Experimental assessment of cyber-resilience of cloud-based IoT architectures in critical water supply infrastructure*

Oleksii Novikov[1,†], Yuliia Lehoida[1,†] ,Iryna Stopochkina[1,*,†] and Mykola Ilin[1,†]

[1] *National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Beresteyskyi Ave. 37 03056 Kyiv, Ukraine*

Abstract

The objective of this study is to improve existing tools for modeling cyber-physical attacks on water supply facilities and to refine methodologies for testing the resilience of cloud platforms interacting with such facilities against denial-of-service (DoS) attacks. The paper proposes mathematical models for DoS attacks, specifically DDoS with stochastic noise, timeout flood, and resource exhaustion and combined attacks. A testing methodology was developed, involving the generation of traffic profiles based on these mathematical models using the k6 tool, alongside a hardware-software testbed that simulates a pump station operation based on the ESP8266 controller. The system was evaluated across various scenarios: direct local connection, the use of the Blynk consumer-grade cloud platform, and the industrial-grade ThingsBoard platform. Resilience during load testing was assessed by measuring latency, error rates, and system recovery time. The findings indicate that the computational limits of edge devices are secondary to the backend architecture in determining overall system resilience. Testing revealed different scenarios during the most complex attacks for different platforms. Blink platform showed denial of service (15,000 ms latency, 62.15% errors) for 14,800 virtual users, or stable operation (510.8 ms latency, negligible percentage of errors) in a critical mode for Thingsboard with 58,500 virtual users. These discrepancies are attributed to the different internal protection mechanisms of the platforms; specifically, asynchronous data processing and the use of message queues are identified as critical factors in protecting critical infrastructure from resource exhaustion attacks. The results obtained in this study, including the proposed testbed, are characterized by ease of reproducibility and an orientation toward open-source testing solutions, which can be readily implemented by critical infrastructure operators for resilience audits of cloud-integrated systems.

Keywords

IoT, cyber-resilience, DoS attacks, cloud-based architecture, water supply.

1. Introduction

The uninterrupted functioning of water supply facilities is an urgent necessity for the successful life of society. Modern water supply facilities may incorporate cyber-physical systems characterized by complex architecture, which can become targets for cyber-physical attacks by malicious actors [1].

Internet of Things (IoT) technologies are widely utilized in the water supply sector for water quality monitoring, resource consumption optimization, and timely leak detection [2]. A contemporary trend that has become prevalent at critical infrastructure facilities is the offloading of certain monitoring and analytical functions to the cloud level. In this case, user access is implemented not directly to the equipment, but to the analytical results on the cloud service dashboard. Such an architecture increases information resilience against local failures that may

*ITS-2025: Information Technologies and Security, December 11, 2025, Kyiv, Ukraine

* Corresponding author.

† These authors contributed equally.

o.novikov@kpi.ua (O. Novikov); lehoida.yuliia@lll.kpi.ua (Y. Lehoida); i.stopochkina@kpi.ua (I. Stopochkina), m.ilin@kpi.ua (M. Ilin)

0000-0001-5988-3352 (O. Novikov); 0009-0006-2464-2835 (Y. Lehoida); 0000-0002-0346-0390 (I. Stopochkina), 0000-0002-1065-6500 (M. Ilin),

© 2025 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

published 2026-11-26

CEUR-WS.org/Vol-4337/paper3.pdf

occur directly at the facility; however, at the same time, it possesses specific properties that critical infrastructure operators must take into account. When selecting and configuring interactions with a cloud platform, it is essential to study its operational features and, particularly, its response to critical situations caused by denial-of-service (DoS) attacks [3].

Such attacks may occur intentionally, when a malicious actor gains control over IoT devices that transmit data to the cloud, or over user devices accessing the information and analytical functions of the dashboard. However, these situations can also arise unintentionally. In modern realities, such as those currently existing in Ukraine during large-scale blackouts, a significant amount of hardware and software in water supply systems becomes disconnected. Upon power restoration, all these devices simultaneously begin connecting to standard services, transmitting locally accumulated data, including to cloud monitoring platforms. This triggers a simultaneous surge in operational activity and leads to an unintentionally caused total or partial denial of service. Consequently, the edge devices of water supply facilities do not receive control instructions from the cloud level in a timely manner, and their operation may be disrupted. Another critical factor is the presence of interdependencies between critical infrastructure objects, which leads to cascading failures and acquires a global impact on the national infrastructure [4].

Under such conditions, it is crucial to have tools and methodologies for testing one's own cloud infrastructure for resilience against various types of denial-of-service attacks. In this case, assessing the system's response to different attack scenarios is only possible experimentally. Since using actual IoT equipment in the required volume is unfeasible, the development of specialized testbeds capable of emulating the operation of a real system and generating the necessary volume of requests to cloud services is of current relevance.

Thus, research dedicated to the creation of hardware and software solutions for modeling and implementing specific types of denial-of-service attacks, as well as the methodology that embodies these solutions, constitutes a relevant task.

2. Modifications

In [5], interdependent energy and water supply systems are examined. The focus of the study is the detection of False Data Injection attacks; among others, the authors propose considering attack types such as Pulse, Random, Ramp, and Scaling. Inherently, these attacks are treated as data manipulation; however, the effect of potential denial-of-service (DoS) is not addressed. In [6], various attack models for energy facilities were proposed, including those leading to denial of service. The article focused on identifying attack parameters, but the role of attacks during interaction with cloud services and the corresponding data analysis were not covered.

Study [7] is centered on Denial-of-Service attacks, but it examines network-layer attacks within a local facility. Interaction with cloud platforms is not analyzed. Work [8] presents mathematical models and methods for executing attacks on a range of industrial control systems (ICS). However, no emphasis was placed specifically on denial-of-service attacks, nor was the practical context of executing these attacks considered.

In [9], an IoT simulation-based testing methodology necessary for cloud system stress testing is provided. An architecture for conducting relevant load tests, capable of representing large volumes of IoT devices, is proposed. This architecture simulates transport devices, and a specialized language was developed to facilitate modeling. This approach highlights the importance of analyzing the quality not only of critical infrastructure objects but also of the cloud platforms they interact with. However, for simpler testing scenarios, the proposed architecture is redundant.

In [10], the performance of certain cloud platforms used for monitoring IoT device behavior was investigated. The raw data for analysis were obtained using web monitoring and analytics tools;

specifically, the collected data include response time, throughput, and traffic volume. However, the operation of these platforms under cyberattack conditions was not investigated.

The review article [11] establishes that experimental research is a common method for evaluating cloud platform performance. Reliability and fault tolerance are subjects of study in several papers, although they are underrepresented (8.4% of the total volume of work) compared to other assessment aspects. Resilience to attacks intentionally provoked by a malicious actor is not separately mentioned in this review; however, it can be assumed that this area is a subset of general fault tolerance. Among the shortcomings of existing works, it is noted that most IoT systems presented are not publicly accessible, and therefore, the reproducibility of the results obtained by the authors is problematic.

In [12], a platform was proposed for stress testing and reproducing complex cyber-physical attacks on water supply systems that can affect water quality. The focus of the work is on the operation of water utilities and their response to harmful impacts. The issue of cloud platforms was not addressed in the study.

From the conducted review, it can be concluded that a gap exists between the results highlighted in the literature and the urgent needs in the field of researching DoS and DDoS attacks for water supply monitoring systems that utilize cloud platforms.The goal of this work is to improve existing tools for modeling cyber-physical attacks on water supply facilities, as well as methodologies for testing the resilience of cloud platforms interacting with such facilities against denial-of-service attacks. This goal is achieved by addressing the following tasks:

1. To extend mathematical models of DoS and DDoS attacks to other classes of implementation that may occur in water supply systems, for the purpose of integration into experimental testing.
2. To propose a laboratory testbed configuration for conducting experiments, consisting of publicly available components and ensuring reproducibility. The testbed must model the operation of a simplified version of a water level control system in reservoirs.
3. To develop a testing methodology.
4. To perform a comparative analysis of the response of well-known open cloud platforms to the conducted testing.

3. Results of a study on the resilience of Internet of Things monitoring platforms to denial of service attacks

This section presents the main results of solving the tasks of the work, in particular, mathematical models of DoS and DDoS attacks are proposed, the design of the proposed laboratory stand is described, the testing methodology is presented, and a comparative analysis of the results is performed based on the Blynk [13] and ThingsBoard [14] platforms.

3.1. Mathematical models of DoS, DDoS attacks

Among different DoS and DDoS cyberattacks, a number of types can be distinguished for water supply facilities, listed in Table 1.

Table 1
Types of Denial-of-Service cyberattacks in water supply

Attack type, Description	Influence details	Consequences for physical processes

DoS (Denial of Service). Overloading a service or resource with an excessive requests	Simple to implement, can be targeted at controllers, sensors or software	Stopping pumping stations, sensor failure, controller malfunction
DDoS (Distributed DoS). Mass requests from numerous attacking nodes	Harder to deflect due to distributed load	Large-scale failures in process control, disruption of network interaction
Pulse. Short but intense load spikes	Sudden load peaks, difficult to predict	Temporary failures of controllers and sensors, disruption of real-time control
Ramp. Gradual increase in load	Hidden resource depletion	Slow degradation of pumps, controllers difficult to detect
Timeout flood. Calling timeouts in request processing	Disrupts the synchronization of physical processes	Emergency situations, delays in water or energy supply
Resource exhaustion Exhaustion of key resources (memory, CPU, communication channels)	Blocks system resources, causes inpredictable crashes	Long-term shutdowns of pumping stations, sensor failure, equipment damage

We propose models of cyberattacks for water supply facilities, which are given in Table 1.

Classic DoS attack model. It leads for instantaneous unavailability, it is best modeled as a Heaviside function. Unlike Pulse, it does not end after Δt, but continues continuously.

$$\lambda(t)=\begin{cases}\lambda_0, t<t_0\\ \lambda_{attack}, t\geq t_0\end{cases}, \quad (1)$$

where λ_0 – normal traffic level, λ_{attack} – attack level, which exceeds the system bandwidth ($\lambda_{attack}\gg Threshold$), t_0 – initial moment of the attack.

DDoS attack model. The main difference in DDoS is the sources (“multiple attacking nodes”). Mathematically, it is the sum of many flows. If we model the total traffic at the victim's input, it looks like the sum of the functions of individual bots, often with the addition of stochastic (random) noise, since the requests are not perfectly synchronized. The appropriate function can be represented as

$$\lambda(t)=\begin{cases}\lambda_0, t<t_0\\ a+\xi(t), t\geq t_0\end{cases}, \quad (2)$$

where N be the number of attack bots; a be total average attack amplitude, $\xi(t)$ be random noise function (because DDoS traffic is less stable than DoS from a single source).

Timeout flood attack model. The goal of this attack is to cause timeouts. The attacker does not have to keep the traffic high all the time, and needs to send a new batch of requests exactly when the previous ones “fall off” due to timeout or the server releases resources. This can be described by a periodic function (a series of pulses).

$$\lambda(t)=\lambda_0+\sum_{n=1}^{M} af\left(\frac{t-(t_0+nT)}{\tau}\right), \tag{3}$$

where a be attack amplitude, T be attack period, which correlates with resources recover, τ be longitude of single burst of requests, f be rectangle-shaped impulse function.

Resource exhaustion model. It was used linear function for ramp attack pattern. For resource exhaustion it was proposed to use exponential function. It explains the situation, when every next stage of the attack consumes resources faster (for example, as in recursive queries), leading to a crash faster than linear growth:

$$\lambda(t)=\begin{cases}\lambda_0, t<t_0\\ \lambda_0 e^{\alpha(t-t_0)}, t\geq t_0\end{cases}, \tag{4}$$

where α be load growth aggressiveness coefficient, which indicates resource depletion rate.

Combined attack (pulse and exponential ramp). This is the most dangerous scenario for control systems, as linear growth reconfigures adaptive protection systems, and shifts the baseline, then pulses break through protection at peak moments. Mathematically, this is described as follows:

$$\lambda(t)=\begin{cases}Ae^{\alpha(t-t_0)}, t\in[0,T]\setminus[t_{p_i}, t_{p_i}+\Delta t], i=0,1,\ldots;\\ Be^{\beta(t-t_0)}, t_{p_i}\leq t\leq t_{p_i}+\Delta t;\end{cases} \tag{5}$$

where α be load growth aggressiveness coefficient for baseline traffic growth, β be load growth for intervals of additional pulse attack action, $B>A$ be constant values.

3.2. Creation of a laboratory testbed for modeling attacks according to given models and corresponding methodology

The proposed testbed (Fig. 1) consists of three primary hardware components: the ESP8266 microcontroller, an ultrasonic distance sensor, and a relay module. The general wiring scheme is described below.

Ultrasonic Sensor (HC-SR04 or equivalent): VCC → 5V; GND → GND; Trig → GPIO (e.g., D5); Echo → GPIO (D6) via a voltage divider; since the ESP8266 operates at 3.3V while the Echo output is 5V.

Relay Module: VCC → 5V; GND → GND; IN → GPIO (e.g., D1).

ESP8266 (NodeMCU or Wemos D1 Mini): power supply 5V or USB; all GND pins are connected to a single common point; the GPIO pins are selected to ensure the correct operation of the sensor and the relay without conflicts. I2C LCD Display is used to display information about the distance to the water surface. At the logical level, the interaction functions as follows: the ultrasonic sensor periodically transmits data to the microcontroller, the ESP8266 generates telemetry and sends it to the cloud platform (Blynk or ThingsBoard), and in response, it can receive commands to switch the relay on or off, modeling the control of physical equipment.

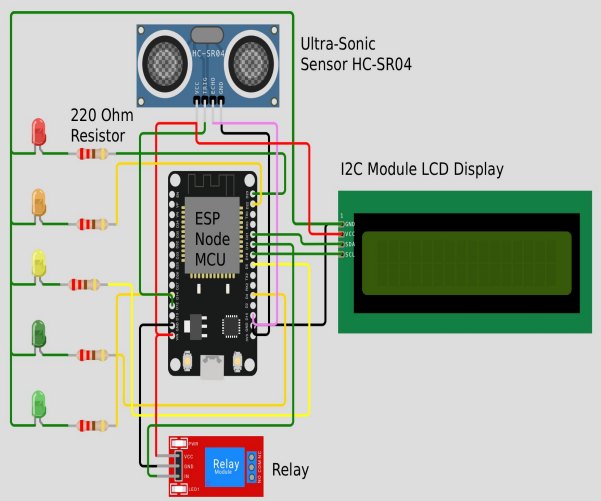


Figure 1: Diagram of a testbed for simulating an IoT node inside a water supply system

The controller performs periodic sensor readings, generates valid MQTT or HTTP messages (depending on the selected platform), transmits them to the cloud system, and processes the responses. Consequently, the testbed behaves as a typical device in water supply systems, industrial automation, and critical infrastructure monitoring.

3.3. Methodology for testing Internet of Things platforms for resilience to cyberattacks

Stage 1. The testing methodology is applied in interaction with a predefined platform. In this study, the Blynk platform was used as a cloud service for system monitoring and control. The Blynk server was targeted for DoS attacks. A specialized Python script was utilized to generate the DoS request stream, simulating a high volume of requests to the Blynk API server to evaluate the system's load resilience. A baseline test was conducted to establish normative indicators for further comparison. Testing was performed both manually and using a custom-developed Python script. This allowed for the determination of normal system response times (latency) under minimal load. The local latency on the ESP8266 was approximately 26–30 ms, while the total end-to-end latency was about 135 ms. These values serve as the reference point for assessing the impact of DoS attacks. To evaluate system stability, a series of tests were conducted with varying load intensities: 1, 3, and 5 requests per second (RPS).

Stage 2. To assess the resilience of edge-level devices–in this case, the microcontroller itself–a stage was implemented where requests were sent directly to the ESP8266, bypassing the Blynk cloud. In the cloudless mode, each request from the script became an individual HTTP connection directly with the ESP8266. Unlike a cloud server, this microcontroller is not designed to handle a large number of simultaneous network requests. Each request to toggle the pump forces it to perform a significant number of resource-intensive operations, such as establishing a connection, processing data, executing the command, and sending a response. These operations are executed sequentially at high request frequencies (e.g., 3 or 5 RPS).

Stage 3. Modeling of attacks was carried out by emulating activity directed at the cloud-level data centers of Blynk Cloud and ThingsBoard, which serve the pump station's IoT system. Implementing a load equivalent to a large number of IoT devices allows for the assessment of data center resilience against DDoS attacks. Under real-world conditions, such a system contains a limited number of sensors and detectors transmitting low-intensity telemetry, along with occasional user control commands. However, within the experiment, a scenario was modeled

where an attacker gains control over the pump station microcontroller and intentionally generates excessive traffic. In this case, even a single compromised device can behave as if a large number of IoT nodes are operating simultaneously, generating high-frequency requests. This specific scenario formed the basis for modeling attacks on the cloud data center. The open-source load testing tool k6 [15] was used to generate excessive traffic. The load generation profile was configured according to the Pulse and Ramp attack models and the proposed equations (1)–(5).

Metrics used. The following metrics were proposed to evaluate the experimental data according to the methodology: VU_{max} - the maximum number of simultaneously active virtual users or devices that the compromised microcontroller simulates; Gen_{req} - this is the total number of requests made from these devices to the cloud; Errors (%) - displays the proportion of requests or messages that were not successfully processed by the system; p95 latency (ms) and p99 latency (ms) indicators demonstrate response times that did not exceed the 95th and 99th percentiles, respectively, allowing for an assessment of latency in peak situations. CPU (%) and RAM (%) indicators show maximum CPU and RAM usage during the test, and Recovery_time (s) indicates how many seconds it takes for the system to return to a stable state after being loaded under attack.

3.4. Results of applying the testing methodology on the Blynk and ThingsBoard platforms

Stage 1 Results. The end-to-end latency remained virtually unchanged even under maximum load. The average latency stayed within the range of 120–130 ms. Furthermore, the success rate of executed commands was 100%. This is explained by the fact that Blynk Cloud servers feature an integrated rate-limiting mechanism, which controls the number of requests a user or device can send to the server or API within a specific timeframe. The Blynk cloud platform applied rate limiting to restrict the number of requests from our device per unit of time. Consequently, even when an excessive number of requests were sent, as during a DoS attack, the cloud queued or rejected them, allowing the end device to remain operational and avoid overloading.

Stage 2 Results. Without the cloud, each request from the developed script became a separate HTTP connection directly to the ESP8266. Since these operations are performed sequentially, at high request frequencies (e.g., 3 or 5 RPS), the device fails to respond in time, which causes an increase in latency (Fig. 2). It should be noted that while latency increased, the packet loss rate remained at zero.

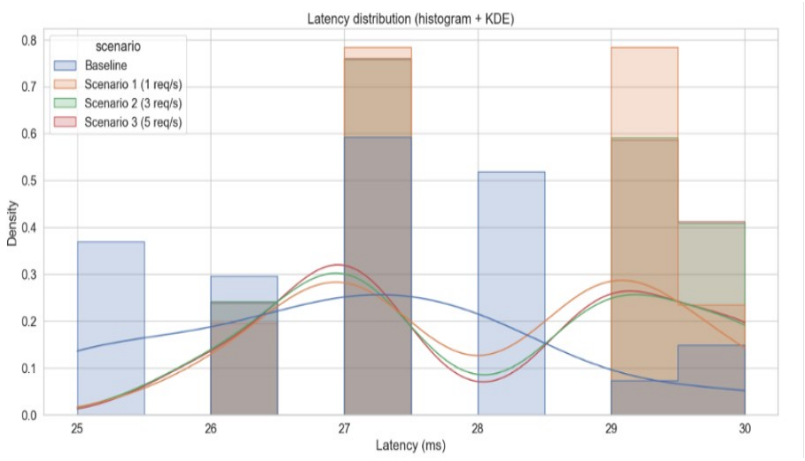


Figure 2: Histogram with Kernel Density Estimation score for different load scenarios

This indicates that the ESP8266, although under significant load, managed to process each request before the script recorded a timeout. This demonstrates that its resources were still sufficient for sequential request processing despite substantial delays. Thus, instead of simply dropping packets, the device was forced to place them in an internal queue, which manifested as increased latency.

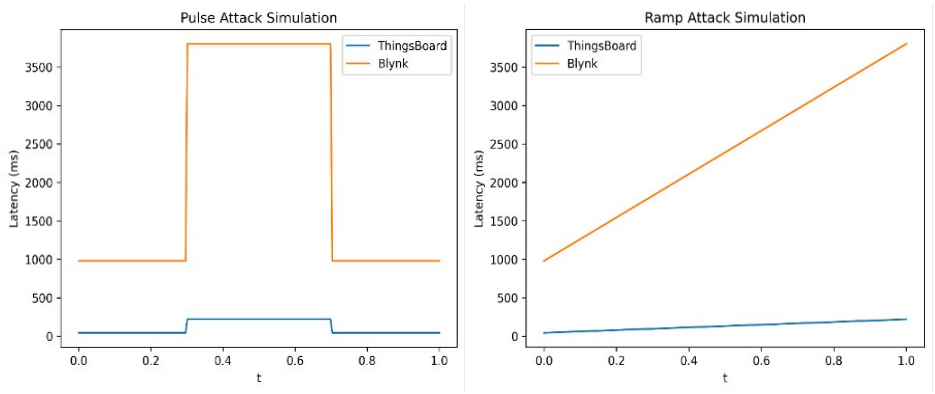
Stage 3 Results. As a result of testing conducted on the ThingsBoard and Blynk platforms, the metric values presented in Table 2 were obtained.

On the ThingsBoard Cloud platform, the *Pulse attack* modeling showed a more resilient response from the data center. In the *Ramp scenario* on ThingsBoard, the compromised device gradually increased its activity. For ThingsBoard Cloud, the maximum number of VUs was 50,000 for both Pulse and Ramp scenarios, as the platform is designed to process large streams of simultaneous messages thanks to its asynchronous architecture and built-in queues [16], allowing for much more efficient load scaling without compromising operational stability.

Table 2
Metrics comparison in experiments with ThingsBoard and Blynk

Platform	Metrics	Pulse attack	Ramp attack
ThingsBoard	VU_{max}	50,000	50,000
	Gen_{req}	1,500,000	1,032,000
	Errors (%)	0.12%	0.03%
	p95 / p99 latency (ms)	45 /220	40 / 200
	CPU (%)/ RAM (%)	85 / 75	75 /78
	Recovery time (s)	35	30
Blynk	VU_{max}	2,000	1,000
	Gen_{req}	324,500	222,000
	Errors (%)	3.6%	1.8%
	p95 / p99 latency (ms)	980 / 3,800	900 / 4,200
	CPU (%) /RAM (%)	94 / 90	92 / 86
	Recovery time (s)	55	45

The latency profile under the influence of Pulse and Ramp attacks on the considered platforms is shown in Fig. 3 and Fig. 4, respectively.



a) **b)**

Figure 3: Platform-based delays under a) Pulse attack; b)Ramp attack

The experimental results indicate that the compromise of a single microcontroller in a real-world IoT pumping station can generate a load that significantly exceeds the normal operating mode, thereby posing a tangible threat to the cloud data center.

As a result of testing with a DDoS attack profile containing a stochastic component, according to equation (2), the metric values presented in Table 3 were obtained.

The results obtained are explained by the fact that different platforms use different approaches as protection mechanisms. In particular, ThingsBoard Cloud uses message queues (Rule Engine), and Blynk Cloud uses the Rate Limiting mechanism. The architecture with queues turned out to be more resistant to attacks with model (2).

Table 3
Stochastic DDoS attack with ThingsBoard and Blynk metrics

Platform/ Metrics	VU_{max} / Gen_{req} (req/sec)	p95 latency (ms)	Errors (%)
ThingsBoard	17,250 / 86,000	28.4 (stable)	0.00%
Blynk	4,120 / 20,000	3,850.12 (critical)	3.84%

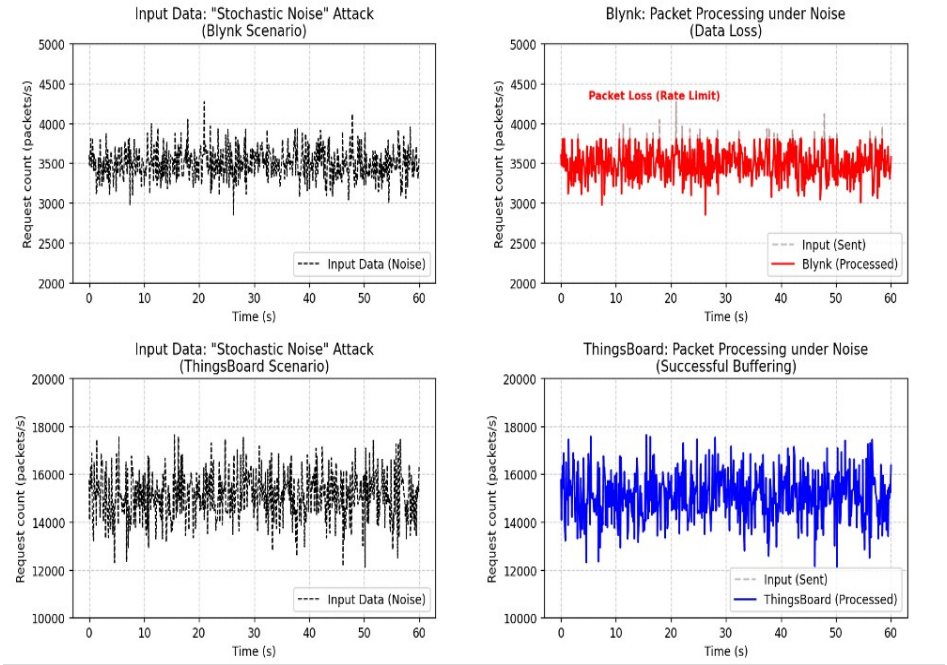


Figure 5: Differences in handling stochastic traffic by cloud platforms

The effects of stochastic traffic processing by cloud platforms are illustrated in Fig.5. The top row, which describes the behavior of Blynk, shows the following: the right graph clearly displays a clipping effect. The processed traffic is capped at a level of 3,800 requests per second (RPS), while the incoming data exceeds this value. All noise peaks exceeding the cutoff line represent lost data (HTTP 429 Too Many Requests error), which leads to instability in pump control.

In Fig. 5, the bottom row, which describes the behavior of ThingsBoard, shows: on the right graph, the blue line completely overlaps the black dashed line. Despite the load being four times higher (15,000 vs. 3,500), ThingsBoard processed all stochastic bursts without loss due to buffering.

The Periodic Timeout Flood model (equation (3)) was utilized for the subsequent experiment. An analysis of the ThingsBoard Cloud logs revealed that the platform successfully amortized a peak load of 25,000 users thanks to its queuing mechanism. Instead, the Blynk Cloud log showed a critical vulnerability of the architecture to pulse attacks. The metrics for this experiment are given in Table 4.

Table 4
Periodic Timeout Flood attack with ThingsBoard and Blynk metrics

Platform/ Metrics	VU_{max} / Gen_{req} (req/sec)	P95 latency (ms)	Errors (%)	Recovery time (s)
ThingsBoard	25,000 / 125,000	195 (controlled)	0.00%	Instantly (after processing the queue)
Blynk	6,000 / 30,000	5,200 (Error503/504 Gateway timeout)	12.5%	5-7

The effects of cloud platforms handling periodic DDoS attack traffic are illustrated in Fig. 6.

Top row Fig. 6 (Blynk) shows the cut-off peaks of the pulses . This is the area of data loss (Error 503) when the connection pool is exhausted. The red fill shows the amount of information lost.

Bottom row Fig. 6 (ThingsBoard) illustrates following: the blue line completely repeats the incoming signal, this means that the platform accepted all 100% of the packets, even during peak loads (25,000 requests), using buffering.

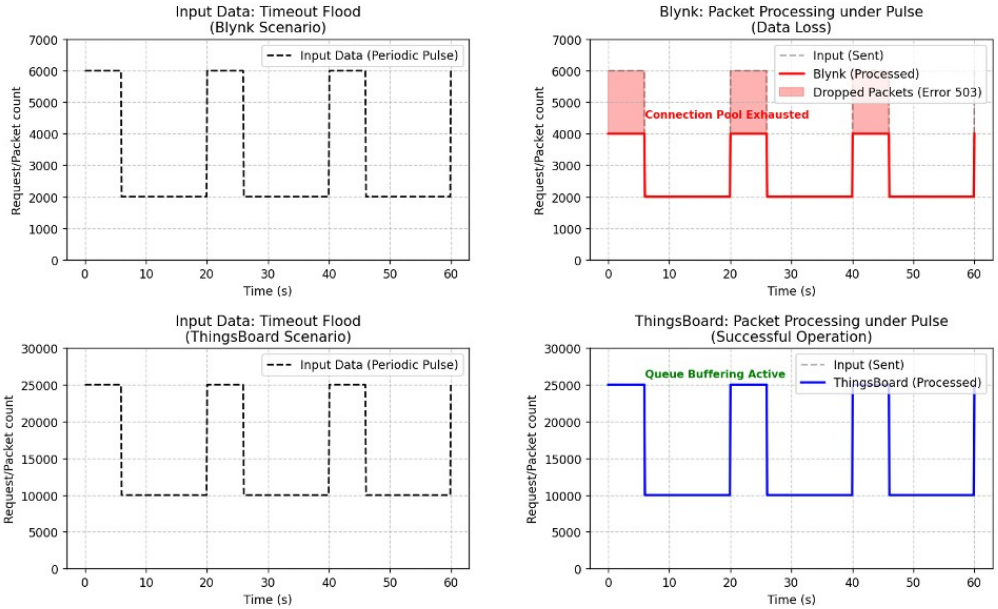


Figure 6: Handling of periodic Timeout Flood traffic by different cloud platforms

The following experiment illustrates the impact of the Resource Exhaustion model, represented by the ratio 4.

Analysis of the ThingsBoard Cloud logs showed that with an exponential increase in load to VU_{max}= 45,100 the system reached peak resource utilization (CPU 98%, RAM 92%). In response to the critical heap memory filling (Heap > 85%), the Garbage Collector and Backpressure mechanisms were triggered, which caused an increase in latency to 410 ms, but allowed avoiding packet loss (0% errors) and maintaining the service's performance.

The Blynk Cloud log showed that when reaching VU_{max}=5,800 Worker threads were completely exhausted (Worker threads exhausted). This led to an avalanche-like increase in latency to 15 seconds and subsequent halt in request processing (Requests/sec=0) with massive generation of 504 Gateway Timeout errors. The total percentage of lost data was 42.50%, indicating a system crash. The main metrics are summarized in Table 5.

Table 5
Resource Exhaustion attack with ThingsBoard and Blynk

Platform/ Metrics	VU_{max}	CPU/ RAM (%)	Errors (%)	Reaction on stress test
ThingsBoard	45,100	98% / 92%	0.00%	Garbage Collection, Backpressure
Blynk	5,800	N/A	42,5%	Thread exhausion and

	(Crash)	timeout

The resilience of platforms to exponential load growth is shown in Fig. 7.

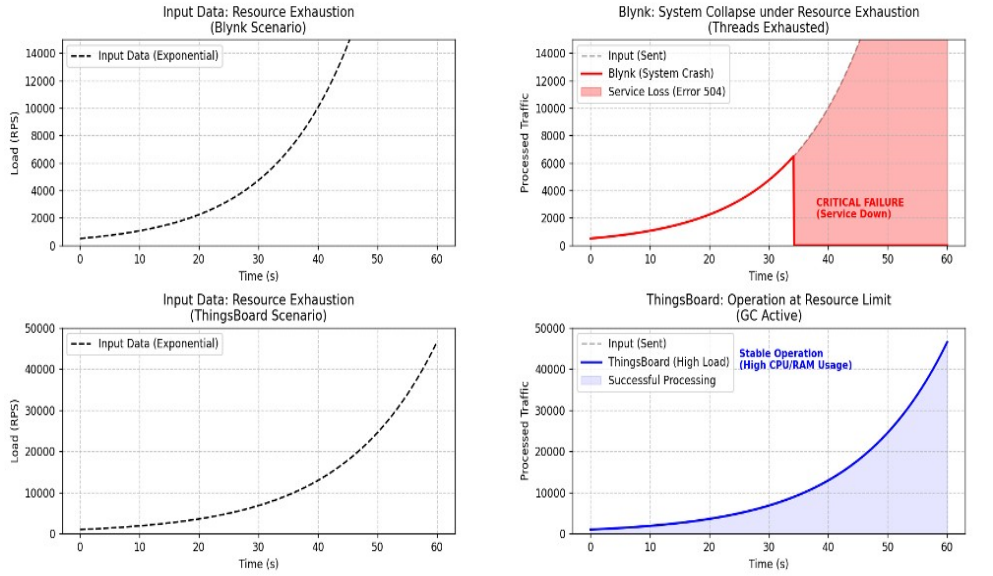


Figure 7: Traffic processing during Resource Exhaustion attack by cloud platforms

The top row of graphs in Fig. 7 (Blynk Cloud) reveals a critical "breaking point" at the 35th second of the test: as soon as the attack intensity reached the threshold of 6,000 requests, the system exhausted its Worker Threads limit and suffered a complete crash. This is reflected by the sharp drop of the processing curve to zero and the red zone indicating a total loss of service.

In contrast, the bottom row of Fig. 7 (ThingsBoard Cloud) demonstrates the platform's resilience, where the processed traffic graph (blue line) remains identical to the incoming load throughout the entire duration. This confirms the system's ability to scale and successfully handle even an extreme flow of 45,000 requests without an emergency shutdown.

The final experiment concerned a combined Pulse and Exhaustion attack model, described by equation (5). The most representative metrics resulting from this experiment are presented in Table 6.

Table 6

Metrics comparison for combined Pulse and Exhaustion attack with ThingsBoard and Blynk

Platform/ Metrics	VU_{max}	P95 latency (ms)	Errors (%)	State after stress test
ThingsBoard	58,500	510.80	0.00%	Active
Blynk	14,800	> 15,000.00	62.15%	Needs to restart

The resilience of platforms to a combined attack is shown in Fig. 8.

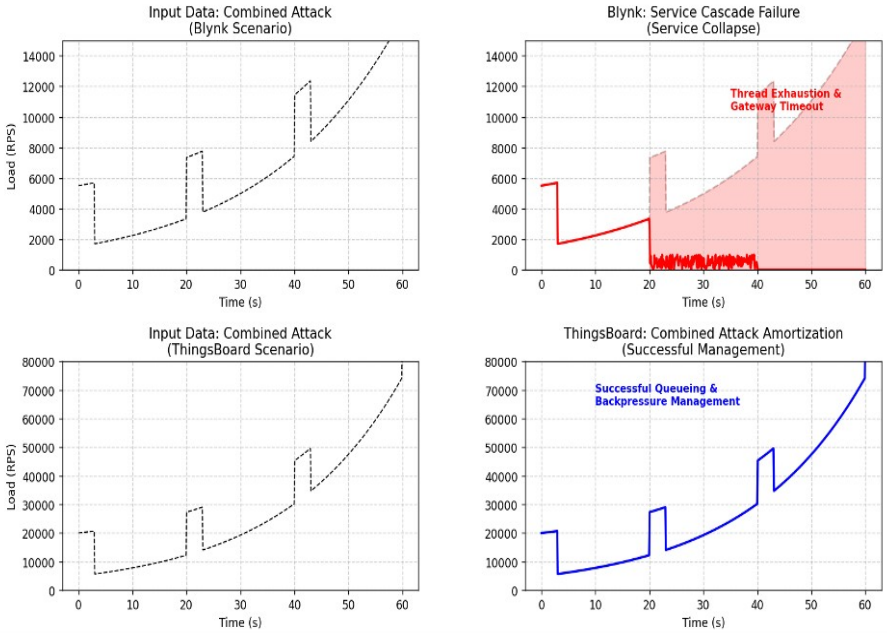


Figure 8: Traffic processing during a combined Pulse and Exhaustion attack by cloud platforms

Figure 8 shows a critical vulnerability in the Blynk part (red graph). The first pulses cause partial data loss, and after the 35th second, when another pulse is superimposed on the exponential growth (Exhaustion attack), the system enters a failure state. The processed traffic drops to zero, which means a complete loss of communication with the object.

In the ThingsBoard part (blue graph), stable operation is revealed even at a load of up to 80,000 RPS (total with pulses). The processed traffic line completely coincides with the input data, which confirms the effectiveness of the queueing mechanisms and non-blocking I/O.

4. Discussion of the results obtained

To enhance the reliability of IoT-based systems, the most effective approach is to use secure, professionally managed platforms. However, it was established that some platforms struggle to handle long-term load increases when a single device generates traffic equivalent to tens or hundreds of sensors. Other platforms (ThingsBoard) proved to be more resilient.

The DDoS Stochastic Noise experiment allows for specific conclusions regarding the performance differences between the platforms. The greater volume and granularity of the ThingsBoard logs are due to the platform successfully processing a data stream four times larger (up to 86,000 RPS) and recording the operation of internal adaptation mechanisms (queue states, Rule Engine load, and memory usage). Conversely, Blynk operated in a denial-of-service mode: upon exceeding the limit, incoming connections were blocked at the gateway, generating only repetitive 429 Too Many Requests errors without providing detailed telemetry of internal processes. This revealed a fundamental difference in the architectural approaches of the studied platforms, which determines their suitability for critical infrastructure.

ThingsBoard Cloud demonstrated high resilience due to an architecture built on asynchronous processing and deep message queues that act as a buffer (amortizer) for incoming traffic. During moments when random noise bursts exceeded the average throughput, the platform did not drop redundant packets but temporarily buffered them in the Rule Engine's internal queues, as evidenced by the recorded increase in RAM utilization. This allowed the system to absorb the peak

load, maintaining zero data loss at the cost of a slight, controlled increase in latency – an acceptable trade-off for ensuring control continuity.

In contrast, Blynk Cloud proved vulnerable to the stochastic nature of the attack due to its reliance on rigid Rate-Limiting mechanisms without sufficient server-side buffering. When the sum of the average load and random noise even briefly exceeded the set limit, a clipping mechanism was triggered, instantly blocking redundant packets and returning an HTTP 429 error. Consequently, the system lost control at critical moments of peak load, demonstrating that static protection methods are ineffective against dynamic attacks on real-world critical infrastructure, where guaranteed data delivery is prioritized over server resource savings.

The Timeout Flood attack experiment also revealed significant operational differences. ThingsBoard Cloud log analysis showed successful processing of requests from 25,000 emulated users via its queuing mechanism. During the surge, the system did not reject requests but buffered them, leading to a queue size increase of up to 5,200 messages and a controlled latency increase to 180 ms, while ensuring 100% data retention.

Conversely, the Blynk Cloud log revealed a critical architectural vulnerability to pulse attacks, where exhausting the 4,000 connection limit led to immediate 503 Service Unavailable errors. Beyond the loss of 12.5% of packets during the attack, the system exhibited inertia: even after the pulse ended, the server required an additional 5 seconds to clear 'hung' connections, during which latency remained critically high (850 ms). These simulation results confirm that IoT system resilience depends critically on architectural connection management mechanisms. For the reliable protection of water supply infrastructure, platforms with an asynchronous architecture capable of amortizing pulse attacks through buffering are required, rather than connection-dropping strategies that result in a loss of situational control.

The exponential Resource Exhaustion attack remains difficult to counter effectively, even within cloud infrastructures. ThingsBoard Cloud logs confirmed that as the load grew exponentially to 45,100 emulated users, the system reached peak resource utilization (CPU 98%, RAM 92%). In response to critical memory saturation (Heap > 85%), the Garbage Collector and Backpressure mechanisms were triggered, increasing latency to 410 ms but avoiding packet loss (0% error rate) and maintaining service operability.

The Blynk Cloud log shows that upon reaching a threshold of 5,800 VUs, total Worker Thread exhaustion occurred. This resulted in a complete halt of request processing (RPS = 0) and mass generation of 504 Gateway Timeout errors.

Testing results for the combined attack model (Pulse + Ramp) confirmed the critical vulnerability of systems with synchronous request processing. The ThingsBoard Cloud platform successfully withstood a load of 58,500 emulated users by effectively combining message queuing during pulses and backpressure during exponential growth. Logs recorded zero data loss, although the system operated at its resource limits (CPU 92%, RAM 88%).

In contrast, Blynk Cloud exhibited the following behavior: the very first pulse against the backdrop of increasing load caused connection pool exhaustion (error 503), and the second pulse led to worker thread exhaustion and a total server shutdown (error 504). As a result, 62.15% of packets were lost, indicating a total failure of monitoring and control functions under these attack conditions. The reason for ThingsBoard Cloud's successful adaptation lies in its use of a non-blocking I/O model and deep message buffering, allowing the platform to accept incoming packets even at CPU loads exceeding 90%, converting peak bursts into temporary queue delays without breaking network connections. Meanwhile, Blynk Cloud’s failure resulted from synchronous request processing, where each attack pulse triggered immediate socket and thread pool exhaustion, leading to a server hang-up state where it ceased responding even to legitimate requests. Ultimately, the primary cause of the discrepancy in results was the resource management method: one allowing for flexible situational control, while the other is based on rigid traffic limits.

The metric values obtained during the experiments (Tables 2–6) lead to the conclusion that for critical nodes (particularly pumping stations), the criterion for communication system operability should not be a general "Online/Offline" status, but rather latency fluctuations and absolute latency values. A sharp increase in latency, even with 0% packet loss, should be interpreted as the onset of an attack or a critical situation for the system.

5. Conclusions

Unlike traditional load testing methods based on linear traffic increases, the proposed models (DDoS with stochastic noise, Timeout Flood, and Resource Exhaustion) allow for the adjustment of the attack profile according to specific mathematical regularities. This ensures higher accuracy in modeling real-world cyber threat scenarios, where attack dynamics adapt to the system's behavior.

The use of open-source tools (k6) combined with low-cost hardware (ESP8266) facilitates the easy replication of the proposed testbed. This represents a significant advantage over existing closed or complex proprietary solutions, enabling critical infrastructure operators to independently conduct regular resilience audits without involving expensive resources.

It has been proven that for water supply facilities, the choice of a cloud platform is not merely an IT issue but directly impacts the physical integrity of the equipment. Latencies (up to 15,000 ms) and high error rates in consumer-grade solutions (Blynk) under attack conditions can lead to a loss of control over pumping units or incorrect automation triggering, threatening emergency situations at the facility.

It was experimentally confirmed that the key factor in resilience against denial-of-service attacks is the presence of asynchronous data processing mechanisms and message queues (as seen in ThingsBoard). This allows the system to remain operational even under extreme loads (up to 58,500 virtual users), which is critical for ensuring uninterrupted water supply during targeted cyberattacks.

The proposed methodology, utilizing the open-source k6 toolkit, can become part of routine operational procedures. This would allow water supply facility operators to independently determine their infrastructure's weak points (maximum number of virtual users/requests), which is crucial before scaling the network or connecting new facilities.

The prospect of further research is to model attacks on extensive water supply systems, including dozens of interconnected pumping stations and reservoirs, to study the 'cascade effect' of cloud service failures.

Declaration on Generative AI

During the preparation of this work, the authors used Gemini Pro for grammar and spelling check. After using these tool, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] S. Adepu et al., "Investigation of Cyber Attacks on a Water Distribution System," 2019. URL: https://arxiv.org/abs/1906.02279;

[2] P. Amanitidis et al., "Intelligent Water Management Through Edge-Enabled IoT, AI, and Big Data Technologies," IoT 7.1 (2026) 5. doi:10.3390/iot7010005.

[3] V. Lakhno et al., "Automation of DDoS attack investigation in industrial control systems using Bayesian networks on Python," in: CEUR Workshop Proceedings, CPITS-II 2024: Workshop on Cybersecurity Providing in Information and Telecommunication Systems II, Kyiv, Ukraine, 2024. URL: https://ceur-ws.org/Vol-3826/short18.pdf.

[4] O. Novikov, G. Vedmedenko, I. Stopochkina, M. Ilin, “Cyber Attacks Cascading Effects Simulation for Ukraine Power Grid,” in: Selected Papers of the XXI International Scientific and Practical Conference “Information Technologies and Security” (ITS 2021), volume 3241 of CEUR Workshop Proceedings, Kyiv, Ukraine, 2021, pp. 23-35. URL: http://ceur-ws.org/Vol-3241/.
[5] A. Abughali, M. Alansari, A. S. Al-Sumaiti, “Defending the Water-Energy Nexus: Strategies for Cyber-Attack Detection and Mitigation,” preprint, 2024. doi:10.2139/ssrn.4743243.
[6] Novikov, O., Shreider, M., Stopochkina, I., M. Ilin, “Cyber attacks simulation for modern energy facilities,” CEUR Workshop Proceedings 3887 (2023) 35–49. URL: https://ceur-ws.org/Vol-3887/.
[7] Kaur, D., M. Sachdeva, “Study of flooding based DDoS attacks and their effect using DETER testbed,” International Journal of Research in Engineering and Technology 2.5 (2013) 879–884. doi:10.15623/ijret.2013.0205151.
[8] O. Novikov, I. Stopochkina, M. Ilin, V. Duduladenko, “Stealthy cyberattacks on control systems using an adaptive soft-constrained optimization method,” Theoretical and Applied Cyber Security (2025) 104-110. doi:10.20535/tacs.2664-29132025.1.333440.
[9] J. Li, B. Moeini, Sh. Nejati, M. Sabetzadeh, M. McCallen, “A Lean Simulation Framework for Stress Testing IoT Cloud Systems,” 2024. URL: https://arxiv.org/abs/2404.11542.
[10] S. A. Okoh, E. N. Onwuka et al., “Performance Analysis of IoT Cloud-based Platforms using Quality of Service Metrics,” J. Wireless and Microwave Technologies 1 (2023) 45-58. doi:10.5815/ijwmt.2023.01.05.
[11] J. B. Minani, F. Sabir, N. Moha, Y.-G. Guéhéneuc, “A Systematic Review of IoT Systems Testing: Objectives, Approaches, Tools, and Challenges,” IEEE Transactions on Software Engineering PP.99 (2024) 1-29. doi:10.1109/TSE.2024.3363611.
[12] D. Nikolopoulos, C. Makropoulos, “Stress-testing water distribution networks for cyber-physical attacks on water quality,” Urban Water Journal 19.3 (2022) 224-237. doi:10.1080/1573062X.2021.1995446.
[13] Blynk, “Low-code IoT Platform,” [Electronic resource]. URL: https://www.blynk.io/.
[14] ThingsBoard, “Open-source IoT Platform,” [Electronic resource]. URL: https://thingsboard.io/.
[15] Baeldung, “How to execute load tests using the k6 Framework,” [Electronic resource]. URL: https://www.baeldung.com/k6-framework-load-testing.
[16] ThingsBoard user guide, “Rule engine. Queues,” [Electronic resource]. URL: https://thingsboard.io/docs/user-guide/rule-engine-2-5/queues/.