://www.consul.io/.

[4] Envoy. https://www.envoyproxy.io/.

[5] gnbsim. https://github.com/omec-project/gnbsim.

[6] Istio. https://istio.io/.

[7] libnetfilter_queue: firewalling, nat and packet mangling for linux. https://netfilter.org/projects/libnetfilter_queue/.

[8] Linkerd. https://linkerd.io/.

[9] Open5gs. https://github.com/open5gs/open5gs.

[10] Openziti is a programmable network overlay and associated edge components for application-embedded, zero-trust networking. https://github.com/openziti.

[11] Mujtahid Akon, Tianchang Yang, Yilu Dong, and Syed Rafiul Hussain. Formal analysis of access control mechanism of 5G core network. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, CCS '23, page 666–680, 2023.

[12] David Basin, Jannik Dreier, Lucca Hirschi, Saša Radomirovic, Ralf Sasse, and Vincent Stettler. A formal analysis of 5G authentication. In *Proceedings of the 2018 ACM SIGSAC CCS*, page 1383–1396, 2018.

[13] Yahuza Bello, Ahmed Refaey Hussein, Mehmet Ulema, and Juanita Koilpillai. On sustained zero trust conceptualization security for mobile core networks in 5G and beyond. *IEEE Transactions on Network and Service Management*, 19(2):1876–1889, 2022.

[14] Dmitry Duplyakin et al. The design and operation of CloudLab. In *Proceedings of the USENIX (ATC)*, pages 1–14, July 2019.

[15] Yiwen Hu et al. Uncovering insecure designs of cellular emergency services (911). In *Proceedings of the MobiCom '22*, page 703–715, 2022.

[16] Umakant Kulkarni, Amit Sheoran, and Sonia Fahmy. The cost of stateless network functions in 5G. In *Proceedings of the Symposium on Architectures for Networking and Communications Systems*, ANCS '21, page 73–79, 2022.

[17] Luís Loureiro, Vasco Pereira, Tiago Cruz, and Paulo Simões. Enhancing 5G core security with eBPF/XDP. In *NOMS 2024-2024 IEEE Network Operations and Management Symposium*, pages 1–6, 2024.

[18] Shiyue Nie, Yiming Zhang, Tao Wan, Haixin Duan, and Song Li. Measuring the deployment of 5G security enhancement. In *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec '22, page 169–174, 2022.

[19] Harsh Sanjay Pacherkar and Guanhua Yan. PROV5GC: Hardening 5G core network security with attack detection and attribution based on provenance graphs. In *Proceedings of the WiSec '24*, page 254–264, 2024.

[20] Altaf Shaik, Ravishankar Borgaonkar, Shinjo Park, and Jean-Pierre Seifert. New vulnerabilities in 4G and 5G cellular access network protocols: exposing device capabilities. In *Proceedings of the WiSec '19*, WiSec '19, page 221–231, 2019.

[21] V Stafford. Zero trust architecture. *NIST special publication*, 800:207, 2020.

[22] Yaru Yang et al. Uncovering security vulnerabilities in real-world implementation and deployment of 5G messaging services. In *Proceedings of the 17th ACM Conference on WiSec '24*, page 265–276, 2024.