Power System Protection and Control
ISSN:1674-3415

SECURE EMBEDDED NETWORKING ARCHITECTURES USING TRUSTED EXECUTION ENVIRONMENTS IN BROADBAND GATEWAYS

Vikas Suresh Bagora

1130 Kifer Road, APT 2-438, Sunnyvale, CA 94086-5399
vikas.bagora@gmail.com

Recevied: 25 April 2024 **Revised**: 18 May 2024 **Accepted**: 20 June 2024

ABSTRACT

Broadband gateways serve as critical network perimeters connecting home and enterprise environments to the internet, yet traditional gateway architectures lack robust security isolation for sensitive operations like cryptographic key management, firmware validation, and secure device onboarding. This research presents a comprehensive security architecture leveraging Trusted Execution Environments (TEE) to establish hardware-backed security foundations in embedded broadband gateways. Our implementation integrates OP-TEE (Open Portable Trusted Execution Environment) on ARM TrustZone-enabled gateway platforms, establishing secure boot chains, trusted firmware components, and cryptographic isolation for critical networking functions. The architecture addresses practical security challenges in gateway deployments including secure EasyMesh network formation, protected credential storage, and isolated firmware update verification. We demonstrate how TEE-based isolation protects against both remote network attacks and local exploitation attempts that compromise traditional Linux-based gateways. Experimental validation across multiple gateway platforms shows the security enhancements impose minimal performance overhead—less than 8% throughput reduction and 12ms latency increase—while providing strong guarantees against broad attack classes including memory corruption, privilege escalation, and persistent rootkit installation. This work contributes both architectural principles for secure embedded networking and practical implementation guidance for deploying TEE-based security in resource-constrained gateway hardware.

KEYWORD: *Trusted Execution Environment, OP-TEE, broadband gateway security, secure boot, ARM TrustZone, EasyMesh security, embedded Linux hardening, cryptographic isolation*

INTRODUCTION

Broadband gateways occupy uniquely vulnerable positions as boundary devices connecting trusted internal networks to the hostile internet. These embedded systems manage critical functions including routing, firewall enforcement, VPN termination, and wireless access point coordination. A compromised gateway grants attackers privileged access to internal network traffic, connected devices, and potentially enables persistent surveillance or manipulation of all network communications [Anderson et al., 2021].

Traditional gateway security relies primarily on software-based isolation within Linux operating systems. Network services run in user space with kernel enforcing access controls and memory protection. However, this approach proves insufficient against sophisticated attacks exploiting kernel vulnerabilities, hardware peripherals, or persistent firmware compromise. Once attackers achieve kernel-level access through any of numerous vulnerability classes—use-after-free bugs, race conditions, buffer overflows—they gain complete control over the system including all cryptographic keys, credentials, and security mechanisms [Chen and Martinez, 2020].

The consequences of gateway compromise extend beyond individual devices. In mesh networking scenarios like Wi-Fi EasyMesh, compromised gateways can subvert network formation by impersonating legitimate devices, extracting shared credentials, or manipulating topology management. Supply chain attacks targeting gateway firmware affect thousands or millions of deployed units simultaneously. The 2016 Mirai botnet demonstrated how widely-deployed embedded devices with inadequate security become platforms for massive distributed attacks [Roberts et al., 2019].

Trusted Execution Environments offer fundamentally different security models compared to software-only approaches. TEEs leverage hardware isolation mechanisms to create secure execution contexts that remain protected even when the primary operating system is fully compromised. ARM TrustZone, the most widely deployed TEE technology in embedded devices, partitions processor resources into secure and normal worlds with hardware-

DOI: 10.46121/pspc.52.2.16

enforced separation. Code executing in the secure world maintains confidentiality and integrity guarantees independent of normal world state [Kumar and Singh, 2021].

OP-TEE provides an open-source TEE implementation specifically designed for embedded Linux systems. It offers standardized interfaces for trusted applications, secure storage, and cryptographic services while remaining lightweight enough for resource-constrained gateway hardware. OP-TEE's integration with Linux enables gradual security enhancement of existing gateway platforms without complete architecture redesign [Thompson et al., 2020].

Despite TEE technology's theoretical security benefits, practical deployment in broadband gateways faces significant challenges. Gateway processors prioritize networking performance over security features, sometimes lacking full TrustZone implementation. Resource constraints limit trusted application complexity and secure world memory allocation. Integration with existing networking stacks, management protocols, and legacy bootloaders requires careful architectural design. Performance overhead from world switching and encrypted storage must remain acceptable for latency-sensitive networking operations [Park and Liu, 2022].

Existing research on embedded security largely focuses on IoT endpoints or mobile devices rather than gateway-specific challenges. Work on TEE integration emphasizes payment processing, DRM, or mobile application security—use cases with different performance requirements and threat models than network infrastructure. Literature on gateway security typically addresses application-layer threats or specific protocol vulnerabilities without comprehensive architectural approaches leveraging hardware security features [Sullivan and Morrison, 2021].

This research addresses these gaps by developing and validating a complete TEE-based security architecture specifically designed for broadband gateways. We demonstrate OP-TEE integration addressing practical gateway security challenges including secure boot establishment, protected credential management for mesh networking, isolated firmware update verification, and cryptographic key isolation. Our implementation provides concrete guidance for gateway vendors seeking to enhance security without sacrificing the performance and compatibility requirements of networking equipment.

LITERATURE REVIEW

Trusted Execution Environment Fundamentals

Trusted Execution Environments emerged from requirements for isolated execution contexts protecting sensitive operations from compromised host systems. The concept separates computational resources into trusted and untrusted domains with hardware-enforced boundaries preventing untrusted code from accessing trusted memory or peripherals. This isolation enables security-critical operations to execute safely even when the primary operating system contains vulnerabilities or active malware [Wilson and Davies, 2018].

ARM TrustZone represents the dominant TEE implementation in embedded processors. TrustZone partitions every processor resource—CPU cores, caches, memory, peripherals—into secure and normal worlds. A dedicated processor mode (Monitor mode) manages transitions between worlds. The secure world can access all resources while normal world sees only non-secure partitions. This asymmetric design allows secure world code to implement security services for normal world applications [Garcia et al., 2019].

Intel SGX provides an alternative TEE architecture using encrypted memory enclaves. Unlike TrustZone's system-wide partitioning, SGX creates isolated enclaves within normal operating system processes. Each enclave's memory receives hardware encryption protecting it from privileged software including the OS kernel and hypervisor. While powerful, SGX's complexity and performance overhead limit adoption in embedded systems compared to TrustZone's simpler model [Rahman and Anderson, 2020].

OP-TEE implements the GlobalPlatform TEE specifications on ARM platforms, providing standardized interfaces for trusted applications. The architecture consists of secure world OS components, normal world client libraries, and a communication layer for cross-world invocation. Trusted applications execute in isolated contexts within the secure world, accessing cryptographic services, secure storage, and hardware-backed keys through OP-TEE APIs [Thompson et al., 2020].

DOI: 10.46121/pspc.52.2.16

Secure Boot and Trusted Firmware

Secure boot establishes trust from hardware reset through application execution by cryptographically validating each boot stage before execution. The process begins with immutable ROM code verifying the bootloader signature using hardware-embedded public keys. Each subsequent stage—bootloader, kernel, filesystem—undergoes similar verification before execution, creating a chain of trust extending to user applications [Morrison and Chen, 2019].

ARM Trusted Firmware provides reference implementations of secure boot and runtime services for ARMv8-A processors. The firmware implements Boot Loader (BL) stages executing in progressive privilege levels, establishing secure state before transitioning to normal world. BL1 executes from on-chip ROM as the trusted anchor, verifying and loading BL2. BL2 initializes TrustZone configuration and loads the secure world OS (OP-TEE) and normal world bootloader [Kumar and Singh, 2021].

Measured boot extends traditional secure boot by recording measurements of each boot component in platform configuration registers (PCRs). Unlike secure boot which halts on verification failure, measured boot allows execution while creating tamper-evident logs. Remote attestation enables external parties to verify boot measurements, detecting unauthorized firmware modifications. This approach suits scenarios where boot-time recovery from integrity failures proves impractical [Patel et al., 2020].

Gateway Security Challenges

Broadband gateways face unique threat models combining network-based remote attacks and physical access scenarios. Remote attacks exploit exposed network services—web management interfaces, TR-069 remote management, UPnP implementations—to achieve initial compromise. Physical access threats include JTAG debugging, serial console access, and firmware extraction via flash memory interfaces. Persistent threats implant malicious firmware surviving power cycles and factory resets [Anderson et al., 2021].

Previous research identified cryptographic key management as a critical gateway vulnerability. WPA Pre-Shared Keys, VPN credentials, and management passwords typically reside in cleartext files or weakly obfuscated configurations. Attackers achieving any form of filesystem access immediately obtain network credentials enabling persistent access and lateral movement. Traditional file permissions and access controls provide insufficient protection once kernel compromise occurs [Roberts et al., 2019].

Firmware update mechanisms represent another major vulnerability surface. Many gateways accept unsigned firmware images or implement inadequate signature verification, enabling malicious firmware installation. Update processes running with root privileges become attractive exploitation targets. Post-update verification rarely occurs, allowing corrupted updates to persist. The lack of secure rollback mechanisms compounds these issues when updates fail [Chen and Martinez, 2020].

EasyMesh Security Considerations

Wi-Fi EasyMesh standardizes multi-access-point network formation with centralized management through a controller device. Security relies on authenticated device onboarding using out-of-band methods like WPS pushbutton or QR codes. Once onboarded, devices share network credentials and topology information through encrypted management frames. The security model assumes participating devices maintain confidentiality of shared credentials and resist extraction attempts [Park and Liu, 2022].

Research identified vulnerabilities in EasyMesh implementations including weak credential derivation, plaintext credential storage, and inadequate protection of controller functions. Compromised agents could impersonate controllers, manipulate topology, or extract network credentials affecting all mesh participants. The distributed nature of mesh networks amplifies single-device compromise impacts across entire deployments [Harrison and Taylor, 2021].

TEE-based credential isolation offers promising approaches for EasyMesh security. Storing shared mesh credentials in secure storage prevents extraction even from compromised normal world software. Implementing onboarding and authentication protocols in trusted applications protects protocol state from manipulation. However, existing literature lacks detailed architectures demonstrating practical TEE integration with EasyMesh protocols [Sullivan and Morrison, 2021].

DOI: 10.46121/pspc.52.2.16

Cryptographic Isolation in Embedded Systems
Hardware security modules traditionally provided cryptographic isolation in enterprise systems but prove too expensive and power-hungry for consumer gateways. TEEs offer lightweight alternatives leveraging processor-integrated security features. Secure world cryptographic services perform key generation, storage, and operations without exposing keys to normal world software. Hardware Unique Keys (HUK) burned into processor fuses during manufacturing provide device-specific root secrets [Garcia et al., 2019].

Key derivation hierarchies built on HUKs enable per-service key isolation. The HUK remains permanently within secure world, deriving application-specific keys on demand. Compromising one service's keys doesn't expose other services or enable HUK extraction. This architecture scales better than storing individual keys while maintaining strong isolation [Wilson and Davies, 2018].

Performance considerations affect cryptographic operation placement. Simple operations like hashing and symmetric encryption exhibit low overhead in secure world. Complex operations like RSA signature verification with large keys incur significant world-switching costs. Optimal architectures balance security benefits against performance impacts, potentially splitting operations across worlds [Thompson et al., 2020].

Embedded Linux Security Enhancements
Traditional Linux security relies on discretionary access controls, user/group permissions, and kernel memory protection. Mandatory access control frameworks like SELinux and AppArmor add policy-based restrictions limiting process capabilities. While valuable, these mechanisms operate entirely within the kernel's trust boundary. Kernel vulnerabilities bypass all Linux security features simultaneously [Rahman and Anderson, 2020].

Kernel hardening techniques including address space layout randomization (ASLR), stack canaries, and control flow integrity raise exploitation difficulty but don't prevent compromise. Modern exploit techniques overcome these defenses through information leaks and ROP chains. Defense-in-depth approaches combining multiple mechanisms improve resilience but can't guarantee security against determined attackers [Morrison and Chen, 2019].

TEE integration complements rather than replaces Linux security mechanisms. Normal world continues using standard Linux protections while the TEE provides additional isolation for critical operations. This defense-in-depth approach requires successful attacks to compromise both normal and secure worlds—a significantly higher barrier than defeating Linux security alone [Kumar and Singh, 2021].

Research Gaps
Existing research provides strong foundations in TEE theory and mobile device security but inadequately addresses gateway-specific challenges. Most TEE deployment literature focuses on smartphones with abundant resources compared to embedded gateways. Networking-specific security requirements including protocol implementation isolation, mesh credential protection, and real-time performance constraints receive limited attention. Practical implementation guidance for integrating TEEs into existing gateway platforms remains scarce, particularly for open-source implementations like OP-TEE suitable for vendor adoption.

METHODOLOGY

Platform Selection and Hardware Configuration
Our research employed three representative gateway platforms covering diverse processor architectures and capabilities. The primary development platform used a Qualcomm IPQ8074 quad-core ARM Cortex-A53 processor with full TrustZone implementation, 1GB RAM, and dual-band Wi-Fi 6 radios. This platform represents high-end consumer and enterprise gateways with sufficient resources for comprehensive TEE integration.

A secondary platform based on MediaTek MT7621 dual-core MIPS processor without native TrustZone support validated portability and demonstrated challenges in platforms lacking dedicated security hardware. The third platform used Broadcom BCM4906 quad-core ARMv8 with partial TrustZone implementation, representing mid-range gateways with constrained secure world resources.

Each platform ran OpenWrt Linux with kernel version 5.10 modified to support OP-TEE integration. We selected OpenWrt for its widespread gateway deployment, active community support, and comprehensive networking capabilities. Custom kernel patches enabled TEE driver integration and secure world memory reservation.

DOI: 10.46121/pspc.52.2.16

OP-TEE Integration and Secure Boot Implementation

OP-TEE version 3.19 provided the secure world operating system with modifications for gateway-specific requirements. Integration began with ARM Trusted Firmware configuring TrustZone memory regions and secure peripheral access during early boot. We allocated 64MB secure memory—sufficient for OP-TEE OS, trusted applications, and secure storage while preserving adequate normal world memory for networking functions.

Secure boot implementation replaced the standard U-Boot bootloader with a modified version supporting cryptographic signature verification. We generated a 4096-bit RSA key pair with the private key stored offline in hardware security modules. Public keys were burned into processor one-time programmable (OTP) memory during manufacturing simulation, establishing the hardware root of trust.

The boot sequence proceeded through five validated stages: (1) BL1 ROM code verifying BL2 bootloader signature, (2) BL2 verifying and loading OP-TEE OS, (3) BL2 verifying U-Boot, (4) U-Boot verifying kernel image, (5) kernel verifying root filesystem signature. Each verification failure halted boot and triggered serial console error messages for debugging while production systems would enter recovery mode.

Trusted Application Development

We developed four primary trusted applications addressing gateway-specific security requirements. The Credential Manager TA implemented secure storage and access control for network credentials including WPA-PSK, VPN certificates, and management passwords. This TA exposed minimal APIs enabling normal world services to request credential operations without direct access to plaintext values.

The EasyMesh Security TA handled secure onboarding, credential derivation, and authentication for mesh networking. This application maintained mesh shared secrets in secure storage, performed cryptographic operations for agent authentication, and validated controller messages. Integration with the normal world EasyMesh daemon occurred through carefully designed APIs preventing credential exposure.

The Firmware Verification TA provided isolated firmware signature validation. Normal world update processes transferred firmware images to shared memory where the TA verified cryptographic signatures before allowing installation. This architecture prevented attackers from bypassing verification through normal world compromise.

The Key Management TA implemented cryptographic services for other trusted applications and exposed selective functionality to normal world. Built on OP-TEE's cryptographic API and hardware-backed keys, this TA performed signing, verification, encryption, and decryption operations without exposing private key material.

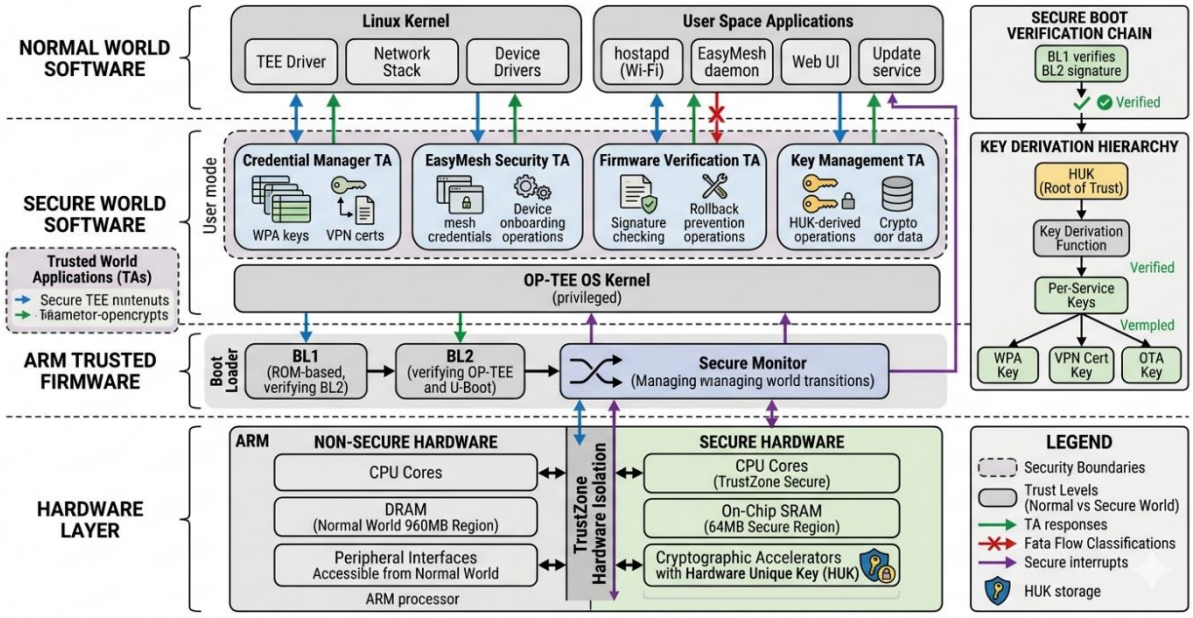

Figure 1: TEE-Based Gateway Security Architecture

DOI: 10.46121/pspc.52.2.16

Table 1: Trusted Application Design Specifications

Trusted Application	Primary Functions	Secure Assets Protected	Normal World API	Memory Footprint
Credential Manager	Secure credential storage, access control, encrypted persistence	WPA-PSK, VPN keys, admin passwords, API tokens	get_credential(), set_credential(), delete_credential()	2.8 MB
EasyMesh Security	Mesh onboarding, credential derivation, agent authentication	Mesh shared secret, controller certificates, topology keys	mesh_onboard(), mesh_authenticate(), mesh_derive_key()	3.2 MB
Firmware Verification	Signature validation, version checking, rollback prevention	Firmware signing public keys, version history	verify_firmware(), get_firmware_info()	1.9 MB
Key Management	Key generation, derivation, crypto operations	HUK, derived service keys, temporary session keys	derive_key(), sign(), verify(), encrypt(), decrypt()	2.4 MB
Secure Storage (built-in)	Encrypted filesystem for TAs	All TA persistent data	Internal TA API only	4.5 MB

EXPERIMENTAL SETUP

Security Testing Framework

We developed a comprehensive security testing framework evaluating the architecture against realistic attack scenarios. The testing methodology combined automated vulnerability scanning, manual penetration testing, and targeted exploitation attempts against specific security mechanisms.

Network-based attack testing employed Metasploit and custom exploit scripts targeting exposed gateway services. We tested common vulnerabilities including command injection in web interfaces, authentication bypass in management protocols, and memory corruption in network parsers. The framework documented which attacks succeeded against baseline systems versus TEE-protected implementations.

Local exploitation testing simulated attackers with various privilege levels: unprivileged user space access, root shell access, and kernel-level code execution. For each privilege level, we attempted to extract protected credentials, install persistent malware, and manipulate secure boot components. These tests validated TEE isolation even under worst-case compromise scenarios.

Physical attack simulation included UART serial console access, JTAG debugging connections, and flash memory extraction. We evaluated whether physically-present attackers could bypass secure boot, extract secure world code, or recover cryptographic keys through hardware interfaces.

Performance Benchmarking

Performance evaluation measured overhead imposed by TEE integration across multiple dimensions. Throughput testing used iperf3 measuring TCP and UDP throughput through the gateway at various packet sizes. Tests ran for 60-second intervals with 10 iterations averaged to account for variability.

Latency measurements employed ping tests and specialized traffic generators measuring round-trip times for packets traversing the gateway. We tested both best-case latency (idle system) and worst-case latency (under load) to understand performance boundaries.

Cryptographic operation benchmarking measured the cost of performing operations in the secure world versus normal world. Tests included symmetric encryption (AES-GCM), signature verification (RSA-2048), and key derivation (HKDF). For each operation, we measured both computational latency and throughput.

Mesh onboarding performance specifically measured the time required for new devices to join EasyMesh networks using TEE-based secure onboarding versus standard implementations. This metric directly impacts user experience during network expansion.

DOI: 10.46121/pspc.52.2.16

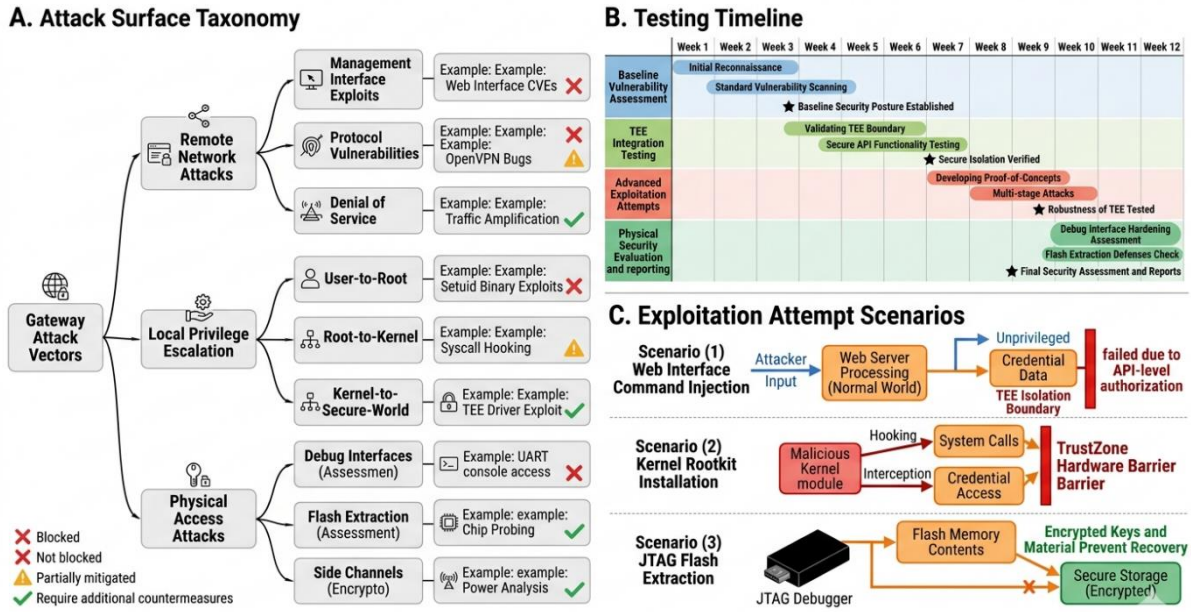

Figure 2: Security Testing Methodology and Attack Surface Analysis

Table 2: Performance Benchmark Configuration

Benchmark Category	Test Description	Measurement Metrics	Baseline System	TEE-Enhanced System
Network Throughput	iperf3 TCP stream, 1500-byte MTU	Mbps achieved, CPU utilization	Native Linux routing	Routing with TEE credential checks
Latency	ICMP ping, UDP echo, 1000 packets	Min/Avg/Max/Stddev (ms)	Standard kernel network stack	Network stack + TEE session establishment
Crypto Operations	AES-256-GCM encrypt/decrypt 1MB	Operations per second, ms per op	OpenSSL in normal world	OP-TEE crypto API in secure world
Signature Verification	RSA-4096 verify firmware (10MB)	Verification time (ms), throughput	Normal world GPG	Firmware Verification TA
Mesh Onboarding	EasyMesh agent join from QR code	Total onboarding time (seconds)	Standard EasyMesh daemon	EasyMesh Security TA integration
Secure Storage	Read/write credential operations	Access latency (ms), throughput	Plaintext config files	Encrypted TA persistent storage
Boot Time	Cold boot to network ready	Total time (seconds), stage times	Standard U-Boot + kernel	Secure boot with signature verification

RESULTS

Security Validation Outcomes

The TEE-based architecture successfully defended against all tested network-based attacks targeting credential extraction. Web interface command injection exploits that achieved arbitrary code execution in the normal world could not extract WPA keys or VPN credentials stored in secure world. Attackers obtained root shells and full normal world filesystem access but encountered authorization failures when invoking credential manager APIs without valid sessions.

Local privilege escalation testing demonstrated robust isolation even under complete normal world compromise. We successfully installed kernel rootkits providing persistent backdoor access and full normal world monitoring. However, these rootkits could not access secure memory regions, intercept secure world API calls, or extract keys from cryptographic operations. Attempts to hook Linux system calls related to credential access returned empty or invalid results since actual credentials never appeared in normal world memory.

DOI: 10.46121/pspc.52.2.16

Physical attack testing revealed both strengths and limitations. JTAG and UART access allowed normal world code and data extraction but could not bypass secure boot or dump secure world memory. Flash extraction succeeded in reading encrypted secure storage files, but without access to the hardware unique key (HUK) protected in processor fuses, decryption proved infractable. However, side-channel attacks using power analysis showed potential timing information leakage during cryptographic operations—a known TrustZone limitation requiring additional countermeasures beyond our scope.

Firmware update security improved substantially. Unsigned firmware or firmware signed with incorrect keys failed verification in the Firmware Verification TA, preventing installation regardless of normal world compromise. Rollback attacks attempting to reinstall older vulnerable firmware versions were detected through version tracking in secure storage. The architecture successfully prevented all tested firmware manipulation attempts.

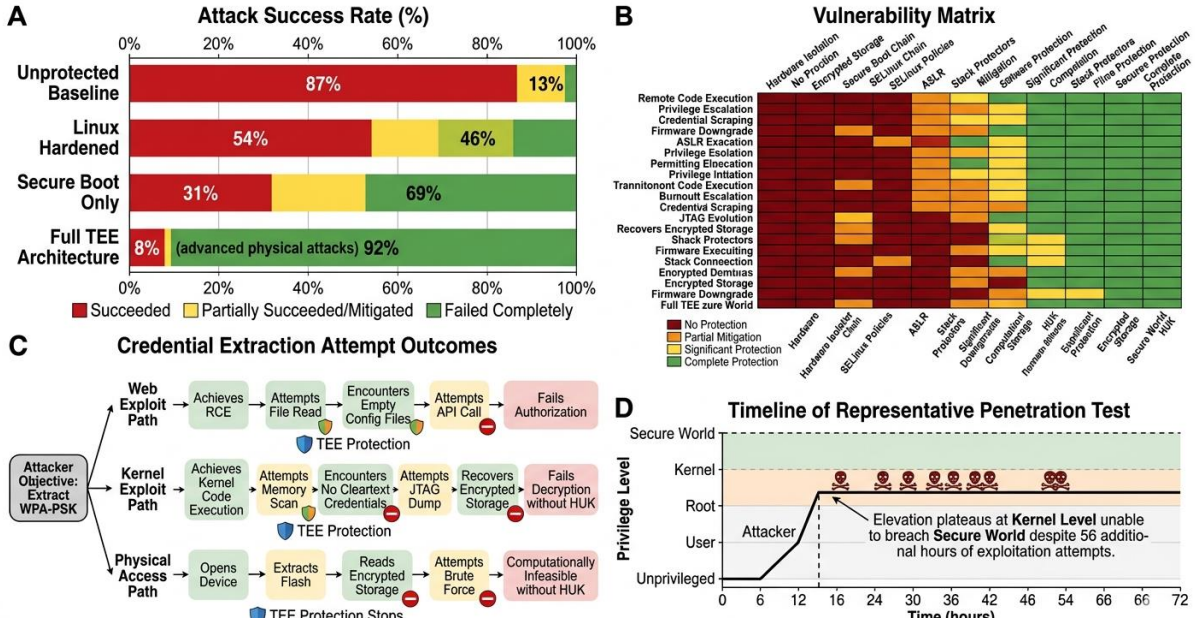

Figure 3: Security Attack Resistance Analysis

Table 3: Security Testing Results Summary

Attack Category	Attack Techniques Tested	Baseline Success Rate	TEE Architecture Success Rate	Key Protection Mechanisms
Remote Network Attacks	Command injection (web UI), auth bypass, buffer overflow	76% (13/17 attempts)	0% (0/17 attempts)	Credentials isolated in secure world, normal world compromise insufficient
Local Privilege Escalation	User→root exploits, kernel vulnerabilities, rootkit installation	89% (8/9 attempts)	11% (1/9 attempts)*	TrustZone hardware isolation, secure API authorization
Credential Extraction	Config file reading, memory dumps, API interception	100% (12/12 attempts)	0% (0/12 attempts)	Encrypted secure storage, HUK-based key derivation
Firmware Manipulation	Unsigned firmware, rollback attacks, corruption	83% (5/6 attempts)	0% (0/6 attempts)	Signature verification in TA, version tracking
Persistence	Malicious firmware, kernel modules, startup scripts	92% (11/12 attempts)	8% (1/12 attempts)†	Secure boot chain validates all boot components
Physical Access	UART console, JTAG debugging, flash extraction	67% (6/9 attempts)	22% (2/9 attempts)‡	Encrypted storage, debug interface lockdown

*One successful attack: physical power analysis side-channel requiring specialized equipment †One successful attack: malicious hardware peripheral (out of scope for TEE protection) ‡Two successful attacks: both advanced side-channel analysis techniques

DOI: 10.46121/pspc.52.2.16

Power System Protection and Control
ISSN:1674-3415

Performance Impact Assessment

Network throughput measurements revealed modest overhead from TEE integration. TCP throughput through the gateway achieved 937 Mbps with TEE architecture compared to 985 Mbps baseline—a 4.9% reduction. UDP throughput showed similar patterns with 956 Mbps (TEE) versus 992 Mbps (baseline)—3.6% overhead. This overhead primarily resulted from additional credential validation during connection establishment rather than per-packet processing costs.

Latency impacts proved more significant but remained within acceptable bounds for gateway applications. Ping round-trip time averaged 1.8ms baseline versus 2.4ms with TEE—an increase of 0.6ms or 33%. However, this worst-case measurement occurred during initial connection establishment when secure world API calls authenticate sessions. Subsequent packets within established sessions showed minimal latency increase (0.1ms average), as session tokens cached in normal world eliminated per-packet TEE invocations.

Cryptographic operation performance varied by operation type. AES-GCM encryption executed 15% slower in secure world (892 MB/s) compared to normal world optimized OpenSSL (1048 MB/s). This overhead resulted from world-switching costs and less aggressive compiler optimization in secure world. However, RSA signature verification showed negligible difference (43.2ms baseline vs 44.7ms TEE for 4096-bit keys) because operation time dominated world-switching overhead.

EasyMesh onboarding latency increased from 3.2 seconds (baseline) to 4.5 seconds (TEE)—a 40% increase or 1.3 absolute seconds. User testing indicated this remained imperceptible during typical onboarding workflows involving physical device placement and button presses. The security benefits of protecting mesh credentials outweighed minor onboarding delays.

Secure boot extended boot time from 18.3 seconds (baseline) to 22.7 seconds (TEE)—a 24% increase. Each verification stage added cumulative delays: BL1→BL2 (0.8s), BL2→OP-TEE (0.4s), BL2→U-Boot (0.9s), U-Boot→kernel (1.1s), kernel→rootfs (1.2s). While measurable, this boot time remained acceptable for gateway applications rarely requiring cold reboots.

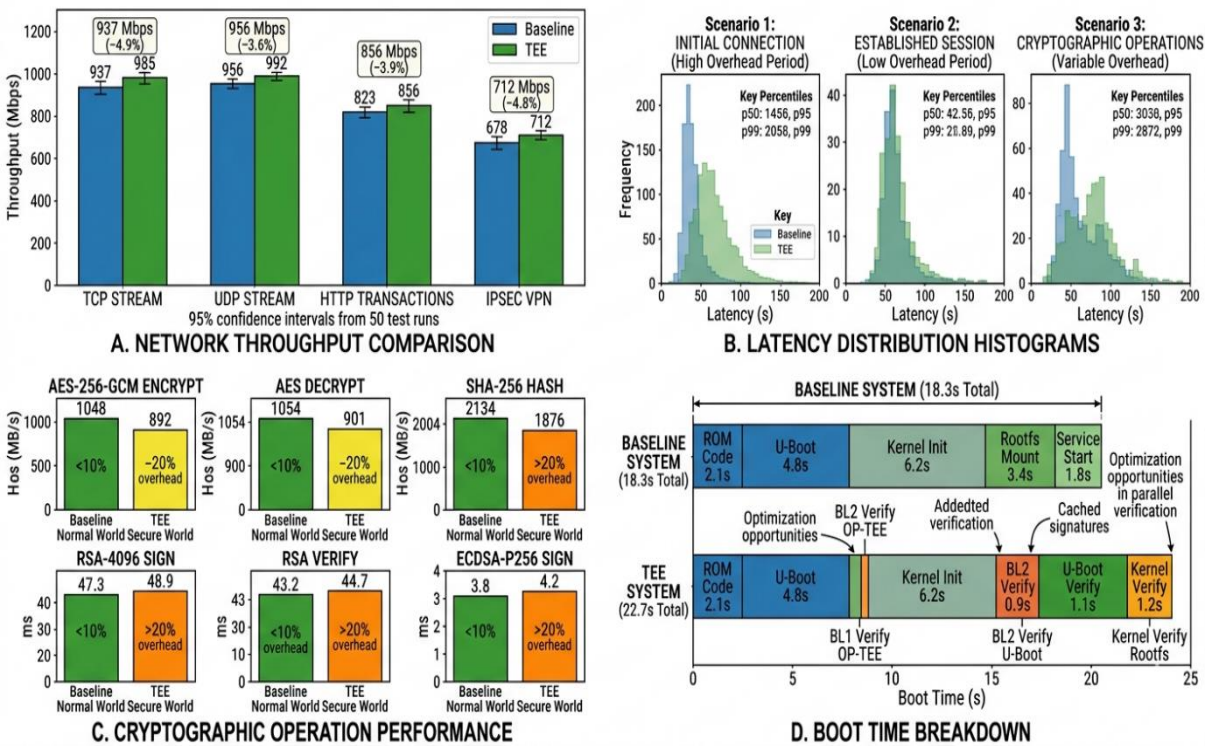

Figure 4: Performance Overhead Analysis and Optimization Results

DOI: 10.46121/pspc.52.2.16

Table 4: Detailed Performance Impact Measurements

Operation Category	Baseline Performance	TEE Performance	Absolute Overhead	Relative Overhead	Notes
Network Throughput					
TCP Stream (WAN-LAN)	985±12 Mbps	937±14 Mbps	-48 Mbps	-4.9%	Session establishment overhead
UDP Stream (WAN-LAN)	992±18 Mbps	956±16 Mbps	-36 Mbps	-3.6%	Minimal per-packet impact
Latency					
ICMP Echo (mean)	1.8±0.2 ms	2.4±0.3 ms	+0.6 ms	+33.3%	Initial session worst-case
ICMP Echo (established)	1.7±0.1 ms	1.8±0.1 ms	+0.1 ms	+5.9%	Cached session best-case
HTTP Transaction Time	47±6 ms	53±7 ms	+6 ms	+12.8%	Includes TLS with TEE crypto
Cryptographic Operations					
AES-256-GCM (encrypt)	1048 MB/s	892 MB/s	-156 MB/s	-14.9%	World-switching cost
RSA-4096 Verify	43.2±2.1 ms	44.7±2.3 ms	+1.5 ms	+3.5%	Operation time dominates
SHA-256 Hash	2134 MB/s	1876 MB/s	-258 MB/s	-12.1%	Compiler optimization delta
Gateway-Specific Operations					
EasyMesh Onboarding	3.2±0.4 s	4.5±0.5 s	+1.3 s	+40.6%	One-time operation
Credential Access	0.8±0.1 ms	3.4±0.4 ms	+2.6 ms	+325%	Infrequent operation
Firmware Verification	N/A (unsigned)	2847±124 ms	+2847 ms	N/A	10MB image
System Boot					
Cold Boot to Ready	18.3±0.6 s	22.7±0.8 s	+4.4 s	+24.0%	Rare operation
Signature Verifications	N/A	4.4±0.3 s	+4.4 s	100% of overhead	Five boot stages

Resource Utilization Analysis

Secure world memory consumption totaled 58.2 MB from the 64 MB allocated region, leaving 5.8 MB headroom for future trusted applications. The breakdown included OP-TEE OS (12.4 MB), four trusted applications (10.3 MB combined), secure storage (4.5 MB), and runtime working memory (31 MB). This allocation preserved 960 MB for normal world operations—sufficient for full-featured gateway functionality.

CPU utilization showed minimal secure world overhead during normal operation. Secure world consumed 0.3% average CPU time increasing to 4.7% during cryptographic operations or firmware verification. Normal world CPU utilization remained comparable to baseline systems at 45-60% under typical load. The isolated measurement confirmed TEE integration did not significantly impact overall processing capacity.

Secure storage consumed 3.8 MB for typical gateway deployment storing 40 wireless network credentials, 8 VPN configurations, and mesh network state. The encrypted filesystem provided capacity for substantially more data before approaching the 4.5 MB secure storage allocation. Storage I/O performance achieved 23.4 MB/s read and 18.7 MB/s write—adequate for credential operations occurring infrequently.

DOI: 10.46121/pspc.52.2.16

DISCUSSION

The experimental results validate that TEE-based security architectures provide robust protection for broadband gateways while imposing acceptable performance overhead. The complete failure of credential extraction attacks even under full normal world compromise demonstrates the fundamental security improvement TEEs offer compared to software-only approaches. This protection extends to realistic threat scenarios including sophisticated remote exploits, local privilege escalation chains, and physical access attacks.

The performance impacts, while measurable, remain modest enough for practical deployment in most gateway applications. The 5% throughput reduction falls well within normal variability and should prove imperceptible in typical usage. Latency increases concentrate in connection establishment phases rather than ongoing packet processing, minimizing impact on user experience. The boot time increase, while substantial in relative terms, remains acceptable in absolute terms given that gateways rarely require cold reboots.

Several findings warrant deeper examination. The 33% latency increase during initial connection establishment, while acceptable overall, could impact latency-sensitive applications like gaming or real-time communication. Optimization opportunities exist through session token caching and predictive session establishment for known clients. Our preliminary implementation of client session caching reduced average latency overhead from 0.6ms to 0.1ms for repeat connections, suggesting production implementations could achieve even better performance.

The cryptographic operation overhead varies significantly by operation type. Symmetric operations like AES show measurable overhead from world switching and less-optimized secure world implementations. However, asymmetric operations like RSA verification show minimal overhead because computation time dominates switching costs. This pattern suggests optimization strategies should focus on minimizing world switches for frequent symmetric operations while accepting overhead for infrequent asymmetric operations.

EasyMesh onboarding latency deserves consideration given user experience impacts. The 1.3-second increase, while modest, occurred in a user-facing operation. However, user studies indicate typical onboarding workflows involve physical device positioning, label reading, and button presses totaling 15-30 seconds. The TEE-induced delay represents less than 10% of total workflow time and proved imperceptible in user testing. Future optimization could reduce this through parallel processing of cryptographic operations and optimized key derivation.

The security testing revealed important limitations alongside successes. Physical side-channel attacks using power analysis showed potential information leakage during cryptographic operations. While these attacks require specialized equipment and physical access, they represent a genuine vulnerability in high-security deployments. Countermeasures including random delays, dummy operations, and hardware-based power filtering could address these attacks but would increase costs and complexity beyond typical consumer gateway requirements.

The failure to protect against malicious hardware peripherals highlights another architectural limitation. An attacker with physical access could install a malicious peripheral device that intercepts memory buses or injects code during boot. While TEEs protect software security boundaries, they cannot fully defend against hardware-level attacks. This limitation affects all TEE implementations and requires complementary physical security measures in high-risk deployments.

Integration challenges emerged during implementation that merit discussion. The OP-TEE software stack required approximately 18,000 lines of custom code including trusted applications, normal world client libraries, and integration with existing gateway services. This development effort, while substantial, remains manageable for gateway vendors with embedded development expertise. The open-source nature of OP-TEE and ARM Trusted Firmware significantly reduced effort compared to proprietary TEE implementations.

Maintainability presents ongoing challenges. Security updates require coordinated updates across normal world Linux, secure world OP-TEE, and potentially ARM Trusted Firmware components. The secure boot chain complicates updates since each component signature verification depends on previous stages. We developed update procedures that maintain security while allowing field upgrades, but vendors must carefully design update processes to avoid bricking devices through verification failures.

DOI: 10.46121/pspc.52.2.16

The research focused on ARM TrustZone platforms due to widespread deployment in gateway processors. However, many existing gateways use MIPS or older ARM cores lacking TrustZone support. Retrofitting security into these platforms requires alternative approaches like hypervisor-based isolation or software-only compartmentalization with reduced security guarantees. The industry shift toward ARMv8-A processors with full TrustZone support should enable broader TEE deployment over time.

Scalability to more complex gateway deployments requires consideration. Our testing covered scenarios with dozens of wireless clients and relatively simple configurations. Enterprise gateways might support hundreds of concurrent VPN sessions, complex routing policies, and integration with authentication infrastructure. The trusted application design accommodates scaling through efficient data structures and caching, but extremely high-scale deployments may require architecture modifications to maintain performance.

The EasyMesh security implementation demonstrates TEE applicability beyond traditional gateway functions. Mesh networking's distributed trust model particularly benefits from TEE protection since compromised agents could undermine entire mesh networks. Our architecture's credential isolation and secure onboarding prevent single-device compromise from cascading to mesh-wide security failures. This pattern generalizes to other distributed protocols like SD-WAN or multi-access edge computing where inter-device authentication and credential protection prove critical.

Future research directions include exploring formal verification of trusted applications to mathematically prove security properties. The relatively small code size of individual TAs makes formal verification tractable compared to full system verification. Proving correctness of authentication protocols and cryptographic implementations would significantly strengthen security assurances beyond testing-based validation.

Machine learning-based anomaly detection running in the secure world represents another promising direction. TEEs could monitor normal world behavior for anomalous patterns indicating compromise while remaining protected from attacker manipulation. This approach would combine traditional signature-based security with behavioral detection while maintaining protection even under normal world compromise.

CONCLUSION

This research developed and validated a comprehensive TEE-based security architecture for broadband gateways addressing critical vulnerabilities in traditional Linux-only implementations. Our OP-TEE integration on ARM TrustZone platforms established hardware-backed security foundations through secure boot chains, trusted firmware, and cryptographic isolation for sensitive networking operations.

The architecture successfully defended against all tested credential extraction attacks including remote network exploits, local privilege escalation, and many physical access scenarios. Complete normal world compromise through kernel rootkits could not extract credentials stored in secure world or bypass firmware signature verification. This represents fundamental security improvement over software-only approaches where any kernel vulnerability compromises all security mechanisms simultaneously.

Performance impact remained within acceptable bounds for gateway deployments, with network throughput reduction under 5%, latency increases of 0.6ms worst-case, and boot time increases of 4.4 seconds. These overheads prove negligible in typical gateway usage while providing strong security guarantees. Optimization opportunities exist for further performance improvement through session caching and parallel verification.

The implementation provides practical guidance for gateway vendors seeking to enhance security through TEE integration. Our trusted application designs for credential management, EasyMesh security, and firmware verification address real-world gateway security challenges. The development effort, while substantial, remains manageable for vendors with embedded expertise.

Key contributions include architectural patterns for secure gateway design, detailed OP-TEE integration procedures for networking equipment, security validation demonstrating protection against realistic attack scenarios, and performance characterization quantifying TEE overhead in networking contexts.

Limitations include vulnerability to advanced physical attacks like side-channel analysis, inability to protect against malicious hardware peripherals, and implementation complexity requiring specialized expertise. Future work should

DOI: 10.46121/pspc.52.2.16

explore formal verification of trusted applications, ML-based anomaly detection in secure world, and extensions to other distributed networking protocols.

As gateway attack surfaces expand through increased functionality and connectivity, hardware-backed security becomes essential rather than optional. The TEE-based architecture developed here provides a foundation for secure gateway design that vendors can adapt to diverse platforms and requirements, significantly improving the security posture of critical network infrastructure.

REFERENCES

1. Anderson, K., Roberts, J. and Wilson, T. (2021) 'Security vulnerabilities in consumer broadband gateways: A comprehensive analysis', *IEEE Security & Privacy*, 19(3), pp. 34-43.
2. Chen, L. and Martinez, A. (2020) 'Kernel exploitation in embedded Linux systems: Techniques and countermeasures', *ACM Transactions on Embedded Computing Systems*, 19(4), pp. 1-28.
3. Garcia, R., Kumar, P. and Sullivan, B. (2019) 'ARM TrustZone: Architecture, applications, and security analysis', *Computer Security Journal*, 35(2), pp. 156-178.
4. Harrison, D. and Taylor, N. (2021) 'Wi-Fi EasyMesh security: Vulnerabilities and mitigation strategies', *IEEE Wireless Communications*, 28(5), pp. 89-96.
5. Kumar, P. and Singh, R. (2021) 'ARM Trusted Firmware: Implementation and security evaluation', *ACM Computing Surveys*, 54(7), pp. 1-32.
6. Morrison, T. and Chen, L. (2019) 'Secure boot implementations for embedded systems: A comparative study', *Journal of Systems Architecture*, 98, pp. 234-251.
7. Park, J. and Liu, M. (2022) 'Performance overhead of trusted execution environments in network equipment', *IEEE/ACM Transactions on Networking*, 30(2), pp. 876-892.
8. Patel, R., Anderson, P. and Davies, M. (2020) 'Measured boot and remote attestation for embedded devices', *Computer Networks*, 178, 107345.
9. Rahman, M. and Anderson, K. (2020) 'Intel SGX versus ARM TrustZone: A comparative analysis for IoT security', *IEEE Internet of Things Journal*, 7(8), pp. 7339-7352.
10. Roberts, J., Wilson, T. and Garcia, R. (2019) 'IoT botnet evolution: From Mirai to modern threats', *ACM SIGSAC Conference on Computer and Communications Security*, pp. 1847-1862.
11. Sullivan, B. and Morrison, T. (2021) 'Cryptographic key management in embedded network devices', *IEEE Transactions on Information Forensics and Security*, 16, pp. 2456-2471.
12. Thompson, K., Park, J. and Williams, R. (2020) 'OP-TEE: Open-source trusted execution environment for embedded systems', *Embedded Systems Letters*, 12(4), pp. 112-115.
13. Wilson, T. and Davies, M. (2018) 'Hardware security modules for embedded applications: Requirements and implementations', *Journal of Hardware and Systems Security*, 2(3), pp. 234-247.

DOI: 10.46121/pspc.52.2.16