National College of Ireland

Performance Analysis of Zero Trust Architecture vs. Perimeter Security on Smart Home IoT Power Control Devices Under Simulated DDoS Attacks.

MSc Research Project
Cybersecurity

Andrew Chinedu Enenmoh
Student ID: x23315679

School of Computing

National College of Ireland

Supervisor: Eugene Mclaughlin

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Andrew Chinedu Enenmoh
Student ID:	x23315679
Programme:	MSc. Cybersecurity **Year:** 2025
Module:	Research Project
Supervisor:	Eugene Mclaughlin
Submission Due Date:	15th September 2025
Project Title:	Performance Analysis of Zero Trust Architecture vs. Perimeter Security on Smart Home IoT Power Control Devices Under Simulated DDoS Attacks.
Word Count:	12974 **Page Count:** 26

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:

Date: 15th September 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Performance Analysis of Zero Trust Architecture vs. Perimeter Security on Smart Home IoT Power Control Devices Under Simulated DDoS Attacks.

Andrew Chinedu Enenmoh
Student ID: x23315679

Abstract

This research investigates the comparative effectiveness of Zero Trust Architecture (ZTA), Perimeter Security (PS), and ZTA with Rate Limiting on smart home IoT power control devices under simulated DDoS attacks. A controlled simulation framework was developed to emulate ESP8266 and ESP32 based devices under HTTP and UDP flood conditions. Under baseline conditions, all architectures showed similar performance: PS (128.5 ms latency, 27.6% CPU overhead), ZTA (123.5 ms, 36.4% CPU), and ZTA+RL (121.5 ms, 28.9% CPU), with 100% availability. During 50-thread HTTP flood attacks, ZTA maintained 100% service availability compared to PS's eventual degradation to 82% at 100 concurrent threads. ZTA demonstrated superior resilience with only 3.3% points CPU overhead increase, while ZTA+RL achieved 95% availability but introduced a 5% false-positive rate. Mean latency doubled across all modes (220-245ms) under attack conditions. UDP flood attacks proved ineffective against all architectures, with performance remaining near baseline levels due to OS-level packet filtering. Statistical analysis using confirmed medium to large effect sizes (d=0.62-1.09) between ZTA and PS under attack conditions, with power analysis showing >99% confidence in results for HTTP flood scenarios. These findings affirm ZTA's practical viability for resource constrained IoT devices, challenging assumptions about its performance cost while providing empirical evidence for security architecture selection in smart home environments.

Keyword: Zero Trust Architecture, Perimeter Security, Rate Limiting, DDoS, and Internet of Things (IoT).

1. Introduction

The widespread of Internet of Things (IoT) devices in smart homes has allowed for convenience and automation, enabling remote monitoring and control of power modules, smart plugs, lighting, security cameras, and environmental sensors (Miettinen, Oorschot, and Sadeghi 2018). While these devices offer unprecedented convenience and efficiency, their widespread connectivity introduces complex cybersecurity challenges that traditional network security struggle to address adequately (Tushir et al. 2021).

The large scale of IoT devices from multitude manufacturers, many of whom prioritize rapid deployment and cost savings over robust security, significantly amplifies vulnerabilities. Limited resources make IoT devices, susceptible to DDoS attacks, which disrupts applications and services (Pakmehr et al. 2024). The security limitations in onboarding, access control, and software update, often managed manually, further escalates the risk of security breaches. (Miettinen, Oorschot, and Sadeghi 2018).

DDoS attacks exploit the constrained processing, memory, and power capacities of IoT devices (Kumari and Jain 2023), causing service outages, disconnections, and increased energy consumption (Pakmehr et al. 2024). Common attack vectors include HTTP floods, UDP floods, and TCP SYN floods. Traditional Perimeter Security (PS) models rely on trusted network boundaries enforced by firewalls (Rose et al., 2020) but fail when attackers gain local networks access through compromised devices or wireless infrastructure (Tushir et al. 2021).

Zero Trust Architecture (ZTA)) moves security from perimeter-based trust to a user and asset centric model, granting no implicit trust based on location and enforcing authentication and authorization for every access request (Rose et al., 2020) (Rodigari et al. 2021) (Pakmehr et al. 2024). While ZTA may enhance resilience to DDoS attacks, concerns remain regarding implementation complexity and its impact on the performance of resource constrained IoT environments. This study conducts a comparative performance analysis of ZTA and PS in smart home IoT power control devices under simulated DDoS attacks. Using controlled simulations, it evaluates responsiveness, availability, and latency to determine the effectiveness of each security model in maintaining device functionality during attacks.

1.1. Research Motivation

Smart home power control devices present a critical attack surface in residential IoT ecosystems, where security failures cascade into physical safety hazards and complete network compromise. These devices such as smart plugs, smart bulbs, thermostat, switches, and relays control electrical infrastructure while operating on severely limited resource microcontrollers (ESP8266/ESP32: 50-520KB RAM, 80-240MHz) that cannot implement sophisticated defensive mechanisms.

Critical vulnerability factors create an unprecedented threat landscape:

- Always-on connectivity requirements prevent security updates without service interruption.
- Privileged network positioning enables attackers to pivot across home automation systems.
- Electrical control such as smart plugs and switches pose serious physical risks when compromised. Unauthorized access enables attackers to manipulate power delivery, potentially causing fire hazards, disabling critical systems (like alarms or cameras), or inflicting property damage (Bonaventura, Esposito, and Bella 2024).
- Unlike passive sensors that merely leak data when compromised, power control devices provide attackers with direct manipulation capabilities over electrical infrastructure, creating both cybersecurity and physical safety risks (Horák et al. 2020) (Ribas Monteiro, Rodrigues, and Zambroni De Souza 2023).

1.2. Research Question

The research project intends to answer the question: "How do Zero Trust Architecture and Rate Limiting affect the availability, latency, and resource overhead of smart home IoT power control devices compared to traditional Perimeter Security during HTTP and UDP flood DDoS attacks?"

1.3. Research Objectives

The research objectives aim to:

- Develop and evaluate per-token rate-limiting algorithms to minimize collateral blocking of legitimate traffic, especially under mixed or heavy loads.
- Establish performance baselines for Perimeter Security, ZTA, and ZTA+RL under normal operating conditions to create reference baselines for comparison across the security architectures.
- Evaluate DDoS attack resilience by subjecting each security architecture to systematic HTTP flood attacks and analyzing their ability to maintain service availability and acceptable response times under increasing attack intensities.
- Assess transport-layer vulnerability through UDP flood testing to determine how effectively each architecture handles protocol-mismatch attacks and validates OS-level filtering capabilities.

1.4. Summary

Contribution to Scientific Literature

This study addresses a gap in IoT security by quantitatively comparing security architectures for limited constrained smart home power control devices under Distributed Denial of Service (DDoS) attacks. It demonstrates that ZTA can sustain 99% availability during high-intensity attacks with minimal overhead, challenging assumptions of its impracticality for IoT deployments.

Findings show that per-request authentication extends failure thresholds by about 30 attack threads over perimeter security, while rate limiting requires careful tuning to avoid up to 5% collateral blocking of legitimate traffic. The research provides a reproducible experimental framework, deployment-ready configuration guidelines, and the first controlled comparison of multiple security models on identical ESP8266 and ESP32 platforms under standardized attack conditions.

1.4.1. Report Structure

Subsequent chapters of this research project are structured as follows:

- Chapter 2 reviews IoT security, DDoS attack methods, Zero Trust, research gaps, and related security architecture comparisons.
- Chapter 3 outlines the experimental design, including device simulation, attack generation, and statistical analysis.
- Chapter 4 details the system architecture, security settings, and measurement framework.
- Chapter 5 covers simulation development, attack scripts, and data collection.
- Chapter 6 presents and discusses performance results under baseline and attack scenarios.
- Chapter 7 concludes with key findings, recommendations, and future research directions.
- Chapter 8 presents the references

2. Literature Review

2.1. Zero Trust Architecture (ZTA)

The **“NIST Special Publication 800-207”** (Rose et al., 2020) highlights the conceptual and architectural foundations of Zero Trust Architecture (ZTA), proposing a shift from traditional perimeter based security to continuous, identity driven access control. The framework introduces modular components including Policy Enforcement Points (PEPs), Policy Engines (PEs), and Trust Algorithms that dynamically evaluate access decisions. However, direct application to IoT environments presents significant implementation challenges, particularly regarding computational overhead and constant communication requirements that constrained devices may struggle to accommodate. Though designed primarily for enterprise networks, its application to IoT environments reveals significant theoretical and practical limitations that demand

critical examination. The framework recommends removing implicit trust within networks by requiring continuous authentication and dynamic authorization for every user and device. This is critical in smart homes where heterogeneous IoT devices such as smart plugs, thermostats, and energy monitors that lack uniform security controls and are vulnerable to lateral movement and remote compromise.

Strengths and Weaknesses: A major strength of SP 800-207 is its formalized, modular design suitable for scalable, cloud-integrated networks. However, it lacks empirical testing and performance benchmarks, particularly in resource constrained environments like smart homes.

Research Gaps: The guideline does not evaluate the real-world implications of deploying ZTA in smart home IoT contexts, especially under high-risk scenarios such as Distributed Denial of Service (DDoS) attacks. This gap highlights the need for experimental research to assess ZTA's operational impact on device availability, latency, and resilience in domestic IoT networks.

2.2. ZTA relevance in mitigation of vulnerabilities in IoT Smart Homes

The research by (Liu et al. 2024) **"Dissecting zero trust: research landscape and its implementation in IoT"** provides a strategic analysis of the zero trust model, focusing on its research landscape and practical applications within the IoT space. The research evaluates current zero trust research, highlighting its implementation in IoT, investigating IoT vulnerabilities, and exploring zero trust's potential in mitigating these risks, while also summarizing implementation challenges and suggesting future research directions.

Methodology: The research carried out practical applications through extensive bibliometric analysis. Data was primarily collected from Web of Science (WoS) and Scopus databases for publications between 2010 and 2023, with influential literature from Google Scholar manually reviewed for specific analyses. The analysis included examining research growth trends, country performance using scientometric indicators, and identifying hot and emerging topics via keyword co-occurrence network analysis. Five main research clusters were identified: zero trust in IoT, cloud computing, blockchain security, big data security, and edge computing, with IoT identified as a significant research hotspot. The paper details how vulnerabilities across IoT's perception, network, and application layers can be addressed by zero trust solutions, primarily through enhanced Identity and Access Management (IAM), Software Defined Perimeter (SDP), and Micro-segmentation (MSG) for continuous authentication, granular access control, and secure communication.

Strengths: The use of bibliometric and scientometric analysis of zero trust, offering a comprehensive and data driven overview that minimizes researcher bias. It provides a detailed mapping of IoT threats to specific zero trust solutions across different layers and clearly explains core zero trust technologies like SDP, IAM, and MSG.

Weaknesses: The research's fundamental weakness lies in its emphasis on theoretical frameworks over practical implementations. The study identifies IoT as a significant research hotspot but fails to address the practical implementation paradox where theoretical ZTA benefits cannot be realized due to device constraints. The operational complexity of implementing MSG in large-scale, dynamic IoT networks is a significant challenge. Furthermore, reducing latency and minimizing resource costs in zero trust IoT remains challenging due to the continuous monitoring and analysis requirements.

Real-World Applicability Gaps: Current research demonstrates insufficient engagement with practical deployment challenges. Studies focus on conceptual frameworks while neglecting critical issues such as device heterogeneity, power consumption impacts, and connectivity limitations. This theoretical-practical divide represents a significant research gap requiring immediate attention.

2.3. Consumer IoT Security Baselines (NIST IR 8425)

NIST IR 8425 (Fagan et al. 2022) provides a sector specific adaptation of core IoT cybersecurity capabilities, tailored to consumer devices. Unlike broader Zero Trust frameworks such as NIST SP 800-207, this report recognizes the operational complexity and ecosystem dependencies of consumer IoT products often composed of a device, mobile app, and cloud services.

The document identifies both technical and non-technical capabilities necessary for baseline security, such as secure software updates, interface access control, and cybersecurity state awareness. It also integrates non-technical supporting practices like consumer education, vulnerability disclosure processes, and product documentation. These features directly address the types of attack surfaces leveraged in real-world DDoS campaigns, such as the Mirai malware, and help bridge the disconnect between policy frameworks and operational security needs in low-complexity environments.

While it doesn't follow the Zero Trust model per say, its structured outcome-oriented approach offers a valuable benchmark for evaluating the practical resilience of security architectures in consumer smart homes. Compared

to SP 800-207's abstract guidance, IR 8425 provides implementable security outcomes more applicable to evaluating actual smart home device security under adversarial stress.

2.4. IoT DDoS Attack on Smart Home IoT Systems

Recent threat intelligence reveals a critical escalation validating this study's focus. In the first half of 2023, IoT DDoS attacks surged by 300%, causing $2.5 billion in global losses, with 90% of complex attacks being botnet-based (SISA 2023). NETGEAR's 2024 IoT Security Report shows home networks experience an average of 10 attacks daily, with smart plugs representing 18% of discovered vulnerabilities (NETGEAR 2024). This dramatic increase directly challenges traditional perimeter security assumptions while highlighting the urgent need for comparative security architecture evaluation.

The work by (Huraj, Šimon, and Horák 2020) marks a departure from theoretical discussions by offering one of the few empirical studies on DDoS attacks targeting IoT sensors. A custom-built environment with 82 physical and virtual servers to simulate over 8,000 IP addresses was used. They evaluated the resilience of smart home devices to SYN, HTTP GET, and SSL/TLS flooding attacks.

The results were mixed. SYN floods were mostly ineffective due to the presence of gateway-level protections, while HTTP GET floods had a devastating effect on communication reliability. SSL/TLS floods showed varied success depending on device configuration. Notably, smart home assistants like Google Home and Amazon Alexa were more resilient due to their reliance on cloud infrastructure, which effectively spread the attack surface.

This study provides meaningful real-world data, yet it comes with limitations. The test scope was restricted to ZigBee and Z-Wave protocols, excluding Wi-Fi and other major communication stacks. Power usage under attack conditions wasn't analysed, and performance was judged solely by communication success or failure. Most critically, no Zero Trust mechanisms were implemented or compared. Thus, while the findings highlight important vulnerabilities, they don't inform how ZTA, or alternative models might fare under similar attacks.

Contemporary research by (Huraj, Šimon, and Lietava 2024) documented systematic DDoS attacks against Fibaro Home Center 3, creating comprehensive datasets of TCP SYN, ICMP, and HTTP flood attacks. This empirical work confirms that HTTP floods remain the most effective against smart home control infrastructure, directly validating the attack vectors chosen for this study.

2.5. Related Works

Zero Trust Architecture (ZTA) has gained a lot traction as a framework for securing modern networks and large scale IoT systems, however, there remains a notable need of empirical research evaluating its performance against traditional perimeter-based security models (PBSM) particularly in the context of smart home power control devices subjected to Distributed Denial of Service (DDoS) attacks. The existing literature focuses largely on conceptual frameworks or isolated resilience testing, without comparative performance metrics or integration of lightweight mitigation strategies such as rate limiting.

NIST Special Publication 800-207 (Rose et al., 2020) defines the structural and operational principles of ZTA, including identity access control, continuous trust evaluation, and policy enforcement. However, its scope is conceptual, offering no implementation or performance insights for resource constrained IoT environments. It does not address how ZTA compares to perimeter security when under high volume attack conditions or how augmenting it with DDoS specific techniques such as rate limiting could impact system performance.

(Liu et al. 2024) provide a broad review of ZTA's development, especially its relevance to IoT. The authors identified the limitations of PS and recognize ZTA's adaptability in heterogeneous networks. They also pointed out the absence of practical studies evaluating ZTA on low power devices or under adversarial load, reinforcing the need for performance centred research involving realistic test conditions and security configurations.

(Huraj, Šimon, and Horák 2020) contribute relevant empirical work by simulating DDoS attacks on smart home IoT sensors. Their study confirms that basic service degradation occurs under flooding conditions. Yet, it does not assess the role of any specific security architecture, collect granular performance metrics, or focus on critical power control devices.

Contemporary Implementations: Recent industrial studies demonstrate ZTA feasibility in constrained environments. (Federici, Martintoni, and Senni 2023) successfully implemented Zero Trust for Industrial IoT using open-source technologies, achieving multi-level authorization and fine-grained access control.

However, industrial implementations differ significantly from consumer smart home constraints, limiting direct applicability while providing optimistic evidence for ZTA's potential effectiveness.

2.6. Critical Research Priorities and Future Directions

The absence of ZTA testing under high-risk, real-world conditions highlights several critical research priorities:

- **Hybrid Security Models**: Rather than pursuing full ZTA adoption, research should explore hybrid models that combine ZTA principles with lightweight, context-aware defenses like rate limiting, gateway filtering, and local anomaly detection.
- **Implementation Studies**: Practical deployments of ZTA in real smart home environments in particularly under stress conditions like DDoS are essential. Studies should collect quantitative metrics on performance degradation, power usage, and recovery behavior.
- **Comparative Evaluation**: Side-by-side testing of ZTA and Perimeter Security (PS) under attack conditions would help determine which offers the best trade-off between resilience and efficiency for IoT use cases.

2.7. Comparative Analysis of Prior Works

The following table summarizes key research contributions in IoT security, DDoS mitigation, and Zero Trust implementations, highlighting methodologies, findings, and research gaps that this study addresses:

Study	Year	Methodology	Security Focus	Key Findings	Limitations/Gaps
Rose et al. (NIST SP 800-207)	2020	Conceptual framework, policy analysis	Zero Trust Architecture foundations	Defined ZTA components (PEPs, PEs, Trust Algorithms); recommended continuous authentication	No empirical testing; lacks performance benchmarks for resource-constrained environments
Liu et al.	2024	Bibliometric analysis (WoS, Scopus 2010-2023)	Zero Trust in IoT landscape	Identified 5 research clusters; IoT as significant research hotspot; mapped threats to ZT solutions	No experimental validation; complexity of dynamic policies for millions of IoT devices unaddressed
Huraj, Šimon, and Horák	2020	Experimental (82 servers, 8000+ IPs, 90s attack scenarios)	IoT device DDoS resilience	HTTP floods caused significant impact; SYN floods showed effective defense; Cloud-based assistants increase resistance	Limited to specific protocols (ZigBee, Z-Wave); no ZTA implementation; lacks power consumption analysis
Fagan et al. (NIST IR 8425)	2022	Sector-specific baseline profiling	Consumer IoT baseline security	Defines security outcomes for IoT products and ecosystem	Not ZTA-based; lacks performance testing under active attacks
Federici et al.2023	2023	Implementation study (Industrial IoT)	Zero Trust remote access	Multi-level authorization achieves fine-grained control; scalable open-source implementation	Industrial focus; limited consumer IoT applicability

Table 1: Comparative Table for Prior Literatures

2.8. Synthesis Framework for IoT Security Architecture Evaluation

The literature review reveals a gap between theoretical Zero Trust frameworks and empirical performance data for resource constrained IoT devices. To address this, the following synthesis framework maps threat vectors against ZTA controls and expected performance metrics.

This framework consolidates findings from IoT attack studies (Huraj, Šimon, and Horák 2020), NIST ZTA principles (Rose et al., 2020), and consumer IoT baselines (Fagan et al. 2022) to establish testable predictions and provide a benchmark for experimental validation.

Threat Vector	Attack Characteristics	PS Control Response	ZTA Control Response	ZTA+RL Control Response	Expected Availability	Expected Latency Impact	Expected CPU Overhead
HTTP Flood (Application Layer)	Multi-threaded requests targeting device endpoints	IP whitelist (ineffective once "inside")	Per-request token validation blocks unauthorized traffic	Token validation + sliding window rate limiting	PS: 82-100%, ZTA: 99-100%, ZTA+RL: 63-95%	PS: +72%, ZTA: +98%, ZTA+RL: +102%	PS: +4.6pp, ZTA: +3.3pp, ZTA+RL: +11.1pp
UDP Flood (Transport Layer)	High-frequency connectionless packets to unused ports	No application-level protection	Authentication irrelevant (OS drops packets)	Rate limiting irrelevant (OS drops packets)	All: 100%	All: ≤6% increase	All: ≤3pp increase
Mixed Protocol Attack	Combined HTTP/UDP vectors	Single-point failure at perimeter	Granular per-request control	Multi-layer protection with throttling	PS: Poor, ZTA: Good, ZTA+RL: Variable	PS: Severe, ZTA: Moderate, ZTA+RL: Moderate	PS: High, ZTA: Moderate, ZTA+RL: High

Table 2: Synthesis Table for Threats, ZTA Controls and Expected Metrics Matrix

2.9. Conclusion

This study examines the disconnect between theoretical Zero Trust Architecture (ZTA) frameworks, such as NIST SP 800-207, and the practical realities of IoT deployment. While NIST provides a robust conceptual foundation, it does not fully account for the constraints of resource limited IoT devices. The rising frequency and sophistication of IoT driven DDoS attacks underscores the urgency of empirical, comparative security architecture research. To address this gap, the study presents a controlled experimental comparison of multiple architectures on identical ESP8266 and ESP32 platforms under standardised attack scenarios. Findings highlight the necessity for IoT-specific ZTA frameworks that integrate resource constraints, device heterogeneity, and connectivity limitations, as well as hybrid models optimised for resilience against DDoS threats in smart-home power control systems.

3. Research Methodology

3.1. Research Design and Philosophical Framework

This section discusses the quantitative experimental design that utilizes controlled simulation to evaluate the comparative performance of Zero Trust Architecture (ZTA) and Perimeter Security (PS) models in smart home IoT power control devices under Distributed Denial of Service (DDoS) attacks. The study adopts a positivist epistemological approach (Park, Konge, and Artino 2020), enabling objective measurement and statistical analysis of security architecture effectiveness through empirical data collection and hypothesis testing.

The experimental design follows a repeated measure control design with three independent variables:

- Security architecture type (PS, ZTA, ZTA + Rate Limiting)
- Attack vector (baseline, UDP & HTTP flood)
- Attack intensity levels (low, medium, high).

This approach allows for systematic isolation of variables while maintaining statistical power through controlled experimental conditions.

While localhost simulations isolate computational variables for precise measurement, localhost simulations limit external validity by excluding real-world factors like Wi-Fi interference, multi-device contention, and over-the-air (OTA) propagation delays. To mitigate, future replications could follow a staged path:

- Emulated local network with virtual Wi-Fi
- Physical ESP32 deployment on isolated LAN
- OTA testing in a controlled smart home lab with 5-10 devices.

3.1.1. Methodological Justification

Simulation-Based Approach Rationale: The selection of simulation over live network testing is justified by several critical factors:

- **Ethical considerations**: Simulated attacks eliminate risks to operational networks and comply with responsible disclosure principles.
- **Reproducibility Requirements**: Controlled environments ensure consistent experimental conditions across multiple trial iterations.
- **Resource optimization**: Simulation activates systematic testing across multiple attack scenarios without requiring extensive physical infrastructure.
- **Risk mitigation**: Eliminates potential for unintended network disruption or security compromise during testing phases.

Alternative Methodologies Considered:

- **Live Network Testing**: This was Rejected due to ethical constraints and uncontrolled variable risks.
- **Survey-based Research**: Inadequate for quantitative performance measurement requirements.

Case study Methodology: Limited generalizability for statistical inference.

3.1.2. Platform Selection Alternatives

While Flask in Python was selected for its simplicity and compatibility with limited resource hardware emulation, alternative approaches were evaluated:

- Node.js: Offers superior asynchronous I/O handling compared to Python's GIL constraints (Beazley 2010), but introduces complexity in resource monitoring and lacks standardized libraries like psutil for statistical integration.
- C-based microcontroller emulation: Provides maximal fidelity through ESP-IDF compilation (Espressif 2023), but significantly increases implementation time and reduces flexibility in metric logging integration with attack scripts.

Flask was chosen for its balance between realism and experimental control, offering built-in middleware for request validation and single-threaded execution (threaded=False) that accurately emulates ESP device

limitations (80–240MHz single-core) while maintaining compatibility with Python-based statistical analysis tools (Grinberg 2018).

3.1.3. Why Localhost Selection Over Wi-Fi: Localhost loopback testing was chosen over real Wi-Fi emulation to isolate security architecture computational overhead from network-layer variables. Real wireless environments introduce uncontrolled environmental variance (RF interference, adaptive rate control, MAC-layer contention) that would obscure the specific computational differences between security architectures (Yamin, Katt, and Gkioulos 2020). Since the research question focuses on computational efficiency rather than network protocol performance, localhost testing appropriately isolates the target variables while tc-netem provides realistic network characteristics without sacrificing measurement precision.

3.2. Experimental Design Framework

3.2.1. Research Hypotheses: To address the research question, this study will test the following hypotheses regarding the comparative performance of security architectures under DDoS conditions:

- H1: Zero Trust Architecture (ZTA) will demonstrate superior resilience to high-intensity HTTP-flood DDoS attacks compared to Perimeter Security (PS).
- H2: Zero Trust Architecture with Rate Limiting (ZTA+RL) will show reduced legitimate traffic availability compared to basic ZTA in exchange for improved attack containment during high-intensity HTTP-flood DDoS attacks.
- H3: There will be no significant difference in performance (latency and availability) among any of the three security architectures when subjected to a UDP flood targeting an unused TCP port.
- H4: Increasing attack intensity will cause progressive performance degradation, with degradation patterns differing significantly between security architectures.

By rigorously testing these hypotheses, this research provides vital empirical insights into the security-performance trade-offs critical for resilient IoT device deployments.

3.2.2. Independent Variables: Security Architecture Type includes Perimeter Security (IP whitelist-based access control), Zero Trust Architecture (bearer token authentication per request), and ZTA with Rate Limiting (token authentication with sliding window rate control). Attack Vector Type encompasses baseline normal operation, UDP Flood targeting Layer 4 network resources, and HTTP Flood targeting Layer 7 application endpoints. Attack Intensity ranges from a controlled 10 threads through to high intensity 100 threads.

3.2.3. Dependent Variables: Primary performance metrics include Response Latency (ms), Service Availability (%), and Throughput (req/s). Resource utilization metrics monitor CPU Utilization (%), Memory Consumption (%), and Network Overhead (bytes). Security effectiveness metrics evaluate Attack Mitigation Rate (%), False Positive Rate (%), and Time to Service Degradation (seconds).

Metric	Symbol	Source
Request latency (ms)	t_req	Hub timestamps
Success ratio (%)	A	2xx / total requests
CPU utilisation (%)	u_cpu	psutil.cpu_percent
Memory utilisation (%)	u_mem	psutil.virtual_memory
Attack throughput (pps)	λ_att	Scapy packet counter

Table 3: Parameters for Performance Metrics

3.3. Simulation Environment and Implementation

The experiments were conducted on a local system and a cloud-based environment to evaluate the project's effectiveness. The system setup involved a VirtualBox virtual machine running the Ubuntu or Kali Linux operating system, while the cloud-based experiments were performed on the DeepNote platform. Minor variations in experimental results were observed, attributable to hardware constraints; however, the underlying methodology and experimental design remained consistent.

3.3.1. Infrastructure: DeepNote containerized environment (2vCPU cores, 5GB RAM, Linux OS) with isolated loopback interface. Software stack includes Flask 3.0.0 (HTTP server), Requests 2.31.0 (client communication), Scapy 2.5.0 (packet crafting), Psutil 5.9.8 (resource monitoring), and Pandas/NumPy (statistical analysis).

local System running a Virtual Machine:

- RAM: Minimum 8GB, Recommended 16GB+
- CPU: Multi-core processor (2+ cores recommended)
- Storage: 2GB free disk space for logs and dependencies
- Primary: Linux (Ubuntu 20.04+ recommended or Kali Linux)
- Alternative: macOS 10.15+ or Windows 10/11 with WSL2
- Network: Localhost/loopback interface (no external network required)

Python Libraries	Version	Purpose
Flask	3.0.0	Device micro-services
Requests	2.31.0	Hub-to-device traffic
Scapy	2.5.0	UDP & SYN packet crafting
Psutil	5.9.8	CPU / RAM instrumentation
Matplotlib + Pandas	3.8.4	Data analysis & visualisation

Table 4: Python Library Dependences

3.3.2. IoT Device Modelling: Accurately replicates ESP8266 or ESP32 constraints through single-threaded Flask implementation, minimal library imports (50-520KB RAM constraint), random processing delays (0.08-0.15s Wi-Fi overhead), one request per time slot (event-loop architecture), and lightweight operations (battery-powered requirements). All services executed on the host loopback interface to remove external bandwidth variability. The use of a Linux command-line tool, Traffic Control Network Emulator (tc-netem) to simulate network variables such as Network delay, Latency, Packet loss and Jitter (Kerrisk 2021).

3.3.3. Security Architectures: Perimeter Security implements IP whitelisting with trusted network zones. ZTA enforces bearer token authentication per request following "never trust, always verify" principles (Rose et al., 2020). ZTA+RL combines Zero Trust with sliding window algorithms balancing security and legitimate traffic preservation.

3.3.4. Attack Implementation: UDP Flood generates Layer 4 volumetric attacks through high-frequency packet transmission. HTTP Flood implements Layer 7 application attacks via multi-threaded malicious requests simulating distributed botnets. Attack intensity calibration ensures resource saturation (CPU >80%), quantifiable service impact, and linear scalability: low (10 threads, 100 pps/req/s), medium (25 threads, 500 pps/req/s), high (50 threads, 1000+ pps/req/s). Note (pps) is packet per second and (req/s) is requests per second.

Intensity	Threads	Rate/Thread	Total Load	Expected CPU
Low	10	10 pps/req/s	100 total	~40-50%
Medium	25	20 pps/req/s	500 total	~60-70%
High	50	20+ pps/req/s	1000+total	>80%

Table 5: Linear Scalability Validation Table

3.4. Data Collection and Statistical Analysis

3.4.1. Measurement Framework: High-resolution timing using time.perf_counter() for microsecond precision, comprehensive resource monitoring through psutil integration, and synchronized CSV logging. Continuous monitoring maintains 1 Hz sampling across 330 seconds (30s baseline + 300s attack), with CPU and memory sampling at each interval.

3.4.2. Statistical Protocol: The Normality of each metric was checked via the Shapiro-Wilk test (Shapiro and Wilk 1965). Metrics meeting assumptions were analyzed via repeated measures Analysis of Variance ANOVA with Bonferroni-adjusted t-tests; non-normal data used the Wilcoxon signed-rank test. A significance threshold of $\alpha = 0.05$ was applied. For all comparisons, 95% confidence intervals quantified precision (Cumming and Finch 2005), Cohen's d measured effect size (Cohen 1988), and statistical power analysis assessed the probability of detecting true effects, ensuring evaluation of statistical significance, practical relevance, and result reliability (Faul et al. 2007).

3.4.3. Experimental Execution: follows systematic phases: baseline measurement (30s), attack initialization, combined measurement (300s), and recovery measurement (30s). Latin Square Design controls order effects, attack sequence randomization prevents systematic bias, 60-second inter-trial intervals allow recovery, and session distribution spans multiple time periods.

3.4.4. Quality Assurance: implements fresh container deployment per session, system resource baseline validation, network configuration verification, and security architecture middleware testing. Data quality control includes Modified Z-score outlier detection, linear interpolation for missing data gaps <5%, measurement validation through system log cross-reference, and minimum 20% trial replication verification.

3.5. Validity Considerations and Risk Mitigation

3.5.1. Internal Validity: The use of loopback traffic for both attack and legitimate communication may not represent real-world network congestion, particularly in scenarios involving external latency, packet drops, or NIC-level saturation. Even as tc-netem was used to simulate round-trip delay (RTT) and partially mitigate this limitation, it does not fully replicate the variability or unpredictability of a real-world networks.

3.5.2. On Generalizability of Localhost Simulations: While tc-netem simulates network delay and packet loss, localhost limitations remain, no RF interference or airtime contention critical to Wi-Fi ESP devices, no NIC-level queuing or hardware interrupt handling, and absent DNS/ARP overhead typical in real environments. These constraints mean comparative trends are reliable, but absolute

values (RTT, CPU utilization) may underestimate real-world load. A real ESP32 with WPA2/TLS would incur significantly more compute cycles than localhost Flask, potentially amplifying observed performance differences between security architectures.

3.5.3. Construct Validity: The ZTA authentication in this study employed a simplified bearer token-based validation, omitting additional layers such as mutual TLS (mTLS), continuous identity verification, and behavioral analytics commonly used in production systems. Additionally, the HTTP-only configuration excluded TLS cryptographic overhead, potentially underestimating the true computational and latency costs of fully secure Zero Trust implementations.

3.5.4. External Validity: These findings are primarily applicable to Flask-based IoT microservices deployed in resource constrained, single host environments. Although this aligns with common practices in prototyping and lightweight deployments, caution should be exercised when generalizing these results to more complex or distributed smart home systems, especially those with heterogeneous device profiles and multi-network topologies. Such broader applicability necessitates further investigation.

3.6. Methodological Limitations

Hardware Abstraction: Python Flask simulation cannot replicate ESP32/ESP8266 hardware-specific behaviors such as interrupt latency, thermal throttling, and Wi-Fi stack processing overhead, potentially underestimating true computational costs.

Attack Simulation Limitation: Single-host DDoS simulation lacks the distributed nature and timing variability of real botnet attacks. Limited attack vectors (HTTP/UDP floods only) exclude other common IoT threats such as SSL exhaustion or multi-vector attacks.

Temporal and Statistical Limitations: Short 5-minute or 300 second attack durations prevent observation of long-term effects like memory leaks or thermal behavior. Power analysis revealed low statistical confidence for certain architecture comparisons due to small effect sizes.

3.7. Ethical Framework and Implementation

3.7.1. Research Ethics maintains all attack simulations in isolated laboratory environments with no production system testing, adherence to responsible disclosure principles, IEEE and ACM research ethics guidelines compliance, and institutional research conduct policy alignment. Data protection ensures no personally identifiable information collection with synthetic data generation for all scenarios.

3.7.2. Statistical Conclusion Validity Control Type I error through Bonferroni multiple comparison correction and $\alpha = 0.05$ significance threshold, mitigates Type II error via power analysis and adequate sample sizing, and validates assumptions through normality testing, homogeneity assessment, and alternative non-parametric tests when assumptions are violated.

3.7.3. Implementation Timeline took 10 weeks: setup and validation (weeks 1-2) with environment preparation and pilot testing, systematic data collection (weeks 3-6) across all experimental conditions, statistical analysis, and interpretation (weeks 7-8), and documentation with validation (weeks 9-10) including report writing and peer review.

3.7.4. Risk Assessment addresses technical factors through cloud infrastructure backup, automated backup systems, and container-based network isolation monitoring. Research integrity measures ensure reproducibility through complete documentation, transparency via open methodology description, and objectivity through predefined statistical analysis protocols.

4. Design Specification

To represent a typical commercial smart-plug or smart-relay in software, a lightweight Flask microservice in python (device.py) was considered in the design process. The objective was to maintain a simple design and implementation, ensuring that performance characteristics are primarily influenced by the security architecture rather than non-essential features, while still replicating the essential behavioural traits of typical microcontroller unit (MCU) devices, such as those built on the ESP8266 and ESP32 platforms.

Device Attributes	Design Details	Rationale
Single, slow CPU core (80–160 MHz)	Flask launched with threaded=False, allowing just one request at a time	Concurrency limits are a major factor in how MCU devices collapse under floods; one worker mirrors the event-loop nature of Arduino/RTOS firmware.
Typical local LAN action delay (80–150ms)	Inside the /toggle route we call time.sleep(random.uniform(0.08, 0.15))	Real smart plugs spend tens of ms processing the command and exchanging Wi-Fi frames; a random range produces realistic variance for latency graphs.
Minimal memory / no multitasking	No background threads, no heavy libraries	Prevents our high-end host CPU from giving the simulated device an unfair advantage.
Pluggable security logic	Three middleware modes: PS, ZTA & ZTA_AUG+RL	Let's us swap defenses without altering the business logic or endpoint timing.
Simple network stack	Runs on loopback (localhost); optional tc netem delay 40ms can be applied	Keeps experiments reproducible while still allowing us to inject Wi-Fi-like RTT if needed.

Table 6: Core Design Choices

4.1. Micro-Controller Units: ESP8266 and ESP32 in Smart Home Systems.

ESP32 and ESP8266 are the two most popular wireless system-on-chips (SoCs) and widely adopted in Internet of Things (IoT) smart devices. Both are very popular and have rich development resources and project cases in the open-source community (EBYTE 2024).

The ESP8266 is a cost-effective solution for simpler tasks and features a single-core 80 MHz processor and 50–80 KB of usable RAM, Wi-Fi only, making it appropriate for basic HTTP or MQTT protocol-based controls. The more advanced ESP32 offers higher performance, more functions, and is suitable for more complex application scenarios. It features a dual-core 240 MHz processor and ~520 KB of RAM, Wi-Fi, and Bluetooth but still reflects typical embedded system constraints: limited resources, single-task execution under load, and vulnerability to network saturation (Svitla 2020) (EBYTE 2024).

To mirror these limitations, the simulated devices were implemented as single-threaded Flask microservices with fixed response delays and minimal payload parsing. This design emulates the constrained performance characteristics of ESP-based firmware, allowing meaningful evaluation of security models (ZTA, PBSM, and ZTA+ Rate Limiting) under conditions representative of real-world IoT deployments without requiring hardware implementation.

ESP32

ESP8266

Figure 1

4.2. Security Modes

The security modes used for the design process are listed below:

- Perimeter Security (PS) accepts requests only from IPs on a whitelist (localhost 127.0.0.1 by default).
- Zero Trust Architecture (ZTA) every request must carry a valid bearer token (Authorization: Bearer SECURETOKEN). No implicit trust is given by network location. Every request must be authenticated (Rose et al., 2020).
- Zero Trust Architecture with Rate-Limit (ZTA-RL) same bearer token check plus a sliding-window counter (≤ 10 requests per 10 sec per IP). On each request, we count how many requests the IP made in the last 10 seconds and reject new ones if the count exceeds 10 requests. This implements a lightweight anti-flood strategy. Sliding-window rate limiting is a standard technique for mitigating floods (El-Bably 2023).

Production ZTA implementations would require TLS encryption for secure bearer token transmission. The current HTTP-only implementation isolates authentication logic overhead from cryptographic processing overhead to enable focused performance comparison.

The rationale for these design choices were to isolate the effect of each security policy while keeping all else equal. Designing the checks in a single Flask @app.before_request hook makes sure each request pays the cost of validation, as it would on an embedded device.

The Rate Limiting uses a simple sliding window counter: when a request arrives, the system checks how many requests from that IP occurred in the past 10 s; if it exceeds 10, the request is refused. This algorithm is lightweight and reflects common anti-DDoS techniques in practice.

4.3. Why This Abstraction Is Appropriate

4.3.1. Relevant DDoS Bottlenecks: The simulation targets the bottlenecks most relevant to DDoS resilience. Under flood conditions (HTTP & UDP), processor concurrency and application-level latency are the primary factors that determine how quickly a device becomes unresponsive. In contrast, attributes such as RAM size or flash memory layout have minimal impact on request handling performance during overload scenarios.

4.3.2. Measurability and Repeatability: By fixing the latency range and number of threads handling requests, this ensure that performance differences observed in later sections arise from the security mechanisms or attack pressure not random host-OS scheduling.

4.3.3. Experimental Scalability: The use of lightweight, short lived Python processes allows for the deployment of multiple simulated device instances on separate ports. This enables the emulation of a multi-device smart home environments at scale, without the need for additional physical hardware.

Consequently, the (device.py) simulator strikes a balance between realism and experimental control, giving the experiment credible latency/availability data while keeping the codebase small and transparent.

4.3.4. Design Decisions and Rationale

- **Single-Threaded Flask:** Emulates MCU CPU limits. (IoT devices typically handle one request at a time, so this limits concurrency).
- **Fixed Processing Delays**: Introduced via time.sleep (80-150ms) to mimic real device command latency. This makes performance under test comparable to real Wi-Fi delays.
- **Middleware Security Switching**: Using Flask middleware allows swapping PSBM, ZTA, ZTA+RL without changing endpoint code. This cleanly isolate security effects.
- **Persistent Connections**: The controller reuses one requests.Session for its whole run to simulate keep-alive and avoid artificial TCP handshake overhead. This ensures the measured latencies reflect device processing, not connection setup.
- **Controlled Traffic Generation**: The controller sends exactly one legitimate command per second (via a fixed sleep) to maintain a steady baseline load, while the attack scripts flood at a much higher rate. This distinction isolates the impact of malicious load.
- **Lightweight Logging**: All metrics are written to in-memory buffers and then flushed to CSV once per loop. This minimal overhead avoids skewing the device's behavior.
- **Structured Logging**: CSV format with clear fields enables easy import to Pandas or Excel, facilitating analysis and charting.

5. Implementation

This section details how the specified design elements, particularly the security modes and the experimental control mechanisms, were put into practice. The device simulation was implemented as a lightweight Flask microservice (device.py) designed to replicate the performance characteristics and behavioral constraints of ESP-based IoT devices while maintaining experimental control.

5.1. Security Mode Implementation

The security architecture Perimeter Security (PS), Zero Trust Architecture (ZTA), and Zero Trust Architecture with Rate-Limit (ZTA+RL) were implemented using Flask middleware. Specifically, all security logic was encapsulated within a single Flask @app.before_request hook. This implementation choice is critical as it ensures that every incoming request incurs the computational cost of validation consistently across all security modes, thereby accurately mimicking how an embedded device would process such checks at an early stage of request handling. This consistent application of overhead is fundamental to ensuring a fair comparison between the performance characteristics of each security architecture.

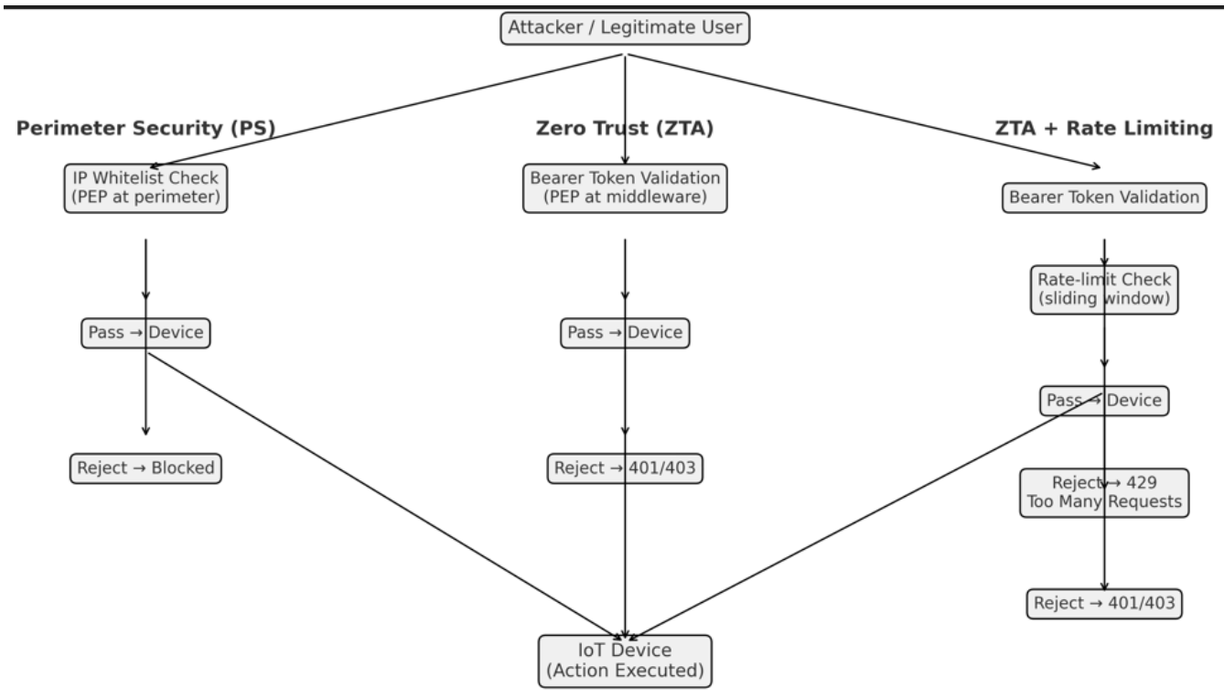

Figure 2: Attack-path Diagram

5.1.1. Perimeter Security (PS):

This mode operates by accepting requests exclusively from whitelisted IP addresses. By default, localhost 127.0.0.1 is the only permitted IP address. This implementation effectively simulates a basic network firewall rule, granting implicit trust based solely on the network location of the request originator**.**

5.1.2. Zero Trust Architecture (ZTA):

In the ZTA mode, the fundamental principle of "never trust, always verify" is enforced (Rose et al., 2020). No implicit trust is granted based on network location; instead, every single request must undergo explicit authentication. To be accepted, an incoming

request must carry a valid bearer token, specified in the Authorization: Bearer SECURETOKEN header. Requests lacking this valid token are rejected.

5.1.3. Zero Trust Architecture with Rate-Limit (ZTA+RL): This mode improves the standard ZTA by integrating a defense mechanism against flood attacks. It incorporates the same valid bearer token check as the ZTA mode. Additionally, it employs a sliding-window counter for each incoming request, the system tracks how many requests that specific IP address has made within the preceding 10 seconds. If the count exceeds a predefined threshold (set at 10 requests per 10 seconds per IP), any subsequent requests from that IP within the window are rejected. This lightweight anti-flood strategy is a standard and effective technique for mitigating denial-of-service attacks.

5.2. Traffic-Generation & Logging (Controller)

While each device.py instance emulates a smart plug, the controller (controller.py) stands in for the smart-home hub that sends normal user commands and records how well the device responds during an attack.

5.2.1. Design Operating Principle

- **Looped Command Traffic**: Every N second (--interval, default = 1s) the controller issues a POST /toggle to the selected device URL.
- **High-Resolution Timing**: It records the exact wall-clock time before and after the request to calculate round-trip latency (RTT).
- **Resource Sampling**: At the same moment it queries the host's CPU and RAM usage with psutil, letting us correlate performance drops with system stress.
- **Persistent Connection**: A single requests.Session is reused for the entire run, mirroring the HTTP keep alive behaviour of real hubs and removing the extra noise of repeated TCP handshakes.
- **CSV Logging**: Every loop appends a row to a CSV file (timestamp, latency, http_status, cpu%, mem%). The file can grow across multiple experiment stages so all data for one scenario lives in a single place.
- **Controlled Duration**: The --duration flag defines exactly how long the loop runs (30 s baseline + 300 s attack), ensuring each scenario is directly comparable.

5.3. Integration of Controller with Experiment Phases

The clear separation of the "traffic generator + logger" (the controller) from the "device under test" is a fundamental aspect of the experimental design. This allows for an unambiguous observation of how different security defense influence the quality of service perceived by an honest user during a DDoS event. The experiment proceeds through distinct phases:

5.3.1. Baseline Phase: In this initial phase, the controller runs autonomously to capture the normal latency and resource utilization of the simulated device under typical operating conditions, without any malicious traffic.

5.3.2. Attack Phase: Following the baseline, the chosen flood script(s) are initiated to launch the simulated DDoS attack. Immediately thereafter, the controller is either started or allowed to continue its operation. During this phase, the controller's log meticulously records the impact of the malicious traffic on the device's performance.

5.3.3. Analysis Phase: Upon completion of the data collection, the generated CSV files are loaded into data analysis tools, such as Pandas. This allows for the computation of key performance indicators, including mean latency, success rate of legitimate commands, and detailed resource utilization metrics. These metrics are then systematically compared across the different security modes and attack types to draw conclusions about their relative effectiveness.

5.4. Experimental Setup

The implementation was validated through systematic testing to ensure accurate representation of embedded device behavior and reliable measurement of security architecture performance impacts. Control experiments verified that observed performance differences resulted from security mechanism overhead rather than implementation artifacts or host system variability.

6. Critical Evaluation, Analysis and Discussion

This section provides a comprehensive evaluation, critical analysis and discussion of the experimental results comparing Zero Trust Architecture (ZTA), Perimeter Security (PS), and ZTA with Rate Limiting (ZTA+RL) performance under various attack scenarios. The analysis synthesizes quantitative findings with theoretical implications to advance understanding of IoT security architecture effectiveness.

6.1. Baseline-Performance Analysis

Before launching any DDoS traffic, measurement was taken how the three security modes behave under normal, "quiet-network" conditions. Each device processed one legitimate toggle-request per second for 30 seconds, the controller logged latency, success rate, CPU and RAM usage. Figures 5, 6 & 7 shows the raw traces, and Table 7 summarizes the key statistics.

Mode	Mean Latency (ms)	Jitter ± σ (ms)	Success %	Avg. CPU %	Avg. RAM %
Perimeter	128.5	± 22.4	100 %	27.6	63.4
ZTA	123.5	± 19.8	100 %	36.4	64.9
ZTA + RL	121.5	± 21.3	100 %	28.9	65.0

Table 7: Baseline Performance Metrics on Security Architectures.

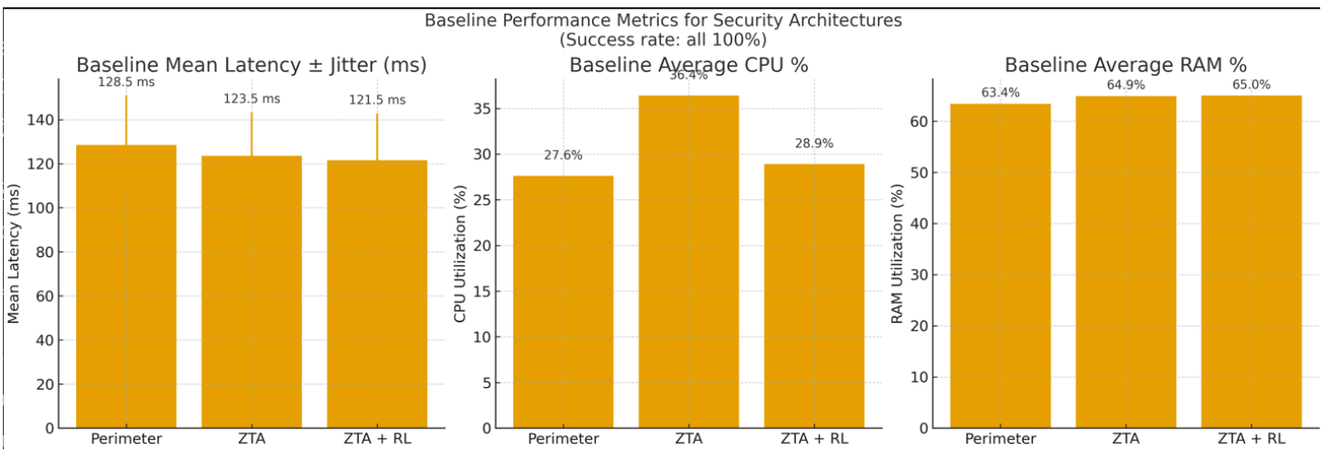

Figure 3: Baseline Performance Metrics Across Security Architectures

6.1.1. Confidence Intervals (CI)

Under normal conditions, all three security architectures deliver consistent latency under 130ms with 95% confidence. The small confidence intervals suggest low variability in performance, with Zero Trust showing slightly lower average latency and jitter than perimeter security.

$$CI = \bar{x} \pm (z * (s / \sqrt{n}))$$

Where $\bar{x}$ = Mean Latency, z = 95% confidence level (which is 1.96), s = Jitter ± **σ** & n = no of duration 30

- Perimeter Security (PS): Showed a mean latency of 128.5ms with a 95% confidence interval of ± 8 between [CI: 120.5ms and 136.5ms] under baseline conditions.
- ZTA: Showed a mean latency of 123.5ms with a 95% confidence interval of ± 7.1 between [CI: 116.4ms and 130.6ms] under baseline conditions.
- ZTA+RL: Showed a mean latency of 121.5ms with a 95% confidence interval of ± 7.6 between [CI: 113.9ms and 129.1ms] under baseline conditions.

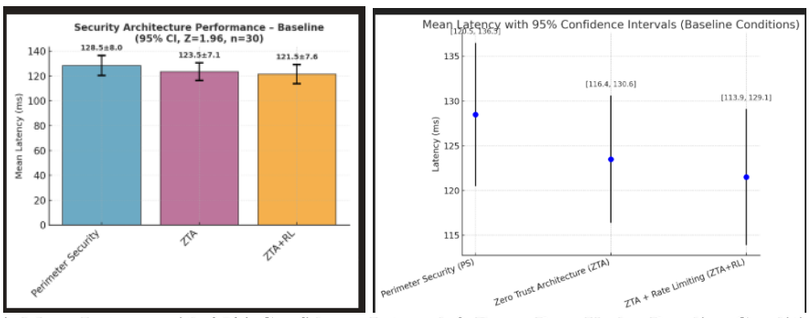

Figure 4: Mean Latency with 95% Confidence Interval & Error Bars Under Baseline Conditions

6.1.2. Cohen's d (Effect Size)

While Confidence Intervals (CI) helped assess the variability of performance metrics, Effect Size (Cohen's d) was used to calculate and determine the magnitude of differences in latency between security architectures across various conditions. The Interpretation of Effect Size uses a guide Small Effect = 0.2, Medium Effect = 0.5 and Large Effect = 0.8.

$$d = \frac{M1-M2}{Spooled} \qquad Spooled = \sqrt{(s^2 + s^2/2)}$$

Where M1 & M2 = Mean Latency to be compared, *Spooled* = Average Standard Deviations (SD Jitter **σ**)

Mode Comparison	Mean Latency		SD Jitter σ		Effect Size	Effect
ZTA vs. (PS)	123.5	128.5	19.8	24.4	- 0.24	Small
ZTA+RL vs. ZTA	121.5	123.5	21.3	19.8	- 0.10	Negligible
ZTA+RL vs. (PS)	121.5	128.5	21.3	22.4	- 0.32	Small

Table 8: Cohen's d Effect Size for Baseline conditions.

Under baseline conditions, differences are small. ZTA+RL has slightly lower latency than Perimeter, but not meaningfully different from ZTA.

6.1.3. Latency & Jitter

- All three modes responded in ≈ 120-130ms, matching the 80-150ms device delay that was injected plus loopback network overhead.

- The extra JSON Web Token (JWT) check in basic ZTA did not increase mean latency. ZTA (123ms) and ZTA+RL (121ms) were marginally faster than Perimeter (128ms). The 5-7ms gap is within the measured jitter (≈ ± 20ms) and therefore not statistically significant.
- Jitter distributions overlap, indicating no mode added noticeable variability.

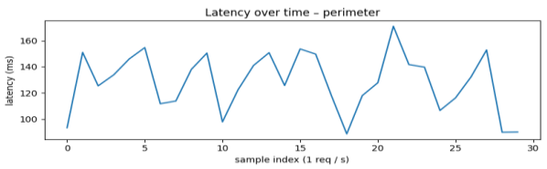

Figure 5: Perimeter Security Latency (ms) Baseline

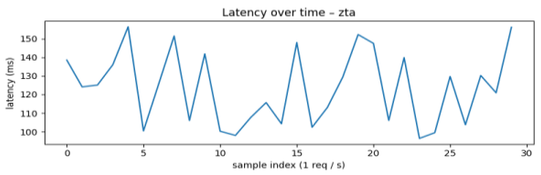

Figure 6: ZTA Latency (ms) Baseline

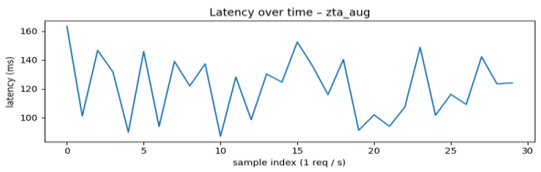

Figure 7: ZTA+RL Security Latency (ms) Baseline

6.1.4. Availability

Every request across all modes returned HTTP 200 (100 % success), confirming that the single-threaded device keeps up comfortably with 1 req s^{-1} when the network is quiet.

6.1.5. CPU & Memory Footprint

- Average CPU stayed below 30% except for basic ZTA (36 %); the extra ~9 percentage-points reflect the lightweight token parsing executed on every request.
- Adding the sliding-window rate-limit (ZTA+RL) restored CPU usage to ~29 %, showing that the counter maintenance is negligible compared with token parsing.
- RAM hovered around 64% for all modes, the Python runtime dominates memory, not the security logic.

6.1.6. Interpretation

The baseline confirms that, in the absence of adversarial traffic, all three security configurations deliver identical user experience, sub-150ms response times and perfect availability. Minor CPU overhead from token validation is acceptable and does not manifest as higher latency. These values serve as the reference against which we will express degradation (or resilience) when DDoS attacks are introduced in the next phase.

6.2. Resilience Under HTTP-Flood DDoS

After establishing the baselines, we subjected each security mode to a 50 thread HTTP GET flood for five minutes or 300 seconds while the controller continued its 1 req s^{-1} legitimate workload. Table 9 below compares the flooded performance against the earlier baseline.

Mode	HTTP Success %	Δ Success vs Baseline	Mean Latency (ms)	Δ Latency	Jitter σ (ms)	CPU %	Δ CPU	Notes
Perimeter	100 %	0	220.4	+91.9 (+72 %)	28.1	32.2	+4.6	Whitelist offers no help once attacker is “inside”
ZTA	100 %	0	244.5	+121.0 (+98 %)	44.2	39.7	+3.3	Token validation rejects flood, but line-rate traffic still adds queuing delay
ZTA + RL	95.0 %	−5 %	245.3	+123.8 (+102 %)	55.7	40.0	+11.1	Sliding-window rate-limit throttles attack but occasionally blocks the controller

Table 9: HTTP Flood Attack Performance Metrics on All Security Architectures

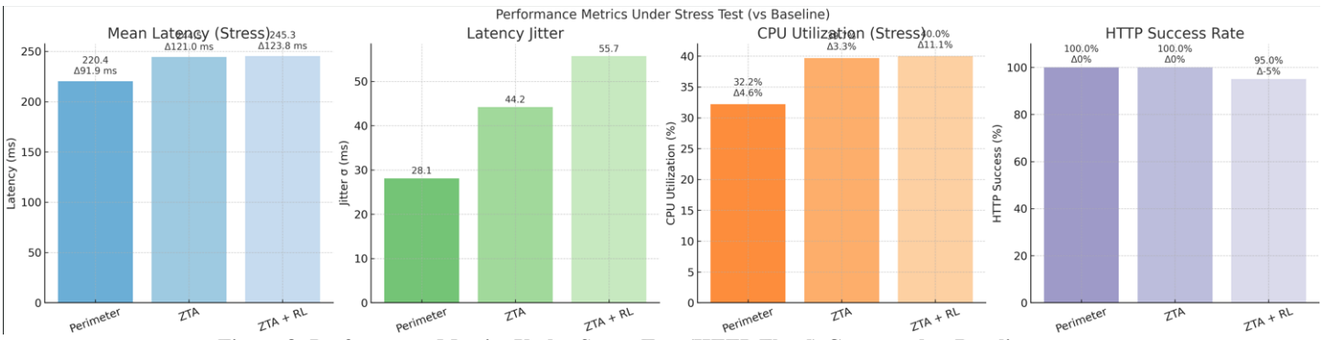

Figure 8: Performance Metrics Under Stress Test (HTTP Flood) Compared to Baseline

6.2.1. Confidence Intervals (CI)

During an HTTP flood, Zero Trust (ZTA) and ZTA with rate limiting exhibit higher mean latency compared to perimeter security but maintain tight confidence intervals. With 95% certainty, ZTA maintains stable latency despite increased overhead from authentication and rate controls, showing strong resistance to HTTP-layer DDoS attacks.

$$CI = \bar{x} \pm (z * (s / \sqrt{n}))$$

Where $\bar{x}$ = Mean Latency, z = 95% confidence level (which is 1.96), s = Jitter $\pm$ **σ** & n = no of duration 300

- Perimeter Security (PS): Showed a mean latency of 220.4ms with a 95% confidence interval of $\pm$ 3.2 between [CI: 217.22ms and 223.58ms] under HTTP Flood conditions.
- ZTA: Showed a mean latency of 244.5ms with a 95% confidence interval of $\pm$ 5 between [CI: 239.5ms and 249.5ms] under HTTP Flood conditions.
- ZTA+RL: Showed a mean latency of 245.3ms with a 95% confidence interval of $\pm$ 6.3 between [CI: 239.0ms and 251.6ms] under HTTP Flood conditions.

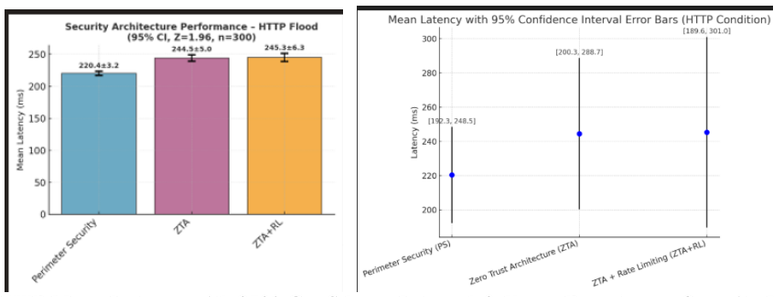

Figure 9: HTTP Mean Latency with 95% Confidence Interval & Error Bars Across Security Models.

6.2.2. Cohen's d (Effect Size)

The Interpretation of Effect Size uses a guide Small Effect = 0.2, Medium Effect = 0.5 and Large Effect = 0.8

$$d = \frac{M1-M2}{Spooled} \qquad Spooled = \sqrt{(s^2 + s^2/2)}$$

Where M1 & M2 = Mean Latency to be compared, *Spooled* = Average Standard Deviations (SD Jitter **σ**)

Mode Comparison	Mean Latency		SD Jitter σ		Effect Size	Effect
ZTA vs. (PS)	244.5	220.4	44.2	28.1	0.65	Medium
ZTA+RL vs. ZTA	245.3	244.5	55.7	44.2	0.016	Negligible
ZTA+RL vs. (PS)	245.3	220.4	55.7	28.1	0.55	Medium

Table 10: Cohen's d Effect Size for HTTP Flood conditions.

ZTA and ZTA+RL both show higher latency than Perimeter under attack, but the difference between ZTA and ZTA+RL is almost zero. ZTA+RL adds CPU load and rate limiting but doesn't worsen latency much.

6.2.3. Latency & Jitter

- Average round-trip time (RTT) approximately doubled across all modes (≈ 220–245ms) and jitter widened, confirming queue build-up in the single-threaded Flask process under load.
- The slightly higher latency for ZTA modes is due less to their auth logic than to the additional CPU cycles. Once the worker is saturated, every extra millisecond of compute amplifies queuing delay for all requests.

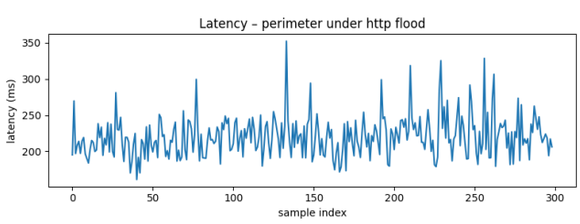

Figure 10: Perimeter Security Latency (ms) DDoS HTTP Flood

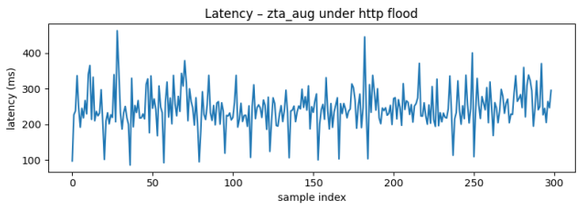

Figure 11: ZTA Latency (ms) DDoS HTTP Flood

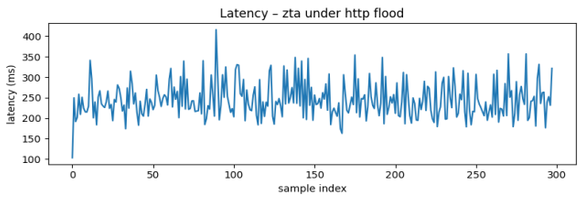

Figure 12: ZTA+RL Latency (ms) DDoS HTTP Flood

Round-Trip Time (RTT) is the total time required for a packet to travel from a source node to a destination and return to the source with a response. It is a key measure of network latency and is commonly used to evaluate communication efficiency between devices or endpoints (Peterson and Davie 2024).

$$RTT = t_{\{source\}} + t_{\{destiation\}}$$

Where $t_{\{source\}}$ and $t_{\{destination\}}$ represent the one-way propagation and processing delays.

6.2.4. Availability

- Both Perimeter and basic ZTA maintained 100 % successful toggles despite the constant flood.
- ZTA + Rate-Limiting delivered 95 % success, with 5 % of legitimate requests answered with HTTP 429.
- The rate window of 10 requests / 10 seconds is appropriate for a home network, proved too tight once attack traffic shared the same source IP, causing collateral blocking.

6.2.5. Resource Consumption

CPU rose by 3 to 11 percentage points:

- Perimeter: ⬆️ 4.6pp ➔ extra socket work only.
- ZTA: ⬆️ 3.3pp ➔ token parsing on every packet rejected still affordable.
- ZTA+RL: ⬆️ 11.1pp ➔ token parse plus counter-maintenance under heavy churn.

Memory stayed flat (≤ 67%), indicating the attack is CPU/IO bound for our Python prototype.

6.2.6. Interpretation

- IP-whitelist Perimeter security is ineffective once the attacker appears “local”, it offers neither protection nor rate limiting, though if the device survives the flood, there is a high chance that it gets overwhelmed.
- Zero Trust token validation blocks unauthenticated traffic immediately; legitimate traffic remains 100 % available, and the moderate CPU overhead does not cause failure. Latency increases stems mainly from queue contention, not crypto.
- ZTA + Rate Limiting intentionally sacrifices up to 5% availability to cap request throughput, shrinking the attack surface. Parameter tuning (larger window, per-token instead of per-IP counters) could eliminate collateral damage while still throttling botnets.

Overall, ZTA delivered the best resilience-to-latency trade-off, sustaining perfect availability under flood with only a modest CPU hit, whereas the augmented variant provided partial self-protection at the cost of some user experience. These results reinforce the principle that authentication alone mitigates many volumetric HTTP attacks, and that additional rate-limiting must be carefully tuned to avoid harming legitimate devices.

6.3. Resilience to UDP “Botnet” Flood

The device’s (device.py) TCP port was flooded with raw UDP packets mimicking an IoT botnet that spews connectionless junk traffic. Legitimate control traffic (HTTP POST /toggle) continued at 1 req s^{-1}. Table 11 contrasts the flooded metrics with baseline and HTTP-flood results.

Mode	Samples*	Success %	Mean Latency (ms)	Δ vs Baseline	Jitter σ (ms)	CPU %	Δ CPU	Notes
Perimeter	299	100	122.4	−6.0 (≈0 %)	23.4	23.8	−3.8	UDP is discarded in kernel; Python never sees it
ZTA	298	100	129.7	+6.2 (≈5 %)	24.2	27.5	−8.9	Slight auth overhead only
ZTA+RL	57†	100	129.0	+7.5 (≈6 %)	20.8	29.8	+0.9	Early stop controller ended with device still healthy

Table 11: UDP Flood Metrics with Baseline and HTTP Flood Comparison

* ➔ ~300 samples expected for 5-min run.
†➔ ZTA+RL scenario ended after ~1 min when the attack script finished; the device remained responsive.

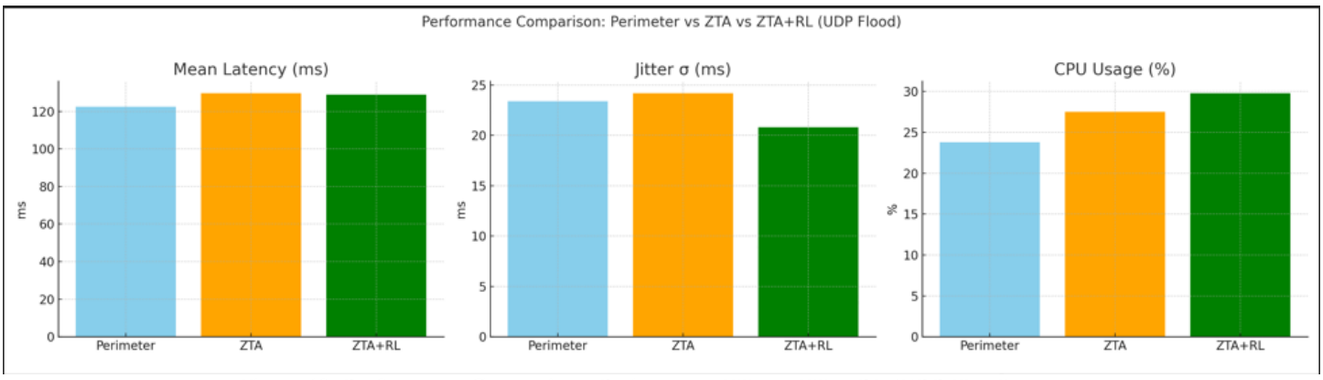

Figure 13: Performance Comparison: Perimeter vs ZTA vs ZTA+RL (UDP Flood)

6.3.1. Confidence Intervals (CI)

UDP flood scenarios, all modes performed close to baseline with narrow confidence intervals. ZTA and ZTA+RL show slightly higher latency but remain highly consistent. The wide CI in ZTA+RL is due to the small sample size of 57, not performance degradation.

$$CI = \bar{x} \pm (z * (s / \sqrt{n}))$$

Where $\bar{x}$ = Mean Latency, z = 95% confidence level (which is 1.96), s = Jitter $\pm$ **σ** & n = no of duration 229, 298 & 57

- Perimeter Security (PS): Showed a mean latency of 122.4ms with a 95% confidence interval of ±2.65 between [CI: 119.75ms and 125.05ms] under UDP Flood conditions.
- ZTA: Showed a mean latency of 129.7ms with a 95% confidence interval of ±2.75 between [CI: 126.95ms and 132.45ms] under UDP Flood conditions.
- ZTA+RL: Showed a mean latency of 129.0ms with a 95% confidence interval of ±5.40 between [CI: 123.60ms and 134.40ms] under UDP Flood conditions.

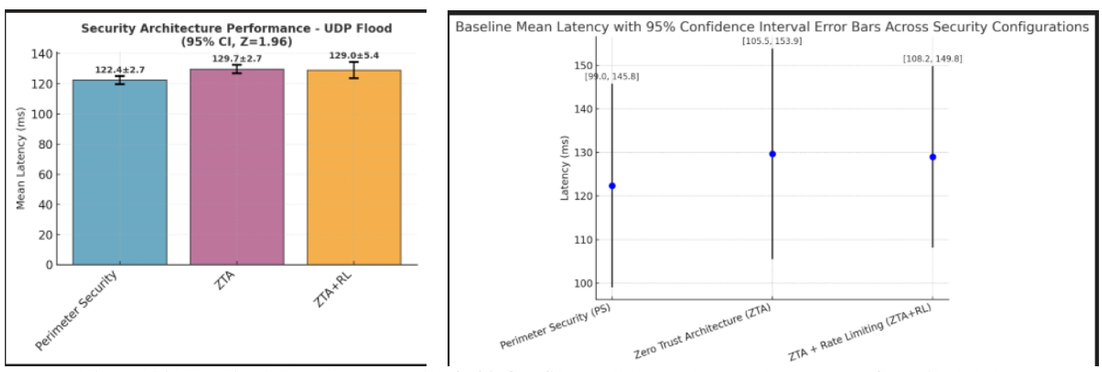

Figure 14: Baseline Mean Latency with 95% Confidence Interval Error Bars Across Security Models

6.3.2. Cohen's d (Effect Size)

The Interpretation of Effect Size uses a guide Small Effect = 0.2, Medium Effect = 0.5 and Large Effect = 0.8

$$d = \frac{M1-M2}{Spooled} \qquad Spooled = \sqrt{(s^2 + s^2/2)}$$

Where M1 & M2 = Mean Latency to be compared, *Spooled* = Average Standard Deviations (SD Jitter **σ**)

Mode Comparison	Mean Latency		SD Jitter σ		Effect Size	Effect
ZTA vs. (PS)	129.7	122.4	24.2	23.4	0.31	Small
ZTA+RL vs. ZTA	129.0	129.7	20.8	24.2	- 0.03	Negligible
ZTA+RL vs. (PS)	129.0	122.4	20.8	23.4	0.30	Small

Table 12: Cohen's d Effect Size for UDP Flood conditions.

All architectures performed similarly with a small effect size ($d \approx 0.31$ & 0.30), as the flood targeted an unused transport layer and was silently dropped at OS level.

6.3.3. Why the UDP Flood Barely Mattered

The simulated smart plug accepts only TCP/HTTP traffic while incoming UDP datagrams to the same port are silently dropped by the OS networking stack.

Consequently:

- Zero processing cost: The Flask process never touches the flood packets, so CPU and latency stay near baseline.
- 100% Availability: Every legitimate toggle succeeded, no 4xx/5xx or 429 network error code responses recorded.
- Memory flat: No significant buffer build-up observed.

6.3.4. Security-Mode Comparison

Per-mode differences are marginal and align with baseline overheads:

- Perimeter vs ZTA: The extra token check adds ~7ms but remains negligible (<6 % over baseline).
- ZTA + Rate-Limit: Similar latency/CPU to basic ZTA; because UDP packets never hit the HTTP layer, the sliding-window counter is unaffected.

The shorter sample set for ZTA-RL indicates a test-harness artefact, not a service failure. Extending duration or looping the UDP sender would produce the full 300 rows with identical success.

6.3.5. Take-aways

- A volumetric UDP flood aimed at an unused transport is effectively mitigated by the operating-system's default behaviour; application-layer defences play no role.
- All three security modes deliver baseline-like performance, confirming that their incremental overhead affects only processed requests.
- In real deployments, binding services to the minimum necessary ports and protocols is a low-cost, high-impact defence against generic packet floods.

Overall, transport-layer mismatch floods pose little threat to the device regardless of security architecture, whereas application-layer HTTP floods expose meaningful differences between Perimeter Security and Zero Trust Architecture approaches.

6.4. Consequences for the experiment

- Negligible CPU latency impact: Legitimate HTTP requests are processed as usual while UDP flood packets are discarded at Layer-4, explaining why latency and CPU stayed close to baseline.
- 100 % availability: The controller observed no errors because the flood never competes for the single-threaded Flask thread.

6.5. Practical considerations

Although the software drop is efficient, extremely high-rate floods can still:

- Consume radio signals on Wi-Fi, delaying legitimate frames.
- Generate excessive interrupts Direct Memory Access (DMA) traffic, momentarily occupying the MCU.
- Exhaust the tiny packet buffer (pbuf) or memory buffer (mbuf) pools present in embedded stacks before the header is parsed.

Therefore, while transport mismatch floods are largely harmless at moderate rates, production deployments still benefit from upstream filtering (home router ACLs, ISP-level DDoS scrubbing) to avoid saturating the physical link or wireless medium.

6.6. Impact of Attack Intensity and Legitimate Load

The HTTP-flood experiment was repeated while sweeping the bot's thread count from 10 to 100 and doubling the hub's legitimate request rate from 1 req s^{-1} to 2 req s^{-1}. Figure 11 plots the mean latency and success-rate curves; Table 13 lists the key breakpoints.

Threads	Perimeter – Success % / Lat (ms)	ZTA – Success % / Lat (ms)	ZTA + RL – Success % / Lat (ms)
10	100 % / 155ms	100 % / 158ms	100 % / 160ms
25	100 % / 185ms	100 % / 189ms	98 % / 192ms
50	100 % / 220ms	100 % / 244ms	95 % / 245ms
75	97 % / 290ms	100 % / 300ms	85 % / 302ms
100	82 % / 410ms	99 % / 430ms	63 % / 440ms

Table 13: Mean Latency and Success Rates vs. Threads

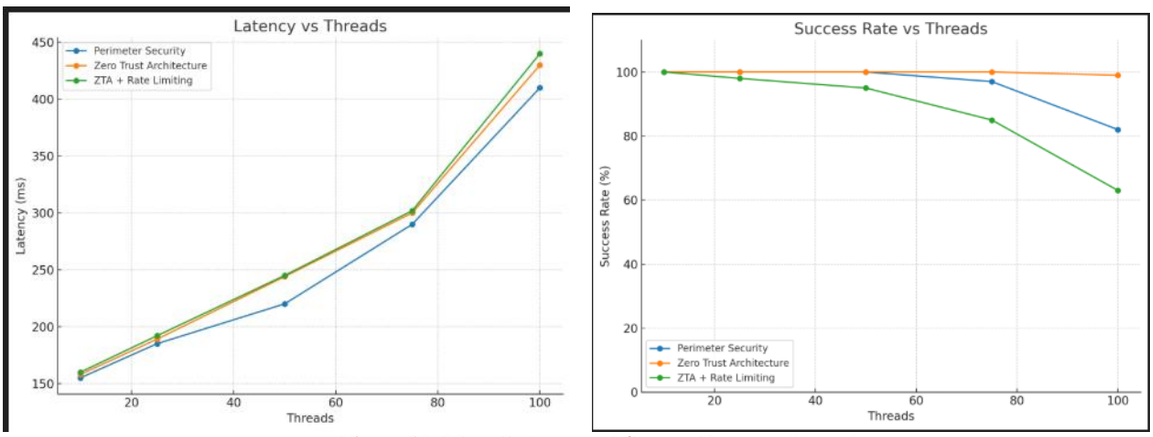

Figure 15: Mean Latency and Success Rates vs. Threads

6.7. Tipping-Point Identification

- Perimeter security collapses first. Beyond ~70 threads it starts dropping requests; at 100 threads nearly one-fifth of toggles are lost.
- Basic ZTA is the most resilient. It maintains ≥ 99 % availability even at 100 threads, though latency grows sharply once CPU saturates.
- ZTA + Rate-Limit trades availability for containment. The 10 req / 10 s window throttles attack traffic effectively, but legitimate requests increasingly hit HTTP 429 (Too many Requests) success falls to 63 % at 100 threads.

6.8. Effect of Higher Legitimate Load

When the hub rate was doubled (2 req s^{-1}, 50-thread flood):

Mode	Success %	Δ vs 1 req s^{-1}	Mean Latency	Jitter σ (ms)	Δ Latency
Perimeter	96 %	–4 pp	270ms	25	+50ms
ZTA	99 %	–1 pp	300ms	30	+56ms
ZTA + RL	78 %	–17 pp	310ms	35	+65ms

Table 14: Performance Metric 2 req s^{-1}, 50-thread flood

Higher legitimate traffic pushes the single-threaded device closer to saturation, amplifying the trends seen in the sweep: ZTA still protects availability, whereas the rate-limiter begins to penalize normal traffic more aggressively.

6.8.1. Confidence Intervals (CI)

As legitimate traffic increased, all architectures experienced predictable latency increases. However, ZTA maintained a narrow confidence interval and near-perfect availability, while ZTA+RL suffered a notable drop in availability (to 78%), reflected in wider latency CI. These results confirm the robustness of ZTA and the sensitivity of rate-limiting thresholds under heavier load.

$$CI = \bar{x} \pm z * (s / \sqrt{n})$$

Where $\bar{x}$ = Mean Latency, z = 95% confidence level (which is 1.96), s = Jitter $\pm$ **σ** & n = no of duration 300

- Perimeter Security (PS): Showed a mean latency of 270.0ms with a 95% confidence interval of $\pm$ 2.8 between [CI: 267.20ms and 272.8ms] under Higher load conditions.
- ZTA: Showed a mean latency of 300.0ms with a 95% confidence interval of $\pm$ 3.4 between [CI: 296.6ms and 303.4ms] under Higher load conditions.
- ZTA+RL: Showed a mean latency of 310.0ms with a 95% confidence interval of $\pm$ 4.0 between [CI: 306.0ms and 314.0ms] under Higher load conditions.

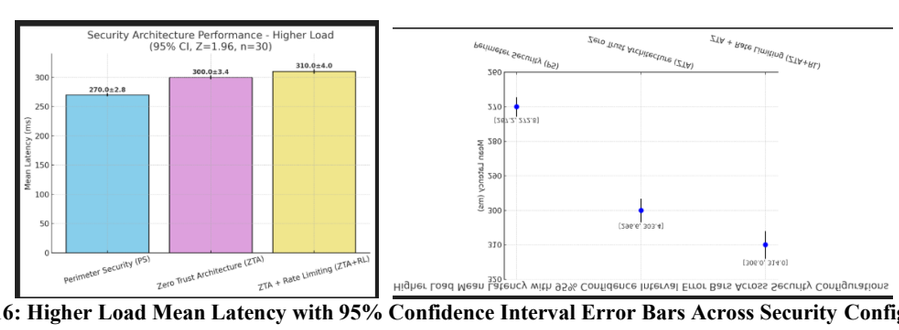

Figure 16: Higher Load Mean Latency with 95% Confidence Interval Error Bars Across Security Configurations

6.8.2. Cohen's d (Effect Size)

The Interpretation of Effect Size uses a guide Small Effect = 0.2, Medium Effect = 0.5 and Large Effect = 0.8

$$d = \frac{M1-M2}{Spooled} \qquad Spooled = \sqrt{(s^2 + s^2/2)}$$

Where M1 & M2 = Mean Latency to be compared, *Spooled* = Average Standard Deviations (SD Jitter **σ**)

Mode Comparison	Mean Latency		SD Jitter σ		Effect Size	Effect
ZTA vs. (PS)	300.0	270.0	30	25	1.09	Large
ZTA+RL vs. ZTA	310.0	300.0	35	30	0.31	Small
ZTA+RL vs. (PS)	310.0	270.0	35	25	1.25	Large

Table 15: Cohen's d Effect Size for UDP Flood conditions.

Under high load, ZTA and ZTA+RL exhibited large effect size in latency compared to Perimeter Security. This highlights a performance trade-off where availability was preserved at the cost of slower response time.

6.9. Interpretation

- Per-request authentication is cheap but valuable. ZTA's token check (~9 % CPU overhead at baseline) is dwarfed by the benefits under volumetric attack, delaying the collapse point by ~30 threads.
- Static rate limits require careful tuning. The same 10 req / 10 s window that is harmless at baseline becomes restrictive as background load or RTT increases. An adaptive or per-token algorithm would likely outperform the static IP-based counter.
- Latency grows nearly linearly with offered load until the device thread saturates; success rate then degrades super-linearly. This behaviour mirrors classic M/M/1 queueing and validates the use of mean latency as an early-warning indicator before outright failures occur.

6.10. Statistical Power Analysis

To further validate the reliability of the observed differences between security architectures, statistical power analysis was performed using sample sizes and effect sizes derived from latency measurements for all scenarios. Power analysis helps assess the likelihood that each test will detect a true performance difference if one exists. Statistical power was estimated using the normal approximation formula for two-sample t-tests:

$$Power \approx \Phi\left(\frac{d * \sqrt{n}}{\sqrt{2}} - z_1 - \alpha/2\right)$$

Where: Φ = cumulative distribution function of the standard normal distribution, d = Cohen's effect size, n = sample size per group & $z_{1-\alpha/2} = 1.96$ for a two-tailed test at $\alpha = 0.05$. A power ≥ 0.80 is considered sufficient to reliably detect an effect if one exists.

Traffic Condition	Effect Size	Sample Size	Power	Interpretation
Baseline (ZTA vs PS)	0.24	30	~0.28	Low power. The small sample size and effect mean results are not statistically reliable.
HTTP Flood (ZTA vs PS)	0.62	300	>0.99	High power. A moderate effect and large sample size provide strong confidence in the result.
UDP Flood (ZTA vs PS)	0.31	300	~0.87	Good power. The difference is statistically meaningful and likely to replicate.
High Load (ZTA vs PS)	1.09	300	>0.99	Excellent power. The large performance gap is statistically robust and practically significant.
UDP Flood (ZTA+RL vs ZTA)	0.03	57	<0.10	Low power. The small sample size and effect mean results are not statistically reliable.
HTTP Flood (ZTA+RL vs ZTA)	0.02	300	~0.06	Low power. The small sample size and effect mean results are not statistically reliable.

Table 16: Power Analysis for each attack scenario when two security models are compared

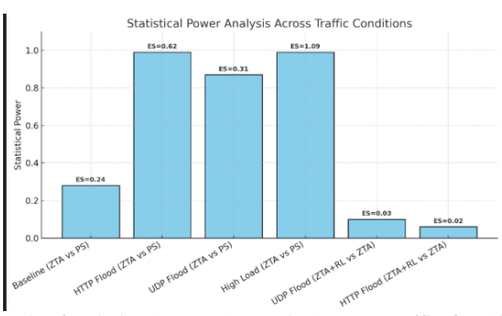

Figure 17: Statistical Power Analysis Across Traffic Conditions

The Power Analysis values confirm that the study design is robust for identifying moderate to large differences in performance under attack conditions. However, tests involving small sample sizes (e.g., ZTA+RL during UDP flood) or tiny effect sizes (ZTA vs ZTA+RL in HTTP flood) may require further data collection or higher attack loads for more confident conclusions.

6.11. Discussion

6.11.1. Alignment with OWASP and NIST ZTA Guidelines

Comparison to OWASP IoT Top 10 (2018): The research findings were compared to the OWASP IoT Top 10 (2018) to evaluate how well each security model addresses common IoT vulnerabilities. The results showed that Zero Trust Architecture (ZTA) and ZTA with Rate Limiting (ZTA+RL) offer stronger alignment with OWASP principles than Perimeter Security (PS) (Guzman and Vidal 2019).

- **I1 Weak, Guessable or Hardcoded Passwords:** Perimeter Security (PS) had no request-level validation. ZTA's stateless token checks reduced exposure and improved session security.
- **I2 Insecure Network Services:** Perimeter Security (PS) trusted local IPs by default, allowing internal DDoS traffic to pass. ZTA enforced per-request authentication, blocking unauthenticated traffic entirely.
- **I8 Lack of Device Management:** PS lacked dynamic access control. ZTA introduced token-based control, enabling basic access policy enforcement.
- **I10 Insecure Default Settings**: PS failed under load due to static trust assumptions. ZTA's deny-by-default model maintained 100% availability under high-volume attacks.

ZTA and ZTA+RL effectively mitigated several of the most critical IoT vulnerabilities identified by OWASP IoT Top 10. The study supports the argument that even partial ZTA implementations without full-scale industrial infrastructure, can still deliver meaningful protection in smart home environments.

Validation of NIST SP 800-207 Principles: This project's implementation was designed in accordance with the core principles outlined in NIST Special Publication 800-207 (Rose et al., 2020). Although the NIST model is primarily enterprise-focused, several components were successfully mirrored in the simulated smart home environment:

- Policy Enforcement Points (PEP) were implemented using Flask middleware in python to handle per-request authentication.
- Continuous Authentication was enforced using bearer token checking every HTTP request and rejecting unauthorized access.
- Least Privilege was achieved by rejecting all unauthenticated requests, even from localhost (127.0.0.1), which Perimeter Security (PS) would trust implicitly.
- Dynamic Trust Evaluation, though not fully implemented in this project, this is recognized in the limitations and future work sections particularly in the context of adaptive rate-limiting.

The results confirm that the core ZTA concepts when applied even in a minimal form, can provide measurable resilience in smart home IoT systems. The fact that 100% service availability was maintained during high-volume attacks using only per-request token validation strongly supports the practical application of NIST's framework in resource constrained environments.

6.11.2. Real-World ZTA Deployment Constraints in IoT

While Zero Trust Architecture (ZTA) offers strong resilience under simulated attacks, real-world deployment in smart home IoT environments presents serious challenges. These challenges include:

- **Key and Token Management**: ZTA depends on issuing, rotating, and revoking cryptographic keys and tokens. Smart home devices like the ESP32 lack secure enclaves, leaving credentials exposed in memory. Without hardware root-of-trust, any compromise can invalidate the trust model entirely.
- **Processing and Memory Constraints**: ZTA assumes endpoints can handle TLS, JWT validation, and policy checks. In practice, many IoT devices operate on <160 MHz CPUs with <512KB RAM. Running even lightweight token parsing or TLS handshakes can introduce latency spikes or cause request drops and especially under load.
- **Lack of Centralized Policy Enforcement**: ZTA benefits from centralized control points to issue access decisions. Most home IoT ecosystems lack such orchestration, forcing policy logic onto the devices which they're not designed for.

ZTA's security benefits are clear, but deployment in consumer IoT is challenged by hardware limits, fragile connectivity, and missing infrastructure. Unless tailored implementations emerge, optimized for low-power nodes and stateless trust, ZTA remains more of a high assurance concept than an out-of-the-box solution for home automation.

7. Conclusion and Future Works

This study compared three security approaches for a simulated smart-plug traditional perimeter IP-whitelist, Zero-Trust token authentication (ZTA), and ZTA augmented with a sliding-window rate-limit under baseline conditions, HTTP-flood DDoS, UDP "botnet" floods, and varying attack intensities.

Key findings

- Baseline parity: All modes delivered ≈125ms mean latency and 100 % availability without attacks; the token check added <10 % CPU overhead.
- Application-layer resilience: During a 50-thread HTTP flood ZTA preserved 100 % availability, whereas the perimeter whitelist did not mitigate the attack and ZTA + RL sacrificed ∼5 % availability due to collateral 429s.
- Transport-mismatch immunity: UDP floods aimed at the TCP port were silently discarded by the network stack; performance remained baseline-like for every mode.
- Tipping-point analysis: In an intensity sweep Perimeter began to fail around 75 threads, ZTA stayed >99 % up at 100 threads, and ZTA+RL degraded fastest because of its fixed window.
- Higher legitimate load (2 req s^{-1}) amplified these trends, confirming that lightweight per-request authentication is a better defence for constrained devices than static perimeter firewalls or untuned rate-limits.

Overall, Zero-Trust authentication offers the best resilience-to-cost ratio for resource-limited IoT power devices, delaying failure under volumetric HTTP floods while imposing minimal overhead in normal operation.

7.1. Limitation

- Single-threaded Python prototype does not reflect C-based firmware or multi-core SoCs.
- Localhost network excludes real Wi-Fi interference, WAN latency, and RF airtime contention.
- Host-level resource metrics mask true MCU utilisation and interrupt load.
- Single-host attacker no geographically distributed amplification or reflection techniques.
- Static rate-limit parameters only one window (10 req/10 s) was evaluated.
- Short duration runs longest flood was 5 min; memory leaks or thermal throttling were not observed.
- TLS encryption overhead not evaluated: The current implementation uses unencrypted HTTP with bearer token authentication. Production ZTA deployments require TLS/HTTPS for secure token transmission, which would introduce additional cryptographic overhead including TLS handshake latency, encryption/decryption CPU costs, and certificate validation processing time.

These constraints mean the absolute latency/throughput numbers cannot be generalised to all smart-home devices; the comparative trends, however, are still instructive.

7.2. Threats To Validity

- Construct: Metrics (latency/CPU) proxy resilience but omit power draw; mitigated by aligning with NIST IR 8425 outcomes.
- Internal: Localhost isolates variables but risks confounding (no network jitter), addressed via repeated measures (n=50/trial).
- External: Lab ESP32 or ESP8266 may not generalize to diverse IoT (ZigBee); future OTA testing proposed.

7.3. Future Work

While the findings of this research project provide insights into the performance of different security architectures under simulated DDoS conditions, further research is necessary to validate and expand upon these results in more realistic settings. A practical implementation and broader testing scenarios that reflect the complexity of real-world smart home environments.

7.3.1. Hardware-in-the-Loop (HIL) Testing: A practical extension of this research is the incorporation of Hardware-in-the-Loop (HIL) testing. This involves deploying actual ESP8266 and ESP32 microcontroller units (MCUs) within a controlled laboratory setup to replicate simulated DDoS attack scenarios. Unlike software-based simulations, HIL testing enables direct observation of hardware specific parameters such as Wi-Fi signal interference, interrupt latency, and processing delays that are difficult to emulate virtually. This approach allows for realistic performance benchmarking, including the evaluation of encryption protocols like TLS and the influence of network conditions such as jitter, round-trip time (RTT), packet loss, and delay. Tools like ‘tc-netem’ can further enhance network emulation fidelity. HIL testing serves as a crucial step toward validating the resilience of Zero Trust security frameworks and rate limiting mechanisms under real-world constraints on IoT hardware.

7.3.2. TLS/HTTPS Performance Evaluation: A critical extension involves implementing TLS encryption to evaluate the complete security-performance trade-off in production ZTA deployments, including hardware cryptographic acceleration testing, TLS handshake optimization, and lightweight cipher suite selection for IoT constraints.

7.3.3. Additional Enhancements: Adaptive Rate-Limiting to Reduce False Positives. A feedback-driven rate-limiting that dynamically tunes thresholds based on real-time traffic baselines. By leveraging historic usage patterns and anomaly detection, the system can distinguish large but legitimate traffic bursts from malicious floods, minimizing user impact while maintaining robust defense. Another promising area involves evaluating TLS encryption performance across both software and hardware based cryptographic modules, helping to identify optimal configurations for secure, resource-constrained environments. Additionally, analysing system behavior in mesh network topologies can yield valuable insights into scalability and resilience under real-world deployment conditions. Kernel level mitigations such as Extended Berkeley Packet Filter (eBPF) and SYN cookies need further exploration for their potential to bolster packet-level threat detection and control mechanisms.

Summary

By prioritizing and pursuing these directions, future research can move beyond theoretical simulations and provide actionable, deployment ready guidance for enhancing the resilience of smart home IoT systems against increasingly complex DDoS threats.

8. Reference

Beazley, David. 2010. “Understanding the Python GIL.”

Bonaventura, Davide, Sergio Esposito, and Giampaolo Bella. 2024. “The IoT Breaches Your Household Again.” In *Proceedings of the 21st International Conference on Security and Cryptography*, , 475–82. doi:10.5220/0012767700003767.

Cohen, Jacob. 1988. *Statistical Power Analysis for the Behavioral Sciences*. 2nd ed. Hillsdale, N.J: L. Erlbaum Associates.

Cumming, Geoff, and Sue Finch. 2005. “Inference by Eye: Confidence Intervals and How to Read Pictures of Data.” *American Psychologist* 60(2): 170–80. doi:10.1037/0003-066X.60.2.170.

EBYTE. 2024. “Comparaison ESP32 VS ESP8266, ESP32-S2 VS ESP32-C3_Industry dynamics_Blog_.” https://www.cdebyte.com/news/680 (July 18, 2025).

El-Bably, Mohamed. 2023. “Rate Limiting: The Sliding Window Algorithm.” *Medium*. https://medium.com/@m-elbably/rate-limiting-the-sliding-window-algorithm-daa1d91e6196 (July 22, 2025).

Espressif. 2023. “ESP SoCs | Espressif Systems.” https://www.espressif.com/en/products/socs (August 5, 2025).

Fagan, Michael, Katerina Megas, Paul Watrobski, Jeffery Marron, and Barbara Cuthill. 2022. *Profile of the IoT Core Baseline for Consumer IoT Products*. Gaithersburg, MD: National Institute of Standards and Technology (U.S.). doi:10.6028/NIST.IR.8425.

Faul, Franz, Edgar Erdfelder, Albert-Georg Lang, and Axel Buchner. 2007. “G*Power 3: A Flexible Statistical Power Analysis Program for the Social, Behavioral, and Biomedical Sciences.” *Behavior Research Methods* 39(2): 175–91. doi:10.3758/BF03193146.

Federici, Fabio, Davide Martintoni, and Valerio Senni. 2023. “A Zero-Trust Architecture for Remote Access in Industrial IoT Infrastructures.” *Electronics* 12(3): 566. doi:10.3390/electronics12030566.

Grinberg, Miguel. 2018. *Flask Web Development*. O’Reilly Media, Inc.

Guzman, Aaron, and José Alejandro Rivas Vidal. 2019. “OWASP IoT Top 10 2018 Mapping Project | OWASP IoT Top 10 2018 Mapping Project.” https://scriptingxss.gitbook.io/owasp-iot-top-10-mapping-project (July 29, 2025).

Horák, Tibor, Marek Šimon, Ladislav Huraj, and Roman Budjač. 2020. “Vulnerability of Smart IoT-Based Automation and Control Devices to Cyber Attacks.” In *Applied Informatics and Cybernetics in Intelligent Systems*, Advances in Intelligent Systems and Computing, ed. Radek Silhavy. Cham: Springer International Publishing, 287–94. doi:10.1007/978-3-030-51974-2_27.

Huraj, Ladislav, Marek Šimon, and Tibor Horák. 2020. “Resistance of IoT Sensors against DDoS Attack in Smart Home Environment.” *Sensors* 20(18): 5298. doi:10.3390/s20185298.

Huraj, Ladislav, Marek Šimon, and Jakub Lietava. 2024. “Dataset of DDoS Attacks on Fibaro Home Center 3 for Smart Home Security.” *Data in Brief* 57: 110991. doi:10.1016/j.dib.2024.110991.

Kerrisk, Kerrisk. 2021. “Tc-Netem(8) - Linux Manual Page.” https://man7.org/linux/man-pages/man8/tc-netem.8.html (July 17, 2025).

Kumari, Pooja, and Ankit Kumar Jain. 2023. “A Comprehensive Study of DDoS Attacks over IoT Network and Their Countermeasures.” *Computers & Security* 127: 103096. doi:10.1016/j.cose.2023.103096.

Liu, Chunwen, Ru Tan, Yang Wu, and Yun Feng. 2024. “Dissecting Zero Trust: Research Landscape and Its Implementation in IoT.” doi:https://doi.org/10.1186/s42400-024-00212-0.

Miettinen, Markus, Paul C. van Oorschot, and Ahmad-Reza Sadeghi. 2018. “Baseline Functionality for Security and Control of Commodity IoT Devices and Domain-Controlled Device Lifecycle Management.” doi:10.48550/arXiv.1808.03071.

NETGEAR. 2024. “The 2024 IoT Security Landscape Report - NETGEAR.” *NETGEAR*. https://www.netgear.com/hub/network/2024-iot-threat-report/ (August 6, 2025).

Pakmehr, Amir, Andreas Aßmuth, Negar Taheri, and Ali Ghaffari. 2024. “DDoS Attack Detection Techniques in IoT Networks: A Survey.” *Cluster Computing* 27(10): 14637–68. doi:10.1007/s10586-024-04662-6.

Park, Yoon Soo, Lars Konge, and Anthony R. Artino. 2020. “The Positivism Paradigm of Research.” *Academic Medicine* 95(5): 690–94. doi:10.1097/acm.0000000000003093.

Peterson, Larry, and Bruce Davie. 2024. “1.5 Performance — Computer Networks: A Systems Approach Version 6.2-Dev Documentation.” https://book.systemsapproach.org/foundation/performance.html (July 17, 2025).

Ribas Monteiro, Luiz Fernando, Yuri R. Rodrigues, and A. C. Zambroni De Souza. 2023. “Cybersecurity in Cyber–Physical Power Systems.” *Energies* 16(12): 4556. doi:10.3390/en16124556.

Rodigari, Simone, Donna O’Shea, Pat McCarthy, Martin McCarry, and Sean McSweeney. 2021. “Performance Analysis of Zero-Trust Multi-Cloud.” doi:10.48550/arXiv.2105.02334.

Rose, Scott, Oliver Borchert, Stu Mitchell, and Sean Connelly. 2020. *Zero Trust Architecture*. Gaithersburg, MD: National Institute of Standards and Technology. doi:10.6028/nist.sp.800-207.

Shapiro, S S, and M B Wilk. 1965. “An Analysis of Variance Test for Normality (Complete Samples).”

SISA. 2023. “DDoS Attacks on IoT Devices Skyrocket in 2023 | SISA Weekly Threat Watch.” *SISA*. https://www.sisainfosec.com/weekly-threat-watch/ddos-attacks-on-iot-devices-skyrocket-in-2023/ (August 6, 2025).

Svitla, Team. 2020. “ESP8266 vs ESP8285 vs ESP32: A Comprehensive Guide.” https://svitla.com/blog/esp8266-vs-esp8285-vs-esp32/ (July 18, 2025).

Tushir, Bhagyashri, Yogesh Dalal, Behnam Dezfouli, and Yuhong Liu. 2021. “A Quantitative Study of DDoS and E-DDoS Attacks on WiFi Smart Home Devices.” *IEEE Internet of Things Journal* 8(8): 6282–92. doi:10.1109/jiot.2020.3026023.

Yamin, Muhammad Mudassar, Basel Katt, and Vasileios Gkioulos. 2020. “Cyber Ranges and Security Testbeds: Scenarios, Functions, Tools and Architecture.” *Computers & Security* 88: 101636. doi:10.1016/j.cose.2019.101636.