A Survey of Implementation-Level Vulnerabilities in LoRa@FIIT and STIoT Infrastructure: Toward an Extensible Security Checklist

Adrián Ondov, Alexander Valach, Pavol Helebrandt, Pavel Čičák, and Katarína Jelemenská
Slovak University of Technology in Bratislava
Bratislava, Slovakia
{alexander.valach, pavol.helebrandt, pavel.cicak, katarina.jelemenska}@stuba.sk

***Abstract*—Security work on LoRa systems is frequently framed through the wireless channel. That view is incomplete for LoRa@FIIT deployments, where a received frame enters a longer implementation chain: LoAP access point software, the STIoT access-point-to-network-server protocol, the LoNES network server, persistent database state, visualization tools, and application-facing interfaces. Vulnerabilities in these layers can invalidate otherwise sound assumptions about the link-layer protocol. This paper surveys the implementation-level attack surface of LoRa@FIIT and STIoT infrastructure, with emphasis on access point registration, STIoT message validation, end-device registration integrity, key and sequence number handling, code injection through application data, database access, remote management, and audit evidence. VulneLoRa is presented as an extensible checklist-driven assessment tool for secure LoRa@FIIT infrastructure. The checklist connects protocol-specific tests, such as SETR flooding, registration replay, and key synchronization checks, with conventional web and infrastructure security controls, such as parameterized database access, credential protection, restricted SSH access, database isolation, and redacted logging. The resulting checklist validates the full path from end-device registration to application output, not only the radio link.**

***Index Terms*—LoRa@FIIT, STIoT, LoRa, IoT security, access point, network server, SQL injection, vulnerability assessment, security checklist, VulneLoRa**

I. Introduction

LoRa-based systems are usually discussed in terms of long-range, low-power wireless communication. This has produced valuable research on jamming, eavesdropping, replay, beacon spoofing, key establishment, and frame-counter behavior [1]–[7]. However, the security of an operational LoRa deployment does not end at the air interface. After reception, a message is handled by access point software, transformed into an IP-based message, processed by a network server, stored in a database, and finally used by applications or visualization tools. Each of these steps creates implementation risks closer to web, API, database, and Linux-server security than to radio communication alone.

This observation is especially important for LoRa@FIIT, a link-layer protocol designed for local LoRa networks where the complexity and overhead of LoRaWAN roaming are not required [8], [9]. In the LoRa@FIIT stack, LoAP access points bridge LoRa communication into STIoT, a TCP/TLS-based protocol intended for communication between access points and the LoNES network server. The network server controls transmission parameters, supports end-device registration, synchronizes session keys and sequence counters, stores uplink and downlink state, and provides data for higher-level applications [8], [10], [11]. A weakness in this implementation path can therefore become a practical compromise even when the LoRa@FIIT packet format behaves as expected.

VulneLoRa implements offensive checks for LoRa@FIIT infrastructure, including replay, eavesdropping, SETR flooding, pre-shared-key extraction, direct and end-device SQL injection, ARP spoofing, and SSH exploitation [12]. Its command-line workflow supports interactive and simulation modes, attack selection, attack-specific configuration, saving and loading of parameters, output collection, and a template for adding new attack scripts. The tool is therefore suitable for a checklist interpretation: each module verifies a concrete implementation control and produces evidence that can be reviewed by a developer or operator.

VulneLoRa serves as the practical basis for a secure LoRa@FIIT and STIoT implementation checklist [12]. The checklist covers access point registration, STIoT setup messages, end-device registration, key synchronization, application-data handling, database access, administrative services, logging, and recovery behavior. New modules can be added as LoAP, LoNES, LoBook, and application integrations evolve, while the checklist categories remain stable.

This paper focuses on three primary aspects. First, it surveys the LoRa@FIIT and STIoT implementation attack surface from an infrastructure perspective, rather than from a radio-only perspective. Second, it organizes known and implemented attacks into trust-boundary categories that are meaningful for secure deployment. Third, it proposes a VulneLoRa-oriented checklist that can be used to evaluate future implementations of LoRa@FIIT infrastructure.

II. Scope and Survey Method

The scope of this survey is implementation-level security of LoRa@FIIT and STIoT deployments. Radio-layer attacks are treated only as context when they interact with access point or server behavior. For example, eavesdropping is discussed not only as signal observation, but also as a test of whether

registration metadata, transmission parameters, sequence counters, or key-handling assumptions become visible through logs, access point output, or backend processing. Denial of service is discussed primarily as SETR flooding, registration stress, key-synchronization abuse, database exhaustion, or service disruption, not as radio jamming.

The survey combines three sources. The first source is the protocol and infrastructure description of LoRa@FIIT and STIoT, including the original protocol paper and the open-source ecosystem around LoAP, LoNES, LoMAB, LoBook, and VulneLoRa [8], [10]–[14]. The second source is the current VulneLoRa module set, which provides concrete checks for replay, eavesdropping, SETR flooding, key extraction, SQL injection, SSH exposure, ARP spoofing, and direct database access [12]. The third source is selected LoRa, IoT, web, and API security literature used to classify implementation failures that appear after LoRa frames enter the IP and application stack [2], [15]–[24].

A vulnerability is considered in scope if it can be triggered or observed through the LoRa@FIIT or STIoT infrastructure without requiring control of the physical LoRa modulation. In practical terms, this includes unauthorized access point registration, weak access point identity checks, insecure TLS deployment, exposed SSH services, leakage of pre-shared or session keys, code injection through decoded application data, direct database exposure, backend flooding, insufficient log redaction, and missing evidence in monitoring. The scope intentionally remains narrow: the paper does not introduce a new radio attack, a new cryptographic protocol, or a general LoRaWAN backend taxonomy.

III. LoRa@FIIT and STIoT Infrastructure

LoRa@FIIT was proposed as a link-layer protocol for LoRa networks to reduce overhead in deployments where roaming is not required [8]. It defines data, hello, emergency, and register messages. Register messages support end-device activation; data, hello, and emergency messages carry ordinary traffic or reliability-related control behavior. The protocol uses XXTEA for encrypted message parts, which reduces padding overhead for short messages compared with AES-128 block alignment in LoRaWAN. The original evaluation reported that a one-byte payload requires 12 B in LoRa@FIIT compared with 29 B in LoRaWAN, which is the main efficiency motivation for the protocol [8].

STIoT, or Secured TCP for IoT, is the protocol used between access points and the network server. It is JSON-based, TCP-oriented, and intended to run over TLS. It organizes communication into access point setup, device registration, key and counter synchronization, and device transmission. SETR and SETA start access point communication and configure radio parameters. REGR and REGA support end-device registration. KEYR, KEYA, and KEYS synchronize session keys and sequence counters. RXL and TXL carry uplink data and downlink responses [8]. Recent LoRaWAN standardization is used here only as comparative context.

TABLE I. Security-relevant STIoT message groups in LoRa@FIIT infrastructure

Group	Messages	Security role
AP setup	SETR, SETA	Starts AP-to-NS communication. Carries AP identity, STIoT version, supported radio parameters, and NS-selected reception configuration.
Device registration	REGR, REGA	Carries device identifier, reception metadata, first configuration, application data, network data, and in LoRa@FIIT the shared key material used during registration.
Key/counter sync	KEYR, KEYA, KEYS	Synchronizes session keys and sequence counters through server-side state keyed by device identifier.
Transmission	RXL, TXL, ERROR	Carries decrypted application data, reception metadata, network configuration, downlink responses, and error conditions.

TR007 provides implementation guidance for producing well-behaved and interoperable end devices, while the 2026 end-device/network-server interoperability test plan shows how onboarding, parameter handling, and server-side behavior can be expressed as repeatable test cases [25], [26]. TS014 defines a standard interface for downloading a Device Profile from a Device Profile Server, while TS018 standardizes QR-code-based device identification for automated onboarding [27], [28]. Table I lists the message groups as checklist targets.

Fig. 1 shows the evaluation network used for VulneLoRa testing. The relevant path is the transition from a LoRa packet into LoAP, LoNES, and the database, where protocol fields become host, service, and database state.

From a security perspective, the infrastructure can be divided into five zones. The first zone is the end device and LoRa interface. The second zone is LoAP, which receives frames, parses identifiers and message types, decrypts or prepares message parts where needed, and converts between LoRa@FIIT and STIoT. The third zone is the STIoT connection, where JSON messages are exchanged over TCP/TLS. The fourth zone is LoNES and its database, where access point records, device records, keys, sequence counters, uplink messages, downlink messages, and radio parameters become persistent state. The fifth zone is the operator and application layer, including LoBook, administrative access, and exported data. VulneLoRa is placed across these zones as an assessment tool: one module may act at the AP host, another at the NS port, another through the simulated end device, and another through database or SSH access.

IV. Threat Taxonomy for Secure Implementation

This section describes selected threats identified from the LoRa@FIIT and STIoT design, the LoAP and LoNES implementation model, and the checks implemented in VulneLoRa [10]–[12], [29]. The taxonomy is implementation-centered and follows the components that a developer or operator must validate when deploying LoRa@FIIT infrastructure.

A. Access Point Registration and Identity

The access point is the first IP-connected actor in the LoRa@FIIT infrastructure. In LoAP, the PacketConverter

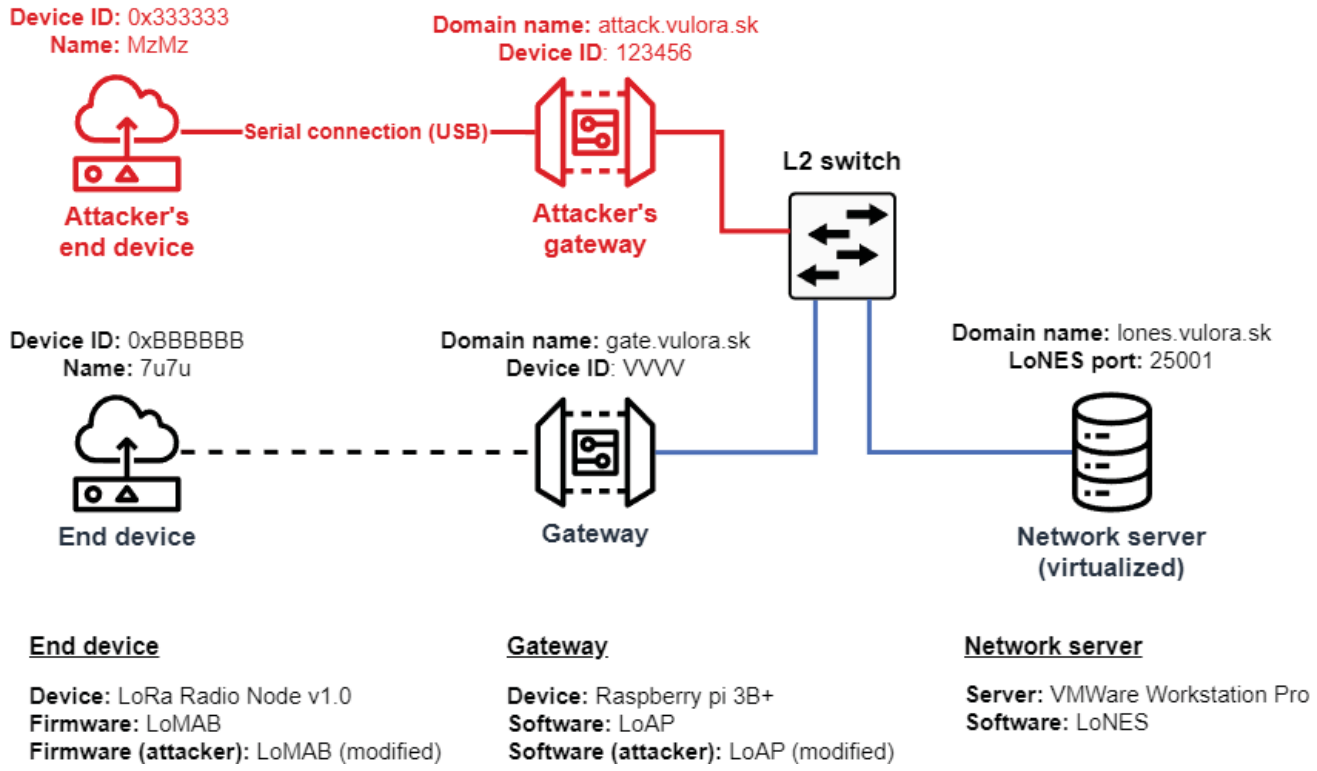


Fig. 1. LoRa@FIIT evaluation network used for VulneLoRa testing. The survey focuses on the implementation path from the LoRa access point to the LoNES network server and database

component connects to the network server and uses STIoT to translate between radio-side packets and server-side JSON messages [11]. The SETR message is therefore security-critical. It introduces an access point to LoNES, declares the STIoT version and LoRa@FIIT standard version, and advertises hardware and radio capabilities such as supported frequencies, spreading factors, coding rates, bandwidths, and maximum power [8]. SETA then returns the server-selected configuration.

The checklist question is precise: can only an authorized LoAP instance complete SETR/SETA setup and remain connected? A weak implementation may accept a client-provided AP identifier as sufficient proof of identity, allocate state for repeated setup attempts, or apply server-side configuration to an AP without clear revocation semantics. The assessment should therefore test registration of an unknown AP, replay of a previously valid SETR, reuse of a removed AP identifier, inconsistent capability declarations, and repeated setup attempts. The required evidence is not only a pass/fail result, but also LoNES logs, AP records in the database, SETA response behavior, and monitoring events for rejected or excessive setup traffic.

B. STIoT Message Validation

The STIoT paper specifies TLS-based communication and server authentication for the AP-to-NS connection [8]. That design reduces exposure only if the implementation validates the peer correctly and then treats every received JSON message as untrusted input. STIoT messages are short, but they are semantically powerful: SETR can influence access point setup, REGR/REGA controls end-device registration, KEYR/KEYA/KEYS exchanges state used for encrypted communication, and RXL/TXL carries decoded data and radio metadata.

A checklist-driven tool must validate both syntax and state. The syntactic checks are message type, required keys, value type, length, range, Base64 encoding, and accepted protocol version. The state checks are stricter: KEYR should not succeed for an unrelated device, KEYS should not overwrite state outside the correct registration flow, RXL should not be accepted for an unknown or inactive device, and TXL generation should not occur from inconsistent server-side state. The STIoT parser acts as an internal API between LoAP and LoNES and must be assessed with API-style negative tests.

C. Registration and Device-State Integrity

End-device registration is the most sensitive LoRa@FIIT workflow because it creates persistent state and prepares later communication. In the STIoT specification, REGR carries the device identifier and reception metadata such as RSSI, SNR, bandwidth, coding rate, spreading factor, time, and for LoRa@FIIT also duty-cycle information. REGA returns the device identifier, transmission power, network configuration, optional application data, and the shared key material required

by the access point during registration. KEYR, KEYA, and KEYS then synchronize the session key and sequence counter between access point and network server [8].

The VulneLoRa replay scenario assesses registration as a state-integrity problem, not only as packet repetition. The workflow listens for registration communication, extracts parameters such as frequency and spreading factor, confirms the target device identity, and then attempts registration through a simulated access point and simulated end device [12]. The resulting checklist item asks whether LoNES and LoAP bind device identity, registration freshness, radio metadata, session key creation, and sequence number initialization strongly enough that a captured or reconstructed registration path cannot introduce a false device or reset relevant state.

D. Secret Management and Host Exposure

Secrets in LoRa@FIIT infrastructure are not abstract cryptographic values. They appear in concrete places: LoNES environment variables, Java keystores, database records, LoAP configuration, packet-converter logs, temporary files, SSH-accessible directories, and notebooks used for visualization or testing. The LoNES deployment model explicitly uses environment variables for the database, pgAdmin, keystore password, server port, log file, and `LONES_PRESHARED_KEY` [10]. The VulneLoRa PSK extraction module is therefore a checklist test for deployment hygiene: it checks whether ordinary host access can disclose pre-shared or validation keys used by the infrastructure.

This category also covers sequence counters. In LoRa@FIIT, session keys and sequence counters are synchronized through KEYR, KEYA, and KEYS. If these values are leaked, reset, or overwritten through database access or host compromise, replay protection and registration integrity are weakened. A secure implementation must therefore define where keys are allowed to exist, who may read them, how they are rotated, and how logs are redacted. The test evidence should include file-permission results, secret-search output, database privilege separation, log inspection, and confirmation that keys do not appear in ordinary diagnostic output.

E. Payload Processing and Code Injection

Application data in STIoT is carried as a Base64-encoded string, but encoding is not a security boundary. After LoAP and LoNES decode or store a value, it may reach a database insert, a LoBook notebook, a CSV export, an API response, or a custom application. The VulneLoRa end-device SQL injection scenario treats a LoRa device as an input channel for backend code injection: the malicious string starts as application data, passes through the LoRa@FIIT/STIoT path, and is evaluated by the backend or by a later processing component if that component constructs SQL, commands, templates, or output unsafely [12].

This claim should be kept specific. The point is not that every LoNES database operation is necessarily implemented with unsafe SQL. The current checklist meaning is that every component consuming decoded application data must be tested as if it were receiving hostile web input. The minimum evidence is whether the payload is stored as data, whether parameterized database access is used, whether rejected payloads are logged without stack traces or query fragments, and whether visualization or export code encodes the value before display. This gives the paper a sharp application-security message without turning it into a generic OWASP list [22], [23].

F. The Availability of Backend Services

Availability attacks against LoRa@FIIT infrastructure do not require radio jamming. SETR flooding directly exercises the AP setup boundary. Registration loops exercise REGR/REGA handling. KEYR and KEYS bursts exercise state synchronization. High-rate RXL messages exercise parsing, database writes, and downstream application updates. The implemented VulneLoRa SETR flooding module is therefore not merely a denial-of-service demonstration; it is a rate-limit and early-rejection test for LoNES setup handling. Recent LoRaWAN work on access-point-side DoS supports the terminology used for AP-side availability stress, while energy-depletion surveys motivate recording rate, retry, and recovery effects rather than treating denial of service as a binary result [5], [30].

A useful assessment records the configuration and the measured effect: target IP address, target port, AP identifier, message limit, inter-message delay, number of accepted SETR messages, number of rejected messages, server response time, database state growth, CPU and memory behavior, and recovery time. These values make the test useful for engineering rather than only for attack illustration. The expected secure behavior is bounded work before AP identity is validated, bounded connection and queue use, deterministic rejection, and observable alerts for excessive setup traffic.

G. Operator Interfaces and Auditing

LoRa@FIIT infrastructure becomes operational through SSH administration, Docker deployment, PostgreSQL or pgAdmin access, LoBook visualization, logs, and application-facing interfaces. These elements are outside the LoRa@FIIT packet specification, but they determine whether a laboratory or production deployment is actually secure. VulneLoRa already reflects this by including SSH exploitation, direct SQL injection, PSK extraction, eavesdropping, and ARP spoofing modules. These modules do not extend the radio protocol; they test the host, network, and operator paths that surround LoAP and LoNES.

The checklist questions should stay concrete. Is SSH exposed on LoAP or LoNES hosts, and does it allow weak-password access? Is PostgreSQL reachable from unnecessary network segments? Does pgAdmin use deployment-specific credentials? Are LoNES and LoAP logs detailed enough to reconstruct SETR floods, repeated registrations, KEYR/KEYS anomalies, and code-injection attempts? Are secrets redacted from those logs? Can LoBook or exported data reveal raw



Fig. 2. VulneLoRa attack selection in interactive mode. The LoRa@FIIT module list represents the first extensible checklist surface

keys, device identifiers, or malicious payloads without encoding? These questions are narrow enough for a FRUCT survey paper and directly traceable to the current tool and infrastructure.

V. VulneLoRa as a Checklist Instrument

VulneLoRa is used here as a first version of a checklist-driven assessment tool for secure LoRa@FIIT infrastructure. The current implementation provides command-line operation, interactive mode, simulation mode, attack selection, attack-specific configuration, saving and loading of parameters, output collection, LoBook-based visualization for selected scenarios, and a template for adding new attack scripts [12]. Each checklist item can correspond to a module with inputs, prerequisites, expected secure behavior, observed behavior, and evidence.

Table II separates stable checklist areas from concrete tests and pairs each area with a remediation technique. For AP registration, the control may later be implemented with mutual TLS, signed AP credentials, or VPN identity. For code injection, the current test focuses on SQL injection, while the same checklist area can cover command injection, unsafe template rendering, or unsafe export handling when such components exist in a deployment.

The VulneLoRa interactive menu in Fig. 2 is the first checklist surface. It lists LoRa@FIIT-focused modules together with a limited LoRaWAN module; the operational assessment target is LoRa@FIIT/STIoT infrastructure.

Fig. 3 shows attack configuration for SETR flooding. The same interaction model is useful for checklist execution because parameters, target scope, and expected evidence can be stored together with the test result.

VI. Reinterpreting Existing Attack Workflows

Figs. 4 and 5 map implemented VulneLoRa workflows to checklist controls. The diagrams are read as assessment workflows: a module executes a controlled test, records the effect on LoAP, LoNES, database state, or logs, and identifies the remediation control that must be present:

1) **Replay attacks**. The replay workflow is a registration-integrity test. It captures registration-related data, extracts transmission parameters, confirms the frequency

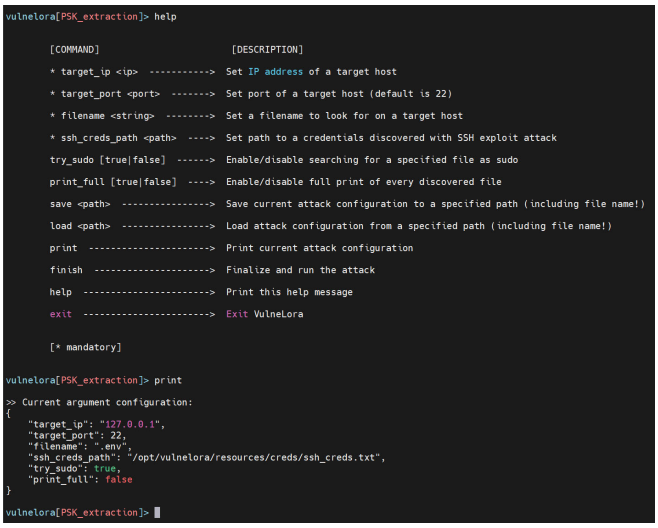


Fig. 3. Example of VulneLoRa parameter inspection during SETR flooding configuration

used by the target device, and then attempts to register a false device through a simulated AP and end device [12]. A successful test indicates that the implementation must strengthen binding between device identity, registration freshness, AP identity, and server-side state transitions.

2) **SETR flooding**. The SETR flood targets the AP setup boundary. It verifies whether LoNES performs expensive work before authenticating or validating the AP, whether repeated setup requests allocate unbounded state, and whether monitoring detects abnormal setup behavior. The assessment is configured through concrete values such as target IP address, target port, AP identifier, message limit, and delay. The mitigation is not radio-layer protection, but stronger access point registration, early rejection, bounded connection handling, and rate-limit evidence.
3) **Pre-shared key extraction**. The PSK extraction workflow verifies host and operational controls. If validation keys can be extracted through SSH compromise, filesystem exposure, environment variables, or over-permissive service accounts, the cryptographic design of LoRa@FIIT is no longer the limiting factor. Secure infrastructure must minimize where secrets exist and must produce evidence that secrets do not appear in logs, notebooks, database exports, or ordinary user-readable files.
4) **End-device SQL injection**. The SQL injection workflow treats a LoRa device as a backend input vector. The decoded data is untrusted input. The module verifies whether the decoder, database layer, visualization tool, and export path keep the malicious value as data or accidentally interpret it as SQL, a command, a template fragment, or an executable expression. The remediation is parameterized data access, constrained decoders, output encoding, and safe error handling along the full

TABLE II. EXTENSIBLE CHECKLIST FOR SECURE LORA@FIIT AND STIOT INFRASTRUCTURE

Checklist area	Primary target component	Security question	Evidence	Remediation technique
Access point registration	LoAP / LoNES	Can only authorized APs complete SETR/SETA setup and remain connected?	Accepted and rejected SETR events, AP database state, SETA behavior	Bind AP identity to a controlled credential Reject unknown AP identifiers Support AP revocation Rate-limit SETR attempts
STIoT message validation	LoNES	Are invalid SETR, REGR, KEYR, KEYS, and RXL messages rejected in the correct protocol state?	Parser result, error message, state-machine log, crash absence	Enforce strict JSON schema Validate message order Bound all fields Reject out-of-state messages before DB updates
End-device registration integrity	LoAP / LoNES / DB	Can an attacker replay, reconstruct, or remap registration state?	Device registry diff, selected radio parameters, sequence number diff	Bind device identity to registration freshness Verify session-key derivation Preserve AP context Initialize sequence counters safely
Secrets and sequence counters	LoAP / LoNES / DB	Are PSKs, session keys, and counters protected from host, log, and database exposure?	Secret-search report, log-redaction result, DB privilege result	Use restricted configuration or secret stores Remove keys from debug output Separate database privileges Define key rotation
Code injection	Decoder / DB / LoBook	Can end-device application data influence SQL, commands, templates, logs, or exports?	Stored-row diff, rejected payloads, sanitized output, error handling	Use parameterized SQL Constrain decoders Limit input length Encode output and errors safely
Service availability	LoNES / DB / queues	Do setup, registration, key-sync, or uplink floods degrade service?	Rate, latency, CPU/memory, accepted messages, recovery time	Authenticate and validate early Apply per-source and per-AP limits Bound queues and database pools Alert on abnormal rates
Remote management attack surface	LoAP and LoNES hosts	Are SSH, database, debug, and dashboard services minimized and strongly authenticated?	Service inventory, authentication result, failed-login logs	Disable password-based SSH where possible Restrict management ports Remove default credentials Segment AP and NS hosts
Audit evidence	Logs / LoBook / reports	Are attacks detectable without disclosing application data or keys?	Redacted logs, LoBook output, VulneLoRa report artifact	Produce structured redacted logs Correlate SETR/REGR/KEYR/KEYS/RXL events Retain assessment artifacts Export checklist reports

processing path.

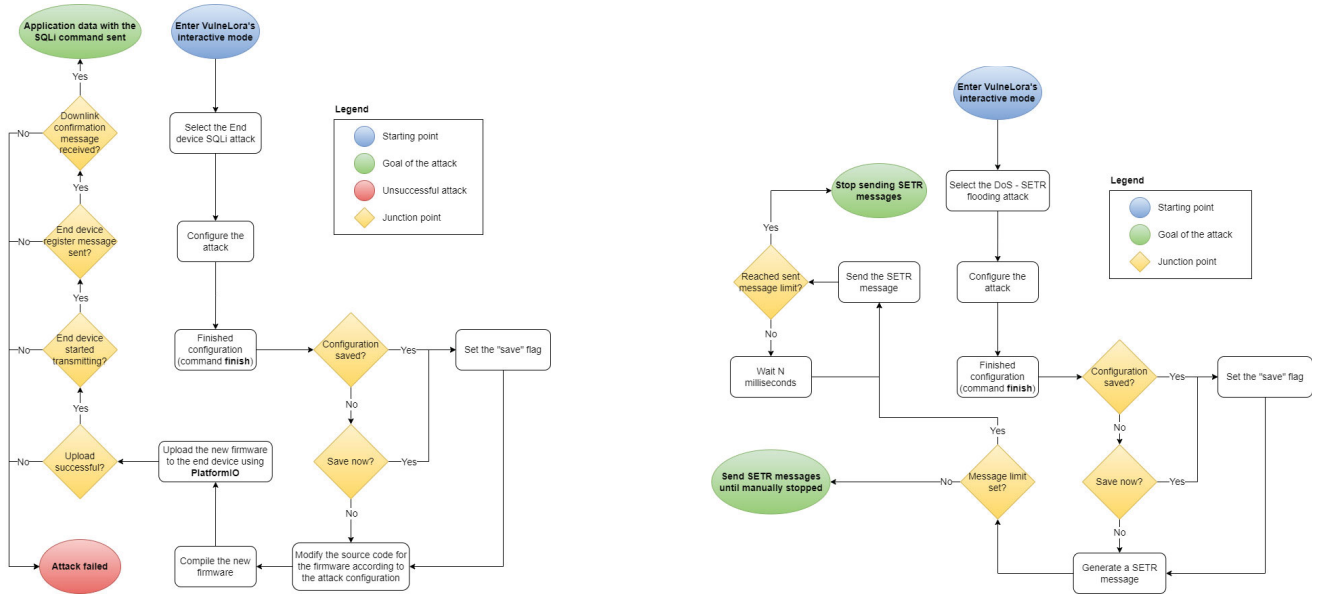
5) **Eavesdropping**. Eavesdropping is often classified as a wireless attack, but the implemented workflow also tests what useful metadata can be obtained from infrastructure communication and logs. In LoRa@FIIT/STIoT, message names, device identifiers, radio parameters, ACK requirements, sequence counters, and key-synchronization events may be enough to guide later replay, false registration, or service-disruption tests. The checklist item is therefore not only about confidentiality. It is also about binding and auditability across the AP-to-NS boundary.

VII. MITIGATION PATTERNS

A secure LoRa@FIIT implementation should be hardened by the same boundaries used in the checklist. Access point registration should reject unknown AP identifiers, bind AP identity to controlled credentials, support revocation, and record failed setup attempts. Where feasible, TLS should be complemented by mutual authentication or an equivalent deployment-level identity. The STIoT parser should validate message type, mandatory fields, field ranges, Base64 fields, protocol version, and the current protocol state before updating LoNES state.

End-device registration should bind device identifier, radio parameters, session key creation, and sequence number initialization. KEYR, KEYA, and KEYS handling should be restricted to the AP and device context for which the operation is valid. Database accounts should separate ordinary message storage from key and sequence number management. Direct database administration should not be reachable from untrusted network segments, and secrets such as `LONES_PRESHARED_KEY` should not be available to ordinary users or appear in debug logs.

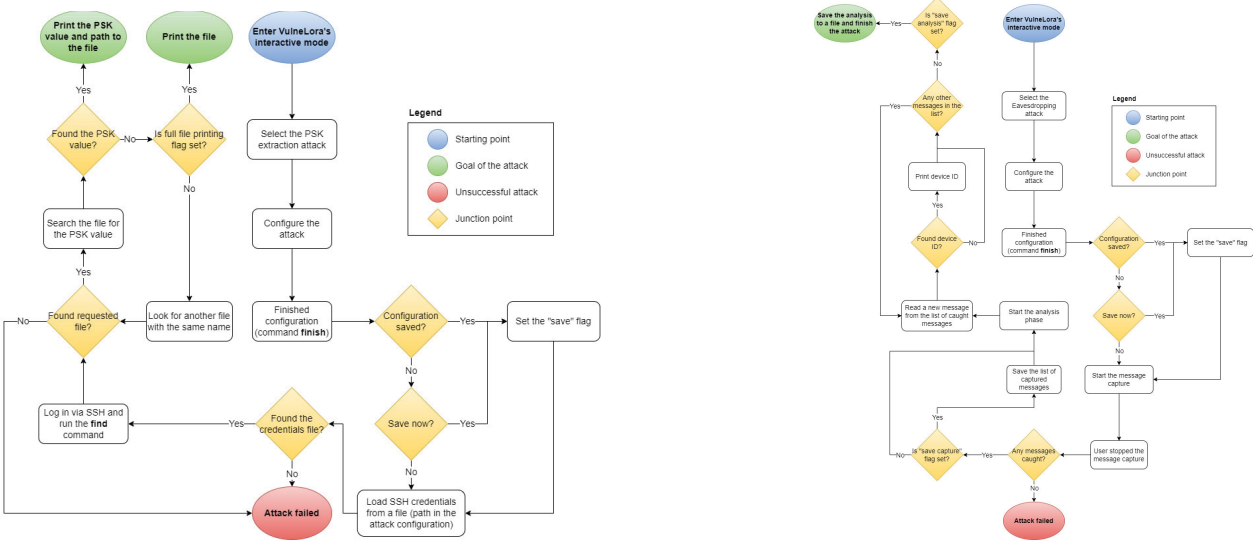
Application-data processing should treat decoded sensor data as hostile input. The concrete controls are parameterized database access, strict schemas for expected payload formats, length and character constraints where protocol semantics allow them, output encoding in LoBook or dashboards, and safe handling of rejected values, consistent with recent IoT



(a) End-device SQL injection workflow

(b) SETR flooding workflow

Fig. 4. VulneLoRa workflows that correspond to code-injection and service-availability checklist areas



(a) Validation-key extraction workflow

(b) Eavesdropping workflow

Fig. 5. VulneLoRa workflows that correspond to secret-management and audit-evidence checklist areas

application-layer security surveys [20]. Availability controls should be measured rather than assumed: SETR floods and registration stress tests should demonstrate rate limits, bounded state allocation, recoverability, and useful monitoring output.

VIII. Discussion and Research Directions

The proposed checklist exposes several research gaps. First, LoRa@FIIT and STIoT security evaluation needs datasets that also include STIoT messages, network server and access point logs, key-synchronization events, and rejected application-data inputs. Without this evidence, it is difficult to compare existing components, such as LoAP, LoNES, and future implementations; recent interoperability and IoT verification work supports this test-evidence orientation [21], [26].

Second, VulneLoRa should evolve from attack execution toward checklist evidence collection. A module is useful for further research only if it records target configuration, input parameters, expected secure behavior, observed behavior, logs, database changes, and mitigation status. This is particularly important for SETR flooding, replay, key extraction, and SQL injection because these tests can otherwise be described only

as demonstrations instead of reproducible assessment results.

Finally, the checklist should remain extensible but bounded. The relevant categories are AP registration, STIoT validation, end-device registration, key and sequence number handling, code injection through application data, database access, service availability, secure remote management, and audit evidence. A fixed list of attacks will eventually become outdated, but these categories match the actual LoRa@FIIT infrastructure and the current VulneLoRa implementation closely enough to guide future work.

IX. Conclusion

This paper reframed LoRa@FIIT security as secure infrastructure implementation, not only as secure radio communication. The STIoT AP-to-NS protocol, the LoNES network server, LoAP access points, database state, and application-data path form a conventional IP and web-facing attack surface. Vulnerabilities such as unauthorized access point registration, SETR flooding, secret extraction, SQL injection through application data, weak SSH exposure, direct database access, or unredacted logs can compromise the system even when the link-layer design is efficient and cryptographically motivated.

We introduced VulneLoRa as a checklist-driven assessment instrument for LoRa@FIIT and STIoT infrastructure. Its current modules already demonstrate several checklist areas; its architecture allows additional modules for AP registration, STIoT schema and state validation, TLS verification, code-injection testing, service-exposure testing, log-redaction checks, and report generation. The main conclusion is practical: a secure LoRa@FIIT deployment requires repeatable validation from end-device registration through access point, network server, database, and stored application payload.

Acknowledgment

Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I05-03-V02-00012. This project has been supported by APVV-23-0137 project “Legal and technical aspects of cybersecurity situational awareness”.

References

[1] E. Aras, G. S. Ramachandran, P. Lawrence, and D. Hughes, “Exploring the security vulnerabilities of LoRa,” in *2017 3rd IEEE International Conference on Cybernetics*, 2017, pp. 1–6.

[2] X. Yang, E. Karampatzakis, C. Doerr, and F. A. Kuipers, “Security vulnerabilities in LoRaWAN,” in *Proceedings of the 3rd ACM/IEEE International Conference on Internet-of-Things Design and Implementation*, ser. IoTDI ’18, 2018, pp. 129–140.

[3] F. Hessel, L. Almon, and F. Álvarez, “ChirpOTLE: A framework for practical LoRaWAN security evaluation,” 2020. [Online]. Available: https://arxiv.org/abs/2005.11555

[4] P. de Moraes and A. F. da Conceição, “A systematic review of security in the LoRaWAN network protocol,” 2021. [Online]. Available: https://arxiv.org/abs/2105.00384

[5] A. Proto, C. C. Miers, and T. C. M. B. Carvalho, “Classification and Characterization of LoRaWAN Energy Depletion Attacks: A Review,” *IEEE Sensors Journal*, vol. 25, no. 2, pp. 2141–2156, 2025.

[6] D. Papatsaroucha, N. Astyrakakis, E. Pallis, P. I. Radoglou-Grammatikis, P. G. Sarigiannidis, and E. K. Markakis, “A Cloud-Based Key Rolling Technique for Alleviating Join Procedure Replay Attacks in LoRaWAN-Based Wireless Sensor Networks,” in *2024 IEEE International Conference on Big Data (BigData)*, Washington, DC, USA, 2024, pp. 2811–2820.

[7] E. Toe, F. Jaafar, and L. C. A. Ferrier, “Securing LoRaWAN in the AIoT Era: A Systematic Mapping Study and a MITRE-Based Threat Matrix,” *IEEE Internet of Things Journal*, vol. 13, no. 3, pp. 3827–3849, 2026.

[8] O. Perešıni and T. Krajčovič, “More efficient IoT communication through LoRa network with LoRa@FIIT and STIoT protocols,” in *2017 IEEE 11th International Conference on Application of Information and Communication Technologies*, 2017, pp. 1–6.

[9] A. Valach, “Optimization of communication of IoT devices,” Ph.D. dissertation, Slovak University of Technology in Bratislava, Faculty of Informatics and Information Technologies, 2023.

[10] LoRaAlex, “LoNES,” GitHub repository, 2023, accessed 2026-09-05. [Online]. Available: https://github.com/loraalex/LoNES

[11] LoRaAlex, “LoAP,” GitHub repository, 2023, accessed 2026-09-05. [Online]. Available: https://github.com/loraalex/LoAP

[12] Adrian Ondov, “VulneLoRa: Vulnerability assessment tool for LoRa technology and LoRa@FIIT protocol,” 2026, accessed 2026-09-04. [Online]. Available: https://github.com/loraalex/VulneLoRa

[13] Valach, Alexander, “LoMAB,” GitHub repository, 2023, accessed 2026-09-05. [Online]. Available: https://github.com/alexandervalach/LoMAB

[14] Valach, Alexander, “LoBook,” GitHub repository, 2023, accessed 2026-09-05. [Online]. Available: https://github.com/loraalex/LoBook

[15] L. S. Laufenberg, “Impersonating LoRaWAN gateways using Semtech packet forwarder,” 2019. [Online]. Available: https://arxiv.org/abs/1904.10728

[16] O. Magnusson, K. Andersson, and M. Gidlund, “A survey on attacks and defences on LoRaWAN gateways,” in *Internet of Things, Smart Spaces, and Next Generation Networks and Systems*. Springer, 2021, chapter publication record available from Chalmers Research.

[17] F. Kuntke, E. Romanenko, S. Linsner, J. Steinbrink, and C. Reuter, “LoRaWAN security issues and mitigation options by the example of agricultural IoT scenarios,” *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 5, p. e4452, 2022.

[18] A.-C. Honfoga, M. Dossou, and V. Moeyaert, “IoT using LoRaWAN: A security analysis,” in *CEUR Workshop Proceedings*, vol. 3789, 2024, pp. 1–5. [Online]. Available: https://ceur-ws.org/Vol-3789/Paper1.pdf

[19] J. C. Osuorah, S. T. Ahmed, A. Annamalai, and M. F. Chouikha, “Vulnerabilities and Layered Security Threats in LoRaWAN Systems: A Comprehensive Survey on Physical, MAC, and Application Layer Weaknesses,” *IEEE Access*, vol. 13, pp. 217 939–217 953, 2025.

[20] N. S. Kumari, H. S. Vimala, C. N. Pruthvi, and J. Shreyas, “Holistic Survey on Security in IoT Application Layer: Attacks, Protocols, and Applications,” *IEEE Access*, vol. 12, pp. 186 957–187 014, 2024.

[21] L. Abuserrieh and M. H. Alalfi, “A Survey on Verification of Security and Safety in IoT Systems,” *IEEE Access*, vol. 12, pp. 138 627–138 645, 2024.

[22] OWASP Foundation, “OWASP Top 10:2025,” 2025, accessed 2026-09-05. [Online]. Available: https://owasp.org/Top10/2025

[23] OWASP Foundation, “OWASP API Security Top 10:2023,” 2023, accessed 2026-06-07. [Online]. Available: https://owasp.org/API-Security/editions/2023/en/0x11-t10/

[24] M. Fagan, K. Megas, K. Scarfone, and M. Smith, “IoT device cybersecurity guidance for the federal government: Establishing IoT device cybersecurity requirements,” National Institute of Standards and Technology, Tech. Rep. NIST SP 800-213, 2021.

[25] LoRa Alliance Technical Committee, “TR007 Developing LoRaWAN Devices v1.1.0,” LoRa Alliance, Tech. Rep., Sep. 2025, accessed 2026-09-06. [Online]. Available: https://resources.lora-alliance.org/document/tr007-developing-lorawan-devices-v1-1-0

[26] LoRa Alliance Technical Committee, “LoRaWAN End-Device and Network Server Interoperability Test Plan v1.1.1,” LoRa Alliance, Tech. Rep., May 2026, accessed: 2026-09-06. [Online]. Available: https://resources.lora-alliance.org/document/21-april-2026-lorawan-end-device-and-network-server-interoperability-test-plan-v1-1-1

[27] LoRa Alliance Technical Committee, “TS014-1.0.0 End Device Capabilities API,” LoRa Alliance, Tech. Rep., Jul. 2026, accessed: 2026-09-06. [Online]. Available: https://resources.lora-alliance.org/home/ts014-1-0-0-end-device-capabilities-api

[28] LoRa Alliance Technical Committee, "TS018-1.0.0 LoRaWAN Device Identification QR Codes for Automated Onboarding," LoRa Alliance, Tech. Rep., 2026, accessed: 2026-09-06. [Online]. Available: https://resources.lora-alliance.org/technical-specifications

[29] A. Valach, "LoRa@FIIT Arduino Library," GitHub repository, 2024, accessed 2026-09-05. [Online]. Available: https://github.com/loraalex/LoRaFIIT

[30] J. S. Nixon, J. J. A. Celin, A. Senthil Kumar, C. Ganesh, U. P. Kumar, and D. Kumar, "LoRaWAN Security Framework to Mitigate DoS Attacks at the Gateway Level using Machine Learning," in *2024 IEEE Pune Section International Conference (PuneCon)*, Pune, India, 2024, pp. 1–3.