National College of Ireland

Configuration Manual

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 in Cybersecurity **Year:** 2025
Module:	MSc Research Project Configuration Manual
Lecturer:	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: 2918	**Page Count:** 20

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):	

Configuration Manual

Andrew Chinedu Enenmoh
Student ID: x23315679

1. Introduction

1.1 Research Overview

This configuration manual provides comprehensive step-by-step instructions for replicating the IoT security architecture evaluation research that compares three distinct security models under simulated DDoS attack conditions. The framework evaluates Perimeter Security (PS), Zero Trust Architecture (ZTA), and ZTA with Rate Limiting (ZTA+RL) to provide empirical evidence for IoT security architecture selection in smart home environments.

As Internet of Things (IoT) devices grows for smart home integration, traditional perimeter-based security models face unprecedented challenges. Smart home power control devices such as intelligent switches, smart plugs, thermostats, and energy management systems represent critical infrastructure that controls essential electrical systems. A security compromise in these devices presents significant safety and operational risks, making robust security architecture selection important (Rose et al. 2020) (Miettinen, Oorschot, and Sadeghi 2018) (Pakmehr et al. 2024) (Fagan et al. 2022).

This research framework simulates ESP8266/ESP32-based smart home power control devices and evaluates how different security models perform under:

- Normal baseline conditions (1 request/second)
- HTTP flood attacks (Layer 7 application attacks)
- UDP flood attacks (Layer 4 transport attacks)
- Varying attack intensities (10-100 concurrent threads)

1.2 Research Objectives

The configuration manual supports research aimed at:

- **Comparative Security Analysis:** Evaluating the effectiveness of different security architectures in protecting IoT power control devices under normal operational conditions and during active cyberattacks.
- **Performance Impact Assessment:** Measuring the computational overhead, network latency, and resource consumption introduced by each security model to understand the trade-offs between security and performance.
- **Resilience Under Attack:** Analysing system behavior, availability metrics, and recovery capabilities during simulated Distributed Denial of Service (DDoS) attacks to determine which architectures maintain operational integrity under stress.
- Implementation Feasibility: Assessing the practical deployment considerations, configuration complexity, and maintenance requirements for each security architecture in resource constrained IoT environments.

2. System Requirements

2.1 Hardware Requirement

The experiments were conducted in both local and cloud-based environments to evaluate their effectiveness. The local setup involved a VirtualBox virtual machine running the Ubuntu or Kali Linux operating systems, 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.

- DeepNote Containerized Environment (Recommended) running with (2vCPU cores, 5GB RAM, running Linux) DeepNote ZTA Project
- A 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
  - Network: Localhost/loopback interface (no external network required)

2.2 Operating System Requirements

- Primary: Linux (Ubuntu 20.04+ recommended or Kali Linux)
- Alternative: macOS 10.15+ or Windows 10/11 with WSL2

2.3 Software Dependencies

- Python: Version 3.9 or higher
- VS Code
- Git: For repository cloning (optional)
- Administrative privileges: Required for UDP flood attacks

2.4 Python Package Dependencies

The following packages are required (requirements.txt):

Python Libraries	Purpose
Flask	Device micro-services
Requests	Hub-to-device traffic
Scapy	UDP & SYN packet crafting
Psutil	CPU / RAM instrumentation
Matplotlib + Pandas	Data analysis & visualisation
Scipy	Supports Statistical Analysis
Packaging	Version compatibility and handle environment checks.
Numpy	Statistical and numerical calculations

Table 1: Python Libraries Dependencies

3. Installation and Setup

3.1 Repository Setup

Step 1: Create Project Directory

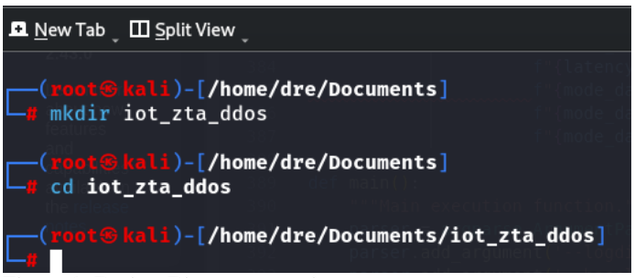


Figure 1: Project Directory creation

Step 2: All the python files are downloaded into the project directory from DeepNote:

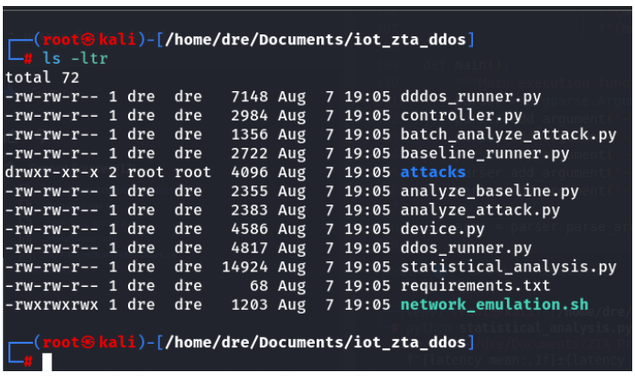


Figure 2: Contents in the Project Directory

File/Module	Description
device.py	IoT device simulator
controller.py	Smart home hub simulator
baseline_runner.py	Baseline experiment runner
ddos_runner.py	DDoS attack experiment runner
analyze_baseline.py	Baseline data analyzer
analyze_attack.py	Attack data analyzer
batch_analyze_attack.py	Batch analysis tool
requirements.txt	Python dependencies
attacks/http_flood.py	HTTP flood attack script
attacks/udp_flood.py	UDP flood attack script
statistical_analysis.py	CI, Cohen's d and Power Analysis
network_emulation.sh	tc-netem

Table 2: Python Files and Shell Script for the Project

3.2 Python Virtual Environment Setup:

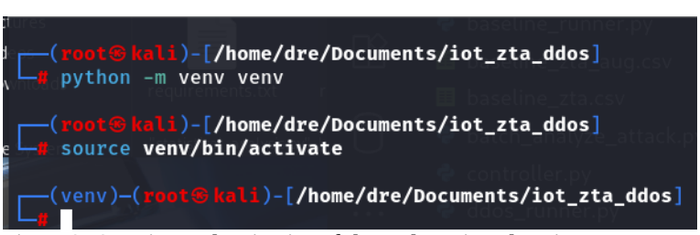


Figure 3: Creation and activation of the python virtual environment

Verification:
Your terminal prompt should change to show `(venv)` prefix, indicating the virtual environment is active.

3.3 Dependencies Installation

Step 1: Upgrade pip

```
┌──(venv)─(root㉿kali)-[/home/dre/Documents/iot_zta_ddos]
└─# pip install --upgrade pip
```

Step 2: Install Libraries and Dependencies

```
┌──(venv)─(root㉿kali)-[/home/dre/Documents/iot_zta_ddos]
└─# pip install -r requirements.txt
```

Step 3: Verify Installations

```
┌──(venv)─(root㉿kali)-[/home/dre/Documents/iot_zta_ddos]
└─# python -c "import flask, requests, psutil, matplotlib, pandas, numpy, scapy; print('All dependencies installed successfully')"
All dependencies installed successfully

┌──(venv)─(root㉿kali)-[/home/dre/Documents/iot_zta_ddos]
└─#
```

4. Project Structure Overview

4.1 Expected Directory Layout

```
iot_zta_ddos/
├── README.md                  # This file
├── requirements.txt           # Python dependencies
├── device.py                  # IoT device simulator
├── controller.py              # Smart home hub simulator
├── baseline_runner.py         # Baseline experiment automation
├── ddos_runner.py             # DDoS attack experiment automation
├── statistical_analysis.py    # Statistical calculations
├── batch_statistical_analysis.py   # Batch Statistical Analysis and Plot
├── tc-netem_ddos_runner.py         # tc-netem emulation for DDoS attack experiment automation
├── network_emulation.sh       # tc-netem emulation shell script
├── analyze_baseline.py        # Baseline data analyzer
├── analyze_attack.py          # Attack data analyzer
├── batch_analyze_attack.py    # Batch analysis utility
├── attacks/
│   ├── http_flood.py          # HTTP GET flood generator
│   └── udp_flood.py           # UDP packet flood generator
├── logs/                      # Generated experiment data
│   ├── baseline_YYYYMMDD-HHMMSS/
│   │   ├── perimeter_baseline.csv
│   │   ├── zta_baseline.csv
│   │   └── zta_aug_baseline.csv
│   └── ddos_YYYYMMDD-HHMMSS/
│       ├── http/
│       └── udp/
└── venv/                      # Python virtual environment
```

4.2 Log File Format

All CSV output files contain the following columns:

- timestamp: Unix timestamp of measurement
- latency: Round-trip time in seconds
- status: HTTP status code (200 = success)
- cpu: CPU utilization percentage
- mem: Memory utilization percentage

5. Security Architecture Configuration

5.1 Perimeter Security (PS)

Configuration: IP whitelist-based access control

- Trust Model: Implicit trust for local network Ips
- Default Whitelist: 127.0.0.1 (localhost)
- Activation: `python device.py --mode perimeter --port 5001`

```
┌──(venv)─(root㉿kali)-[/home/dre/Documents/iot_zta_ddos]
└─# python device.py --mode perimeter --port 5001
 * Serving Flask app 'device'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5001
 * Running on http://10.0.2.15:5001
Press CTRL+C to quit
```

Figure 4: Checking the device and security connectivity

Security Logic:

```
    if args.mode == "perimeter":
        if ip not in args.whitelist:
            return "Forbidden (perimeter)", 403
```

Figure 5: Checking the device and security connectivity

5.2 Zero Trust Architecture (ZTA)

Configuration: Bearer token authentication per request

- Trust Model: "Never trust, always verify"
- Token Requirement: `Authorization: Bearer SECURETOKEN` header
- Activation: `python device.py --mode zta --port 5001`

```
┌──(venv)─(root㉿kali)-[/home/dre/Documents/iot_zta_ddos]
└─# python device.py --mode zta --port 5001
 * Serving Flask app 'device'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5001
 * Running on http://10.0.2.15:5001
Press CTRL+C to quit
```

Figure 6: Checking the device and security connectivity

Security Logic:

```
elif args.mode.startswith("zta"):
    token = request.headers.get("Authorization", "").replace("Bearer ", "")
    if token != args.token:
        return "Unauthorized (ZTA)", 401
```

5.3 ZTA with Rate Limiting (ZTA+RL)

Configuration: ZTA + sliding window rate limit

- Trust Model: Token authentication + request throttling
- Rate Limit: 10 requests per 10 seconds per IP
- Activation: `python device.py --mode zta_aug --port 5001 --rate 10 --window 10`

```
┌──(venv)─(root㉿kali)-[/home/dre/Documents/iot_zta_ddos]
└─# python device.py --mode zta_aug --port 5001 --rate 10 --window 10
 * Serving Flask app 'device'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5001
 * Running on http://10.0.2.15:5001
Press CTRL+C to quit
```

Figure 7: Checking the device and security connectivity

Security Logic:

```
_rate_log[ip] = [t for t in _rate_log[ip] if t >= window_start]
if len(_rate_log[ip]) >= args.rate:
    return "Too Many Requests", 429
_rate_log[ip].append(now)
```

5.4 Device Simulation Parameters

Each simulated device includes:

- Processing Delay: 80-150ms (mimics Wi-Fi + MCU latency)
- Single Threading: `threaded=False` (emulates ESP8266/ESP32 constraints)
- Memory Constraints: Minimal library imports
- Realistic Jitter: Random processing time variation

6. Baseline Experiments

6.1 Baseline Data Collection

Baseline experiments measure normal performance without attacks.

Command Syntax:
python baseline_runner.py --duration <seconds> --port <port>
Standard Baseline (30 seconds per security mode):

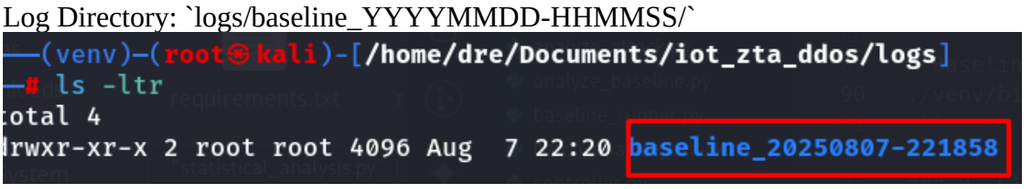
```
┌──(venv)─(root㉿kali)-[/home/dre/Documents/iot_zta_ddos]
└─# python baseline_runner.py --duration 30 --port 5001
```

6.2 Baseline Process Flow

- **Perimeter Security**: Device starts with IP whitelist → Controller sends 1 req/s → Metrics logged
- **ZTA**: Device starts with token auth → Controller sends authenticated requests → Metrics logged
- **ZTA+RL**: Device starts with token + rate limit → Controller sends requests → Metrics logged

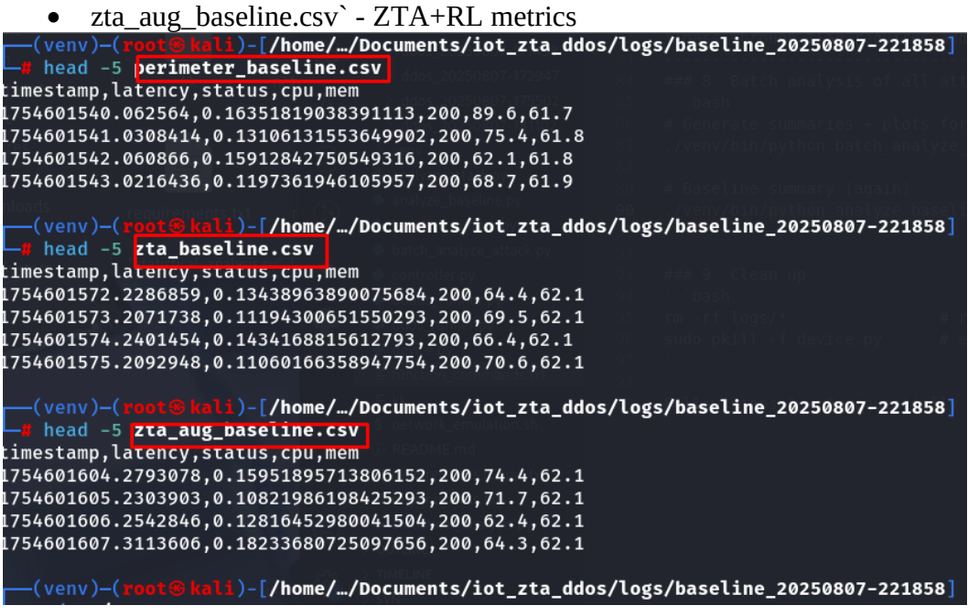
6.3 Expected Baseline Outputs

Log Directory: `logs/baseline_YYYYMMDD-HHMMSS/`



Files Created:

- perimeter_baseline.csv - Perimeter security metrics
- zta_baseline.csv - ZTA metrics
- zta_aug_baseline.csv` - ZTA+RL metrics



Typical Baseline Results After Plots:

Mode	Latency (ms)	Success Rate	CPU Usage
Perimeter	~128.5	100%	~27.6%
ZTA	~123.5	100%	~36.4%
ZTA+RL	~121.5	100%	~28.9%

Table 2: Baseline Values

6.4 Baseline Verification

Success Indicators:

- All CSV files contain 30 rows (1 per second)
- Success rate = 100% across all modes
- Latency values between 80-150ms
- CPU usage stable, not spiking

7. Simulating DDoS Attacks

7.1 Attack Types Overview

HTTP Flood Attack:

- Target: Application layer (Layer 7)
- Method: Multi-threaded HTTP GET requests
- Impact: Overwhelms device request processing

UDP Flood Attack:

- Target: Transport layer (Layer 4)
- Method: Raw UDP packet transmission
- Impact: Tests protocol-level filtering

7.2 Running HTTP Flood Experiments

Standard HTTP Flood (5 minutes or 300 secs, 50 threads):

```
┌──(venv)─(root㉿kali)-[/home/dre/Documents/iot_zta_ddos]
└─# python ddos_runner.py --attack http --thread 50 --duration 300 --port 5001
```

Attack Intensity Sweep (10-100 threads):

```
┌──(venv)─(root㉿kali)-[/home/dre/Documents/iot_zta_ddos]
└─# for for n in 10 25 50 75 100; do
 python ddos_runner.py --attack http --threads "$n" --duration 300 --port 5001
done
```

7.3 Running UDP Flood Experiments

Standard UDP Flood (requires administrative privileges):

```
┌──(venv)─(root㉿kali)-[/home/dre/Documents/iot_zta_ddos]
└─# python ddos_runner.py --attack udp --thread 50 --duration 300 --port 5001
```

7.4 Combined Attack Testing

Both HTTP and UDP:

```
┌──(venv)─(root㉿kali)-[/home/dre/Documents/iot_zta_ddos]
└─# python ddos_runner.py --attack http udp --thread 50 --duration 300 --port 5001
```

7.5 Attack Process Flow

- Device Startup: Security mode configured and health endpoint ready
- Attack Launch: Background flood process starts
- Legitimate Traffic: Controller continues 1 req/s toggle commands
- Metrics Collection: 300 seconds of combined legitimate + malicious traffic
- Cleanup: Attack and device processes terminated

7.6 Expected Attack Outputs

Log Directory: `logs/ddos_YYYYMMDD-HHMMSS/`

```
┌──(venv)─(root㉿kali)-[/home/dre/Documents/iot_zta_ddos/logs]
└─# ls -ltr
total 12
drwxr-xr-x 2 root root 4096 Aug  7 22:29 baseline_20250807-221858
drwxr-xr-x 3 root root 4096 Aug  8 04:18 ddos_20250808-041843
drwxr-xr-x 3 root root 4096 Aug  8 05:48 ddos_20250808-054840
```

HTTP Attack Files

```
http/
├── perimeter_http.csv    # Perimeter under HTTP flood
├── zta_http.csv          # ZTA under HTTP flood
└── zta_aug_http.csv      # ZTA+RL under HTTP flood
```

```
┌──(venv)(root㉿kali)-[/home/dre/Documents/iot_zta_ddos/logs/ddos_20250808-041843/http]
└─# ls -ltr
total 48
-rw-r--r-- 1 root root 15086 Aug  8 04:23 perimeter_http.csv
-rw-r--r-- 1 root root 15303 Aug  8 04:28 zta_http.csv
-rw-r--r-- 1 root root 15728 Aug  8 04:33 zta_aug_http.csv
```

```
┌──(venv)(root㉿kali)-[/home/dre/Documents/iot_zta_ddos/logs/ddos_20250808-041843/http]
└─# head -5 perimeter_http.csv zta_http.csv zta_aug_http.csv
==> perimeter_http.csv <==
timestamp,latency,status,cpu,mem
1754623126.080893,0.25098395347595215,200,92.4,65.3
1754623127.4965234,0.6652178764343262,200,87.3,65.3
1754623128.4686456,0.6318466663360596,200,88.6,65.3
1754623129.4403608,0.6024527549743652,200,87.8,65.4

==> zta_http.csv <==
timestamp,latency,status,cpu,mem
1754623428.6317716,0.2303621768951416,200,78.2,65.1
1754623430.0509517,0.6472632884979248,200,82.4,65.3
1754623431.0643525,0.6588695049285889,200,84.6,65.3
1754623431.92302,0.5168437957763672,200,81.4,65.3

==> zta_aug_http.csv <==
timestamp,latency,status,cpu,mem
1754623731.3018963,0.19636893272399902,200,88.7,65.6
1754623732.8902974,0.7800090312957764,200,92.9,65.6
1754623733.7418199,0.6189558506011963,200,86.1,65.6
1754623734.7392013,0.6154239177703857,200,82.8,65.6

┌──(venv)(root㉿kali)-[/home/dre/Documents/iot_zta_ddos/logs/ddos_20250808-041843/http]
```

UDP Attack Files:

udp/

- perimeter_udp.csv # Perimeter under UDP flood
- zta_udp.csv # ZTA under UDP flood
- zta_aug_udp.csv # ZTA+RL under UDP flood

```
┌──(venv)(root㉿kali)-[/home/dre/Documents/iot_zta_ddos/logs/ddos_20250808-054840/udp]
└─# ls -ltr
total 12
-rw-r--r-- 1 root root 3245 Aug  8 05:49 perimeter_udp.csv
-rw-r--r-- 1 root root 3227 Aug  8 05:50 zta_udp.csv
-rw-r--r-- 1 root root 3178 Aug  8 05:51 zta_aug_udp.csv
```

```
┌──(venv)(root㉿kali)-[/home/dre/Documents/iot_zta_ddos/logs/ddos_20250808-054840/udp]
└─# head -5 perimeter_udp.csv zta_udp.csv zta_aug_udp.csv
==> perimeter_udp.csv <==
timestamp,latency,status,cpu,mem
1754628523.3552778,0.24463438987731934,200,89.5,66.1
1754628524.2538493,0.13886117935180664,200,79.3,65.5
1754628525.2393847,0.12291598320007324,200,87.7,65.6
1754628526.2849298,0.16802000999450684,200,89.4,65.7

==> zta_udp.csv <==
timestamp,latency,status,cpu,mem
1754628587.5986907,0.4525187015533447,200,64.4,66.0
1754628588.2859063,0.13849568367004395,200,83.6,66.2
1754628589.2743347,0.10509657859802246,200,87.1,66.3
1754628590.2988594,0.1266014575958252,200,80.3,66.4

==> zta_aug_udp.csv <==
timestamp,latency,status,cpu,mem
1754628649.9906707,0.11117935180664062,200,65.3,65.8
1754628651.6361349,0.20225071907043457,200,41.2,66.3
1754628652.5630705,0.1289207935333252,200,91.8,66.3
1754628653.571573,0.1363980770111084,200,81.2,66.3
```

Typical HTTP Results After Plot:

Mode	Latency (ms)	Success Rate	Notes
Perimeter	~220.4	100%	No protection against local attacks
ZTA	~244.5	100%	Token validation blocks flood
ZTA+RL	~245.3	95%	Rate limiting causes some false positives

Table 3: HTTP Flood Values

Typical UDP Results After Plot:

Mode	Latency (ms)	Success Rate	Notes
Perimeter	~122.4	100%	OS-level packet filtering
ZTA	~129.7	100%	Minimal impact
ZTA+RL	~129.0	100%	Near-baseline performance

Table 4: UDP Flood Values

7.7 Wi-Fi Latency Emulation 40ms Round Trip Time (RTT)

The use of tc-netem involves configuring the Linux traffic control tool to introduce a fixed 40 ms round-trip delay on network packets, simulating the typical latency experienced over a wireless connection for testing and performance evaluation.

Adding a 40ms delay emulating Wi-Fi Latency in the real world.

```
┌──(venv)(root㉿kali)-[/home/dre/Documents/iot_zta_ddos]
└─# sudo tc qdisc add dev lo root netem delay 40ms
```

```
┌──(venv)(root㉿kali)-[/home/dre/Documents/iot_zta_ddos]
└─# python ddos_runner.py --attack http --threads 50 --duration 300 --port 5001
```

Removing the 40ms delay after using the tc-netem

```
┌──(venv)(root㉿kali)-[/home/dre/Documents/iot_zta_ddos]
└─# sudo tc qdisc del dev lo root netem
```

Adding a 40ms delay emulating Wi-Fi Latency in the real world using a python and shell script.
Make sure the network_emulation.sh is in the same folder as tc-netem_ddos_runner.py

```
┌──(venv)(root㉿kali)-[/home/dre/Documents/iot_zta_ddos]
└─# python tc-netem_ddos_runner.py --attack http --port 5001 --duration 300 --logdir logs --threads 50 --interval 1.0 --netem --delay 40 --loss 0 --jitter 5
```

7.8 Higher Load Testing

Test with increased legitimate traffic.

```
┌──(venv)(root㉿kali)-[/home/dre/Documents/iot_zta_ddos]
└─# python ddos_runner.py --attack http --threads 50 --interval 0.5 --duration 300 --port 5001
```

8. Data Analysis and Visualization

8.1 Baseline Analysis

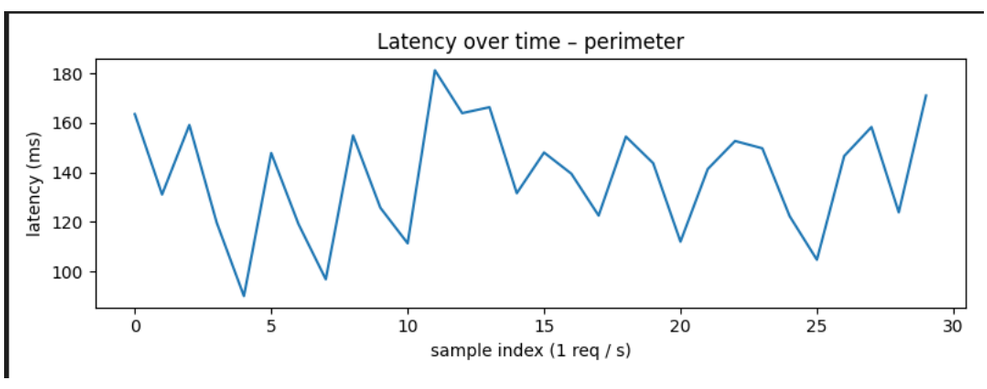
Command:
python analyze_baseline.py --logdir logs/baseline_<timestamp> --plot

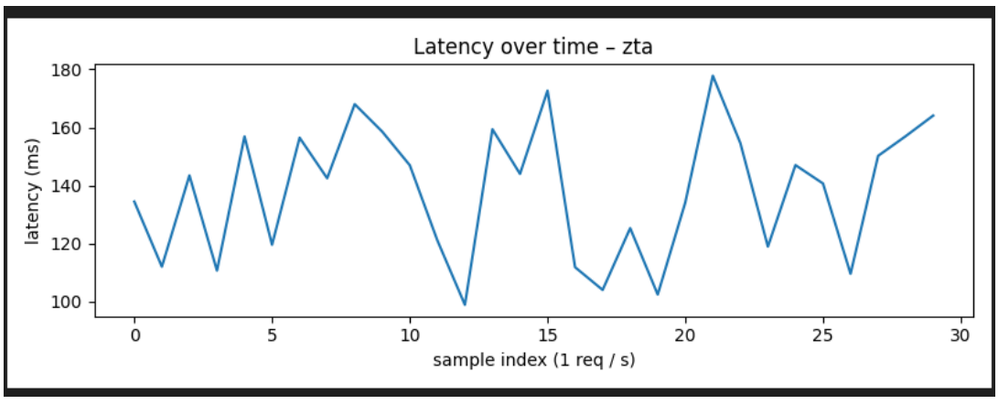
```
┌──(venv)(root㉿kali)-[/home/dre/Documents/iot_zta_ddos]
└─# python analyze_baseline.py --logdir logs/baseline_20250807-221858 --plot
```

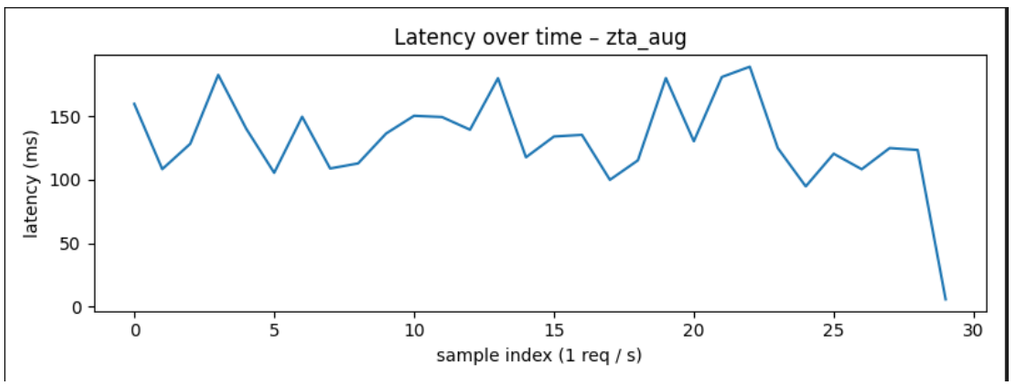
Outputs:

- Console summary table with mean latency, success %, CPU/memory usage
- PNG plots: latency_perimeter.png, latency_zta.png, latency_zta_aug.png

```
┌──(venv)(root㉿kali)-[/home/dre/Documents/iot_zta_ddos]
└─# python analyze_baseline.py --logdir logs/baseline_20250807-221858 --plot
[+] Plot saved: latency_perimeter.png
[+] Plot saved: latency_zta.png
[+] Plot saved: latency_zta_aug.png

Baseline Summary (per security mode):
     mode  samples  latency_mean_ms  latency_std_ms  success_pct  cpu_avg_pct  mem_avg_pct
perimeter       30           138.41           22.94       100.00        70.57        61.85
      zta       30           138.06           22.79       100.00        68.63        62.03
  zta_aug       30           131.07           35.43        96.67        68.02        62.06

Done.
```







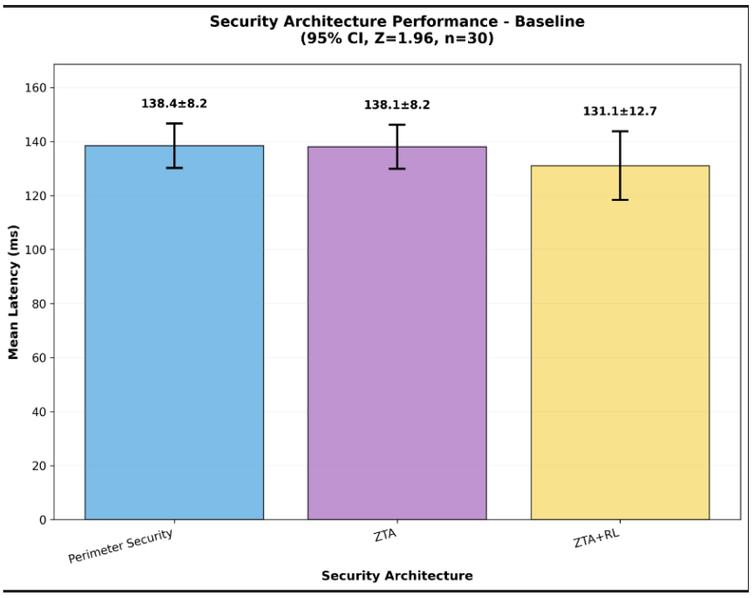
Confidence Interval

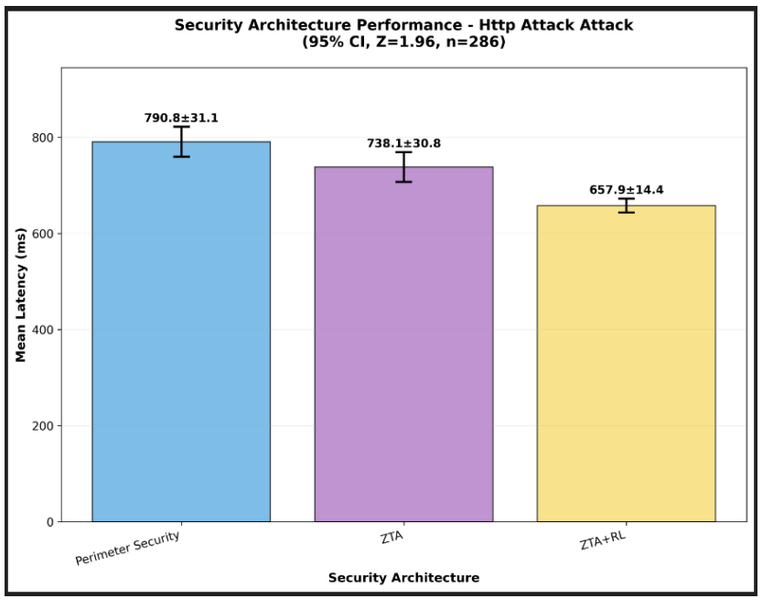
Command:

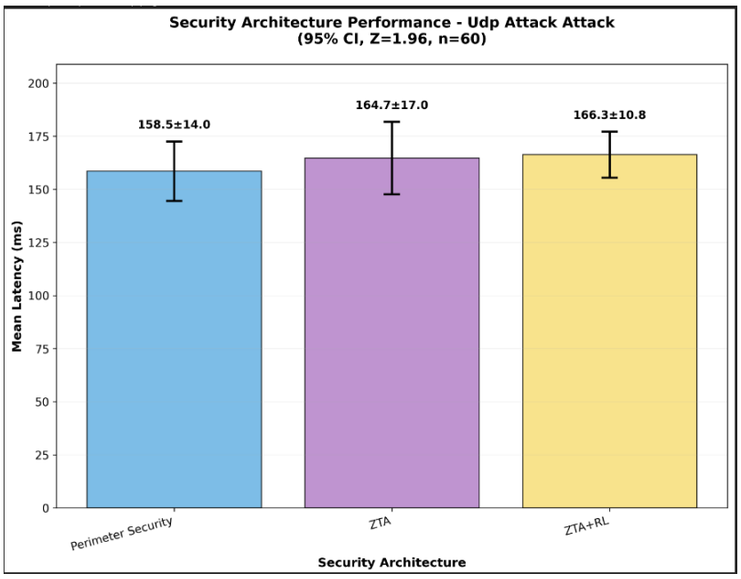
```
┌──(venv)(root㉿kali)-[/home/dre/Documents/iot_zta_ddos]
└─# python confidence_interval_analyzer2.py --baseline logs/baseline_20250807-221858/ --plot
```

Outputs:

- Console summary table with mean latency, success %, CPU/memory usage
- PNG plots: 95% CI Test with a Z-Value of 1.96 for Perimeter Security, ZTA and ZTA+RL. latency_comparison_baseline.png, latency_comparison_http.png and latency_comparison_udp.png







8.2 Attack Analysis

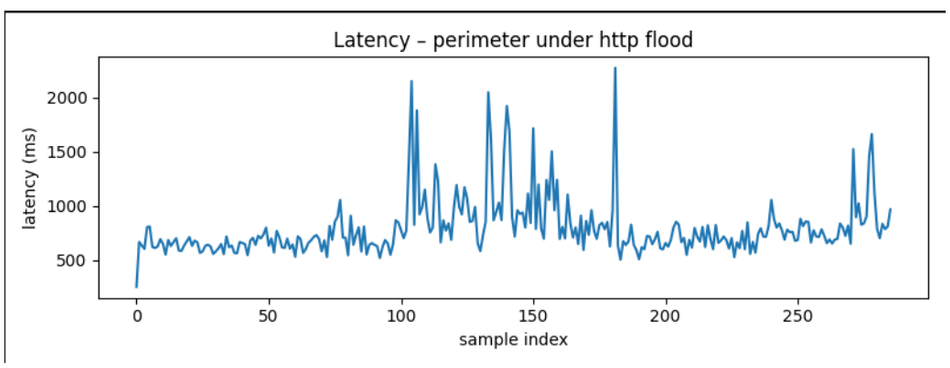
Single Attack Directory:

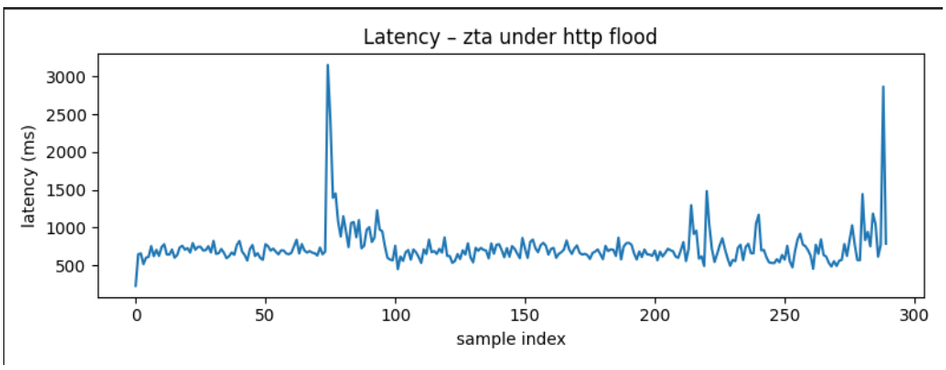
```
┌──(venv)(root㉿kali)-[/home/dre/Documents/iot_zta_ddos]
└─# python analyze_attack.py --dir logs/ddos_20250808-041843/http/ --plot
```

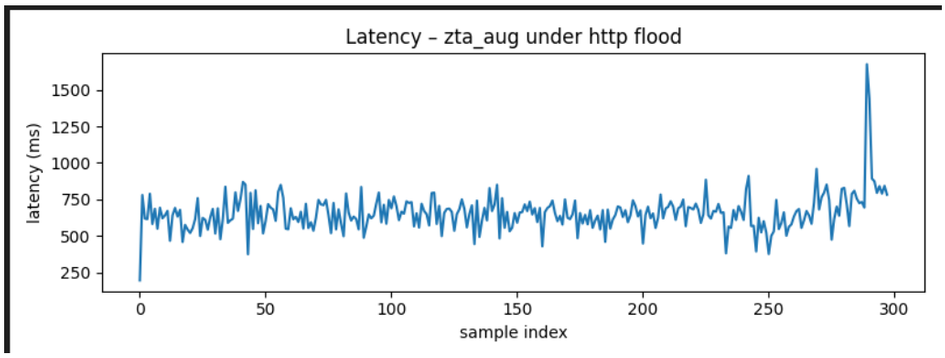
Outputs:

- Console summary table with mean latency, success %, CPU/memory usage
- PNG plots: latency_perimeter.png, latency_zta.png, latency_zta_aug.png

```
┌──(venv)(root㉿kali)-[/home/dre/Documents/iot_zta_ddos]
└─# python analyze_attack.py --dir logs/ddos_20250808-041843/http/ --plot
[+] Plot saved: latency_zta_aug_http.png
[+] Plot saved: latency_zta_http.png
[+] Plot saved: latency_perimeter_http.png

Attack Summary:
attack      mode  samples  latency_mean_ms  latency_std_ms  success_pct  cpu_avg_pct  mem_avg_pct  error_pct
  http   zta_aug      298           657.89          126.92        93.29        83.40        66.01       6.71
  http       zta      290           738.13          267.84        99.66        80.94        65.51       0.34
  http perimeter      286           790.80          268.76       100.00        81.37        65.31       0.00

Done.
```







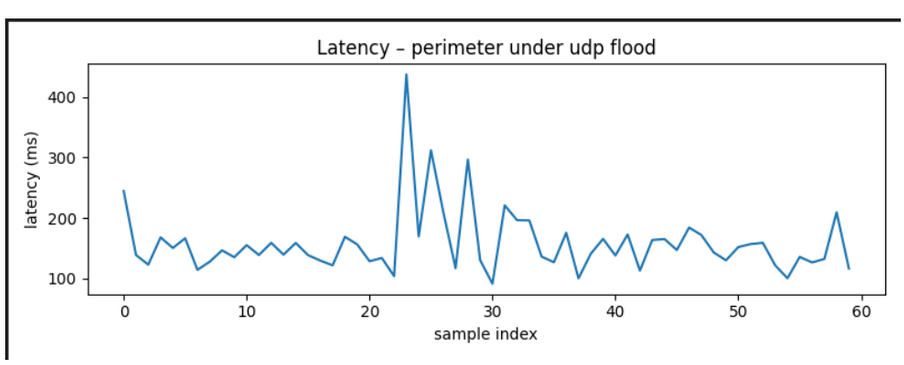
Batch Analysis (All Attacks):

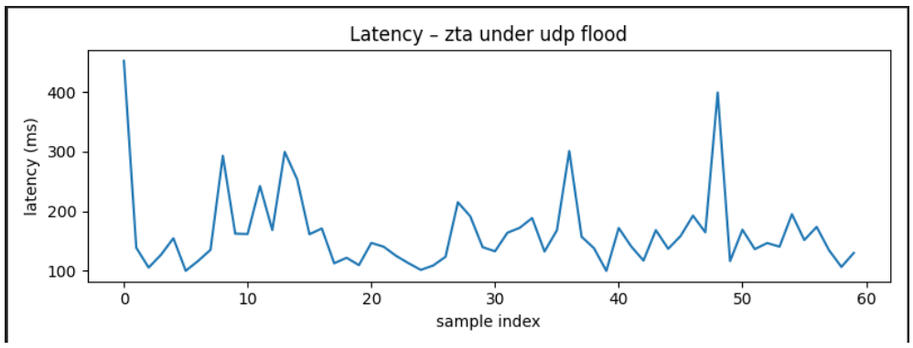
```
┌──(venv)(root㉿kali)-[/home/dre/Documents/iot_zta_ddos]
└─# python batch_analyze_attack.py --root logs --attack http udp --plot
```

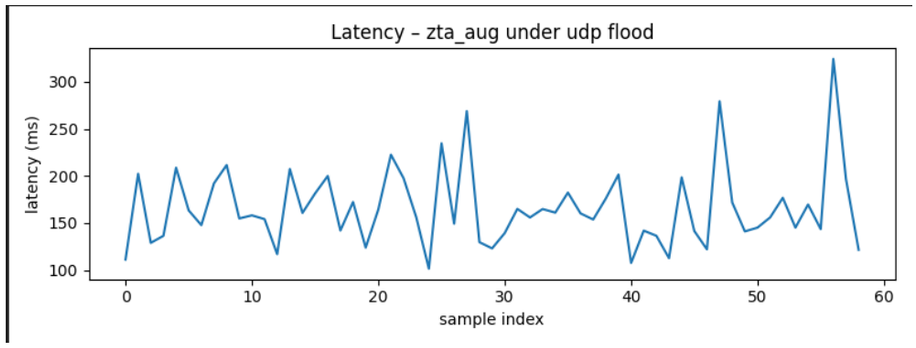
Outputs:

- Console summary table with mean latency, success %, CPU/memory usage
- PNG plots: latency_perimeter.png, latency_zta.png, latency_zta_aug.png

```
┌──(venv)(root㉿kali)-[/home/dre/Documents/iot_zta_ddos]
└─# python batch_analyze_attack.py --root logs --attack http udp --plot

==== logs/ddos_20250808-041843/http ====
[+] Plot saved: latency_zta_aug_http.png
[+] Plot saved: latency_zta_http.png
[+] Plot saved: latency_perimeter_http.png

Attack Summary:
attack      mode  samples  latency_mean_ms  latency_std_ms  success_pct  cpu_avg_pct  mem_avg_pct  error_pct
  http   zta_aug      298           657.89          126.92        93.29        83.40        66.01       6.71
  http       zta      290           738.13          267.84        99.66        80.94        65.51       0.34
  http perimeter      286           790.80          268.76       100.00        81.37        65.31       0.00

Done.

==== logs/ddos_20250808-054840/udp ====
[+] Plot saved: latency_zta_aug_udp.png
[+] Plot saved: latency_perimeter_udp.png
[+] Plot saved: latency_zta_udp.png

Attack Summary:
attack      mode  samples  latency_mean_ms  latency_std_ms  success_pct  cpu_avg_pct  mem_avg_pct  error_pct
   udp   zta_aug       59           166.30           42.33       100.00        80.22        66.14       0.00
   udp perimeter       60           158.54           55.15       100.00        82.09        66.02       0.00
   udp       zta       60           164.71           67.29       100.00        80.80        66.27       0.00

Done.
```







8.3 Generated Visualization

Latency Plots:

- X-axis: Sample index (time progression)
- Y-axis: Latency in milliseconds
- Shows performance degradation under attack

Example Plot Interpretation:

- Baseline: Stable ~120ms with ±20ms jitter
- HTTP Flood: Doubled latency ~240ms with increased variability
- UDP Flood: Near-baseline performance (OS-level filtering)

8.4 Attack Summary

attack	mode	samples	latency_mean_ms	success_pct	cpu_avg_pct
http	perimeter	300	220.4	100	32.2
http	zta	300	244.5	100	39.7
http	zta_aug	300	245.3	95	40

9. Statistical Analysis

9.1 Comprehensive Statistical Evaluation

A python script (statistical_analysis.py) with each metric's formula was used for the calculation of Confidence Intervals, Effect Size and Power Analysis against the logs form the baseline and DDoS attack measurements. T

Without plot visuals

```
┌──(venv)(root㉿kali)-[/home/dre/Documents/iot_zta_ddos]
└─# python statistical_analysis.py --logdir logs/ --baseline baseline_20250807-221858 --attack ddos_20250808-041843
```

With plot visuals

```
┌──(venv)(root㉿kali)-[/home/dre/Documents/iot_zta_ddos]
└─# python statistical_analysis_plots.py --logdir logs/ --baseline baseline_20250807-221858 --http-attack ddos_20250808-041843 --udp-attack ddos_20250808-054840 --plots
```

Batch Attack plot

```
┌──(venv)(root㉿kali)-[/home/dre/Documents/iot_zta_ddos]
└─# python batch_statistical_analysis_plots.py --root logs --attack http udp --plots
```

9.2 Key Performance Metrics Interpretation

Latency Metrics:

- Mean Latency: Average response time in milliseconds
- Standard Deviation (Jitter): Variability in response times
- 95% Confidence Interval: Range containing 95% of measurements

Availability Metrics:

- Success Percentage: HTTP 200 responses / total requests × 100

- Error Percentage: Failed or rejected requests
- False Positive Rate: Legitimate requests blocked by rate limiting

Resource Metrics:

- CPU Utilization: Average processor usage during experiment
- Memory Utilization: Average RAM consumption

9.3 Statistical Measures Calculated

9.3.1 Confidence Intervals (95% CI) (Cumming and Finch 2005)

Formula: CI = $\bar{x} \pm z \times (s / \sqrt{n})$
Where: $\bar{x}$ = mean, z = 1.96, s = std dev, n = sample size

9.3.2 Effect Size (Cohen's d) (Cohen 1988)

Formula: $d = (M_1 - M_2) / \text{Spooled}$
($Spooled = \sqrt{(s^2 + s^2/2)}$)

Interpretation:

- Negligible: $d < 0.2$
- Small: $0.2 \leq d < 0.5$
- Medium: $0.5 \leq d < 0.8$
- Large: $d \geq 0.8$

9.3.3 Statistical Power Analysis (Faul et al. 2007)

Formula:

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

Interpretation:

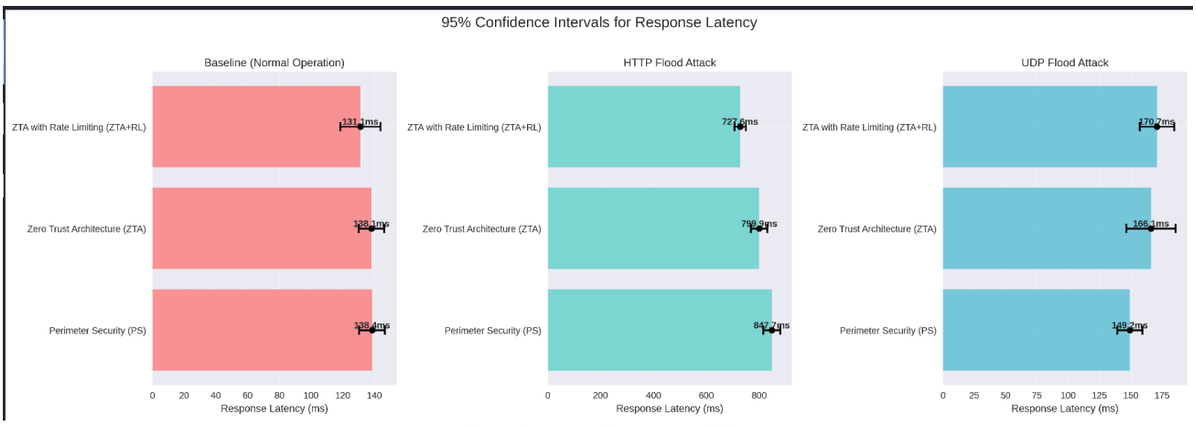
- Low: Power < 0.5
- Moderate: $0.5 \leq$ Power < 0.8
- Adequate: Power ≥ 0.8

Sample Statistical Output

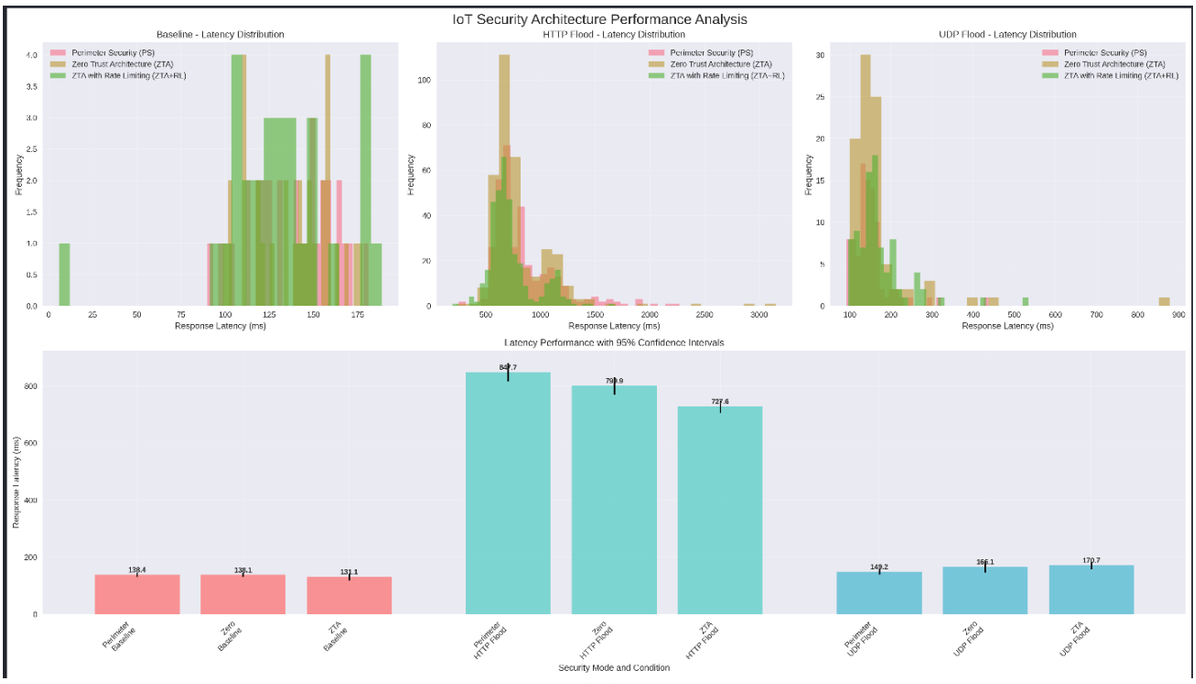
```
================================================================================
CONFIDENCE INTERVAL ANALYSIS (95% Confidence Level)
================================================================================

Baseline (Normal Operation):
--------------------------------------------------
Mode                          Mean (ms)    95% CI Lower 95% CI Upper Margin ±     n
--------------------------------------------------------------------------------
Perimeter Security (PS)   138.41       130.20       146.62       8.21         30
Zero Trust Architecture (ZTA) 138.06       129.91       146.22       8.16         30
ZTA with Rate Limiting (ZTA+RL) 131.07       118.39       143.75       12.68         30

HTTP Flood Attack:
--------------------------------------------------
Mode                          Mean (ms)    95% CI Lower 95% CI Upper Margin ±     n
--------------------------------------------------------------------------------
Perimeter Security (PS)   847.66       815.69       879.62       31.96        336
Zero Trust Architecture (ZTA) 799.92       768.76       831.08       31.16        343
ZTA with Rate Limiting (ZTA+RL) 727.65       705.94       749.36       21.71         352

UDP Flood Attack:
--------------------------------------------------
Mode                          Mean (ms)    95% CI Lower 95% CI Upper Margin ±     n
--------------------------------------------------------------------------------
Perimeter Security (PS)   149.18       139.10       159.25       10.08        90
Zero Trust Architecture (ZTA) 166.06       146.45       185.67       19.61        90
ZTA with Rate Limiting (ZTA+RL) 170.71       156.95       184.47       13.76         89
```

```
================================================================================
EFFECT SIZE ANALYSIS (Cohen's d)
================================================================================

Baseline Conditions:
------------------------------------------------------------
Comparison           Cohen's d    Effect Size     Mean Diff
------------------------------------------------------------
ZTA vs Perimeter     -0.015       Negligible      -0.35
ZTA+RL vs ZTA        -0.235       Small           -6.99
ZTA+RL vs Perimeter  -0.246       Small           -7.35

HTTP Flood Conditions:
------------------------------------------------------------
Comparison           Cohen's d    Effect Size     Mean Diff
------------------------------------------------------------
ZTA vs Perimeter     -0.161       Negligible      -47.74
ZTA+RL vs ZTA        -0.284       Small           -72.27
ZTA+RL vs Perimeter  -0.468       Small           -120.01

UDP Flood Conditions:
------------------------------------------------------------
Comparison           Cohen's d    Effect Size     Mean Diff
------------------------------------------------------------
ZTA vs Perimeter     0.224        Small           16.89
ZTA+RL vs ZTA        0.057        Negligible      4.64
ZTA+RL vs Perimeter  0.370        Small           21.53

================================================================================
STATISTICAL POWER ANALYSIS
================================================================================

Baseline Conditions:
----------------------------------------------------------------------
Comparison           Effect Size  Power        Interpretation
----------------------------------------------------------------------
ZTA vs Perimeter     0.015        0.050        Low (<50%)
ZTA+RL vs ZTA        0.235        0.099        Low (<50%)
ZTA+RL vs Perimeter  0.246        0.103        Low (<50%)

HTTP Flood Conditions:
----------------------------------------------------------------------
Comparison           Effect Size  Power        Interpretation
----------------------------------------------------------------------
ZTA vs Perimeter     0.161        0.317        Low (<50%)
ZTA+RL vs ZTA        0.284        0.754        Moderate (50-79%)
ZTA+RL vs Perimeter  0.468        0.991        Adequate (≥80%)

UDP Flood Conditions:
----------------------------------------------------------------------
Comparison           Effect Size  Power        Interpretation
----------------------------------------------------------------------
ZTA vs Perimeter     0.224        0.186        Low (<50%)
ZTA+RL vs ZTA        0.057        0.058        Low (<50%)
ZTA+RL vs Perimeter  0.370        0.418        Low (<50%)

================================================================================
BATCH ANALYSIS COMPLETE
================================================================================

Interpretation Guidelines:
- Effect Size: Negligible (<0.2), Small (0.2-0.5), Medium (0.5-0.8), Large (≥0.8)
- Statistical Power: Low (<50%), Moderate (50-79%), Adequate (≥80%)
- Confidence Intervals: Narrower intervals indicate more precise measurements
- Success Rate: Percentage of HTTP 200 responses (availability)

Visualization files generated in 'plots' directory:
- confidence_intervals.png: 95% CI comparison across conditions
- effect_sizes.png: Cohen's d effect sizes and interpretations
- power_analysis.png: Statistical power analysis
- comprehensive_summary.png: Complete performance overview
- analysis_report_20250809_203700.txt: Detailed analysis report

Next steps:
1. Review the statistical analysis results above
2. Examine the generated plots (if --plots was used)
3. Check sample sizes to ensure adequate statistical power
4. Consider the practical significance of observed effect sizes

┌──(venv)(root㉿kali)-[/home/dre/Documents/iot_zta_ddos]
└─#
```

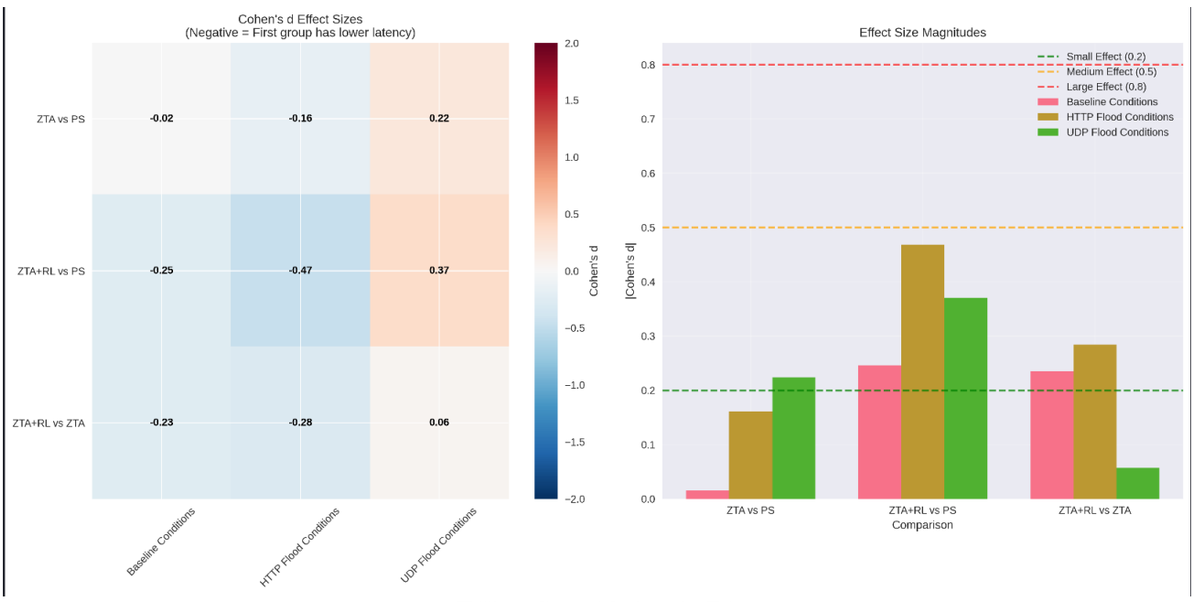
Statistical Analysis



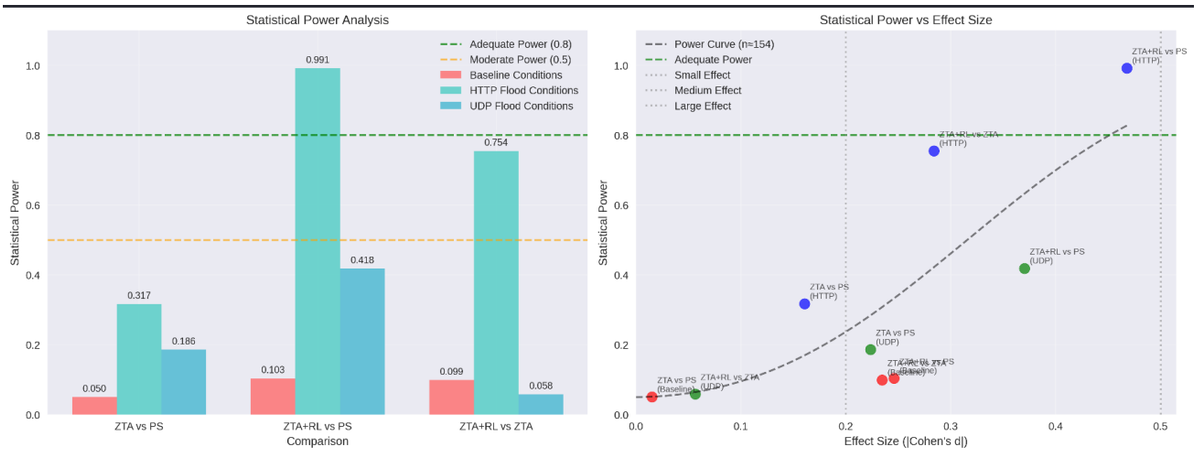
Confidence Interval CI



Statistical Summary



Effect Size (Cohen’s D)



Power Analysis

10. Troubleshooting Guide

Common Installation Issues

Issue: `pip install` fails with permission errors
Solution: Use --user flag
pip install --user -r requirements.txt

Issue: Python version incompatibility
Check version
python --version
Use specific version
python3.9 -m venv venv

Runtime Issues

Issue: **Device fails to start**
Check port availability
netstat -tuln | grep 5001

Try different port
python device.py --mode zta --port 5002

Issue: Health endpoint timeout
Verify device is running
curl http://127.0.0.1:5001/health

Expected response: "ok"

Issue: UDP flood permission denied
Linux/macOS - requires root
sudo python ddos_runner.py --attack udp --duration 300 --port 5001

Or skip UDP testing
python ddos_runner.py --attack http --duration 300 --port 5001

Data Analysis Issues

Issue: No CSV files found
Check log directory exists
ls -la logs/
Verify experiment completed successfully
tail logs/baseline_*/perimeter_baseline.csv

Issue: Plotting fails
Install additional dependencies
pip install matplotlib seaborn

Check file permissions
chmod 644 logs/*/*.csv

Performance Issues

Issue: Experiments run slowly
Cause: Insufficient CPU/RAM
Solution: Reduce duration or thread count
python ddos_runner.py --attack http --threads 25 --duration 180 --port 5001

Issue: Memory exhaustion
Cause: Large dataset accumulation
Solution: Clear logs between experiments
rm -rf logs/*
Verify cleanup
lsof -i :5001 || echo "Port 5001 is free"

Reset network settings
sudo tc qdisc del dev lo root netem 2>/dev/null || true

11. Replication Notes and Limitations

11.1 Simulation vs. Real Hardware

Localhost Testing Rationale:

- Controlled Environment: Eliminates external network variability
- Reproducibility: Consistent results across different systems
- Security: No risk to production networks
- Measurement Precision: Isolates security architecture overhead

Simulation Trade-offs:

- No RF Interference: Real Wi-Fi devices experience airtime contention
- No Hardware Constraints: Python simulation doesn't reflect ESP32/ESP8266 limitations
- Simplified Network Stack: Missing interrupt handling, DMA overhead
- No TLS Overhead: Production ZTA requires encrypted token transmission

11.2 Reproducibility Guidelines

Environment Consistency:

- Use identical Python version across replications
- Maintain same dependency versions from `requirements.txt`
- Run experiments during low system load periods
- Close unnecessary applications during testing

Temporal Considerations:

- Run multiple trials and average results
- Account for system warm-up effects
- Consider time-of-day performance variations

Statistical Validity:

- Minimum 30 samples for reliable confidence intervals
- Multiple experimental repetitions recommended
- Document any anomalous results or system conditions

11.3 Extension Opportunities

Hardware-in-the-Loop (HIL) Testing:

- Deploy on actual ESP32 or ESP8266 hardware
- Measure real Wi-Fi latency and interference
- Evaluate TLS encryption overhead
- Test power consumption under load

Advanced Network Emulation:

- Use tc-netem for Wi-Fi delay simulation
- Implement packet loss and jitter
- Test mesh network topologies
- Evaluate load balancing strategies

Improved Security Models:

- Implement mutual TLS authentication
- Add behavioral anomaly detection

- Test adaptive rate limiting algorithms
- Evaluate machine learning-based threat detection

Primary Sources

- Flask Documentation: https://flask.palletsprojects.com
- Requests Documentation: https://docs.python-requests.org/en/latest
- Scapy Documentation: https://scapy.readthedocs.io
- Psutil Documentation: https://psutil.readthedocs.io
- Matplotlib Documentation: https://matplotlib.org/stable
- Pandas Documentation: https://pandas.pydata.org/docs
- NumPy Documentation: https://numpy.org/doc/stable/
- SciPy Documentation: https://docs.scipy.org/doc/scipy/

References

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.

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.

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.

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.

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.