Field Devices Under Fire: A Security Assessment of Handheld Radionuclide Isotope Identification Devices

Justin Carney
Augusta University
Augusta, Georgia, USA
jucarney@augusta.edu

Jeffrey Morris
Augusta University
Augusta, Georgia, USA
jefmorris@augusta.edu

Anthony Rizi
Augusta University
Augusta, Georgia, USA
arizi@augusta.edu

Edward Tremel
Augusta University
Augusta, Georgia, USA
etremel@augusta.edu

Michael Nowatkowski
Augusta University
Augusta, Georgia, USA
mnowatkowski@augusta.edu

Abstract

Handheld Radionuclide Isotope Identification Devices (RIIDs) are widely used in homeland security, emergency response, and nuclear safeguards operations. Despite their operational importance, there is limited independent analysis of their comparative performance and almost no public assessment of their security posture. This paper presents a technical examination of a leading commercial system—the ORTEC RADEAGLE —focusing on embedded-system architecture, wireless communication behavior, and susceptibility to modern attack surfaces. Drawing on vendor documentation, empirical interface testing, firmware analysis, and automated vulnerability scanning, we identify systemic risks across multiple layers, including weak authentication, clear-text data transmission, outdated JavaScript libraries, missing security headers, and firmware components that exhibit indicators of unsafe memory handling. Our assessment demonstrates that compliance with external standards such as ANSI N42.34 and IEC 61000-4 does not guarantee resilience against hardware-or software-level threats. These findings underscore the need for a formalized cybersecurity evaluation process for radiation-detection field devices and provide a foundation for strengthening future procurement, deployment, and regulatory guidance.

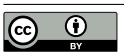
CCS Concepts

• **Security and privacy** → **Embedded systems security**; *Software and application security*; • **Hardware** → Sensor devices and platforms.

Keywords

RIID, radiation detection, security assessment, Bluetooth, Wi-Fi, embedded systems

ACM Reference Format:
Justin Carney, Jeffrey Morris, Anthony Rizi, Edward Tremel, and Michael Nowatkowski. 2026. Field Devices Under Fire: A Security Assessment of Handheld Radionuclide Isotope Identification Devices. In *2026 ACM Southeast Conference (ACMSE 2026), April 23–25, 2026, Troy, AL, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3746467.3801514



ACMSE 2026, April 23–25, 2026, Troy, AL, USA

ACM ISBN 979-8-4007-2062-8/2026/04
https://doi.org/10.1145/3746467.3801514

1 Introduction

RIIDs operate in mission-critical contexts-homeland security, emergency response, and nuclear safeguards where false negatives or device denial-of-service have operational and safety consequences. Contemporary RIIDs incorporate commodity computing features such as embedded Linux, USB/Bluetooth/Wi-Fi connectivity, and browser-based interfaces. While ANSI N42.34 specifies detection/identification performance and IEC 60529 specifies ingress protection (IP codes), none of these standards include cybersecurity requirements such as authentication, encrypted transport, secure boot, or configuration integrity validation. As a result, vendors can be standards-compliant yet lack basic cybersecurity protections [1, 2, 9, 15].

This paper addresses this gap through a security-focused five-layer analysis of a commercially available RIID (the ORTEC RADEAGLE). Our evaluation examines the physical layer, communication interfaces, operating system and firmware, application and web services, and the human interface. The assessment combines documentation review, interface enumeration, filesystem inspection, firmware analysis, automated web-application scanning, and a controlled proof-of-concept (PoC) experiment targeting configuration-integrity weaknesses.

Our findings reveal systematic vulnerabilities across all layers, including clear-text network services, hard-coded credentials, and unsigned configuration files that can disable the isotope identification, outdated client-side libraries with known CVEs, missing security headers, and the absence of secure-boot or code-signing protections. The PoC demonstrates that configuration tampering can suppress nuclide identification without triggering any user-visible integrity warnings.

The contributions of this work are fourfold:

(1) A five-layer security assessment of a mission-critical field device;
(2) A clear articulation of the *standards gap* affecting RIIDs, identifying what ANSI N42.34/N42.42/IEC 60529 cover and what cybersecurity requirements they omit;
(3) A controlled demonstration of how configuration-integrity failures impair operational detection; and

(4) A consolidated set of mitigations spanning integrity, access control, transport security, hardening, file permissions, and operational controls.

Section 2 provides background on RIIDs, vendor-documented capabilities, and relevant standards. Section 3 presents our layered security assessment and analysis methods. Section 4 details the PoC experiment and qualitative operator observations. Section 5 presents recommended mitigations, and Section 6 concludes with implications and future work.

Our analysis is structured into five layers: the physical layer, communication interfaces, the operating system and firmware, application and web services, and the human interface. This layered organization reflects how the attack surface naturally decomposes in embedded systems and supports a systematic, repeatable evaluation process.

2 Background

2.1 RADEAGLE Overview

Handheld RIIDs are designed to detect and identify gamma-emitting radionuclides in portable, time-sensitive field environments. Vendor documentation for ORTEC's innoRID line describes devices such as the RADEAGLE as rugged, battery-powered instruments suitable for survey, emergency response, and safeguards applications [15, 16]. These devices provide real-time spectrum viewing, dose-rate monitoring, optional neutron detection, and remote access via USB, Bluetooth, and Wi-Fi. They operate across wide environmental ranges and support multiple identification modes, emphasizing probability and ease of use rather than cybersecurity posture [9].

2.2 Standards

The RADEAGLE cites compliance with ANSI N42.42 for detection and identification performance and exports data in the ANSI N42.34 format [9]. Its enclosure meets IEC 60529 ingress-protection ratings and satisfies EMC Directive 2014/30/EU [9, 15]. ANSI N42.34 defines detection sensitivity, identification accuracy, response time, and environmental robustness, while IEC 60529 defines resistance to dust and liquid intrusion. These performance and environmental requirements do not address cybersecurity concerns.

Standards gap.

While ANSI N42.34 and N42.42 ensure performance and data-format requirements, and IEC 60529 addresses enclosure ingress protection, none of these standards require basic cybersecurity controls for connected RIIDs—e.g., authenticated administration, TLS on remote interfaces, secure boot, firmware/configuration signing, or role-based access. Our findings, such as clear-text services, default credentials, and unsigned configuration/firmware, arise in the context of the current standards and demonstrate that these vulnerabilities can persist even in devices that fully meet existing performance and environmental requirements [1, 2].

2.3 Prior Comparative Studies

Studies conducted by Savannah River National Laboratory (SRNL) and others [10] primarily focus in validating the performance standards that guide commercial RIIDs or on comparing different detector technologies. These comparisons establish benchmarks for energy resolution, identification time, false-alarm rates, and environmental robustness, offering context for how compliant devices are expected to operate under ANSI N42.34. While these studies are vulnerable for evaluating radiological performance, they do not assess cybersecurity posture or the risks associated with modern network-enabled features.

2.4 Wireless and Embedded Device Security Context

The inclusion of commodity wireless communication protocols in embedded instruments such as Bluetooth Low Energy (BLE) and Wi-Fi introduces security challenges common across the Internet of Things (IoT) ecosystem. BLE security surveys show susceptibility to Man-in-the-Middle (MITM) attacks, allowing adjacent attackers to intercept or modify transmitted data [4]. BLE implementations are also affected by chipset-level denial-of-service vulnerabilities (e.g., the SweynTooth family), which can crash the communication stack and disrupt telemetry or control channels [4]. Companion mobile applications that interface with BLE devices often employ insecure authentication mechanisms or weak API protections, increasing the risk of unauthorized access to configuration data. Additionally, many embedded devices lack a secure, authenticated Over-The-Air (OTA) update mechanism, leaving them persistently vulnerable to newly disclosed security flaws. When these weaknesses appear in mission-critical devices, such as RIIDs, the operational impact extends beyond confidentiality concerns to include the integrity and availability of safety-relevant alarms and measurements.

2.5 Related Work

Marques et al. [12] reviewed mobile radiation detection (MRD) systems, highlighting innovations in compact detectors, UAV integration, and cooperative sensor networks. They note connectivity features such as Wi-Fi and USB for real-time data transfer and remote control, which enhance flexibility but introduce potential attack surfaces. Our work examines these connectivity channels in RIIDs and identifies vulnerabilities, including clear-text communication, outdated web components, and weak authentication mechanisms. By proposing a formalized cybersecurity evaluation process, we complement performance-oriented studies and ensure that future radiation detection technologies remain resilient against evolving threats.

Bluetooth security research has advanced through formal verification and automated fuzzing techniques. Wu et al. [19] introduced a ProVerif-based model for Bluetooth Classic, BLE, and Mesh protocols, verifying 418 security properties and uncovering 82 violations, including two previously unknown vulnerabilities. Hou et al. [8] complemented this with a machine-learning-driven fuzzing framework that discovered eight vulnerabilities across consumer and IoT devices, illustrating the scalability of automated approaches for protocol-level security. While these studies significantly improve protocol assurance, they focus on general-purpose devices and do not address cyber-physical risks in mission-critical systems.

Chichester et al. [6] developed portable telemetry systems for post-blast radiological dispersal device (RDD) scenarios, integrating GPS and wireless communication to support geospatial mapping and forensic analysis. Their work advanced emergency response

capabilities by enabling real-time data collection and reconstruction of radionuclide distribution. However, these systems prioritize operational efficiency without considering the security posture of connected instruments, leaving potential attack surfaces unaddressed.

Makoye and Mina [11] propose a technology-driven roadmap for border nuclear security, emphasizing advanced detection hardware, imaging systems, and AI-powered analytics to improve interdiction capabilities. Their vision reflects the growing reliance on intelligent systems for rapid threat detection and decision support. However, these approaches assume compliance with ANSI N42.34 [2] guarantees robustness, overlooking device-level cybersecurity.

3 Methodology

3.1 Layered Security Assessment

We assess connected field devices across five layers:

(1) **Physical/Hardware**: enclosure/ports inspection, tamper avenues, boot-integrity assumptions.
(2) **Communication Interfaces**: USB/BLE/Wi-Fi/Serial enumeration, traffic capture, service/credential discovery.
(3) **OS/Firmware**: OS inventory, services/accounts, update path, boot chain, persistence.
(4) **Application/Web Services**: web UI review, security headers, dependency versions, automated DAST (OWASP ZAP) with manual confirmation.
(5) **Human Interface**: operator controls, alarm visibility, and safety-critical toggles under normal and degraded states.

This layered security assessment guided our experiments and is portable to other RIIDs and embedded devices with smaller architecture.

3.2 Physical/Hardware

Through physical inspection, we discovered that the RADEAGLE incorporates an off-the-shelf embedded computer—specifically a BeagleBone-class single-board computer (SBC), as illustrated in Figure 1—rather than a purpose-built, hardened hardware platform [7]. The use of a commodity SBC for core processing and communications introduces significant physical-layer security considerations that warrant careful evaluation. The BeagleBone Black (BBB), much like a Raspberry Pi–class device, expands the attack surface by exposing accessible interfaces, removable storage, and general-purpose debugging capabilities. Prior research on commercial SBC security highlights two primary threat categories:

(1) **Side Channel Attack (SCA) Vulnerability:** Different platforms like the Beagle Board ARM Cortex-A8 and Raspberry Pi have been explicitly shown to be susceptible to SCA attacks. Encryption keys or any other identification algorithms can be exploited by passively monitoring different power consumption phases or electromagnetic emissions during different cryptographic operations [3]. Although the RADEAGLE device must be shielded for EMC (per IEC 61000-4), it does not account for internal hardware leakage that can potentially be measured at different debug ports or power traces. What this means is that the device can still be subjected to layer 1 attacks via physical access.

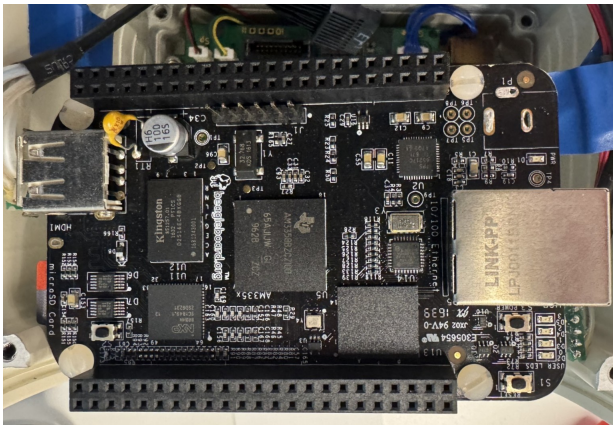


Figure 1: BeagleBoard-Class Embedded Processor

(2) **Physical Persistence and Espionage:** Due to the use of a BeagleBoard-class single-board computer running a general-purpose Linux operating system, the device exposes standard hardware interfaces common to embedded platforms, such as USB ports, storage media and potential debugging headers. These interfaces on the BeagleBoard are not inherently secure; if left unprotected, they may create opportunities for physical tampering. A threat actor with physical access could attempt to replace the storage media, inject modified firmware, or install a hardware implant. The risk arises not from the hardware platform itself, but from the absence of preset secure boot mechanisms, firmware signing, and tamper-resistance controls. Without the cryptographic verification of the system software, the persistent modification of the device behavior becomes highly feasible under physical access conditions.

The first aspect of this investigation must therefore focus on the physical security of the device, including access to internal debugging interfaces, shielding effectiveness against internal leakage, and the possibility of hardware-based tampering that leverages the embedded platform's low-cost, open-source hardware design ethos.

3.3 Communication Interfaces

The RADEAGLE implements a variety of communication protocols, supporting wired and wireless connectivity. Supported methods include Universal Serial Bus (USB 2.0) for host and mass-storage connections, Bluetooth 802.15.1, Wi-Fi 802.11(a–e) (via dongle or built-in for submersible models), and legacy Serial ports (RS-232-RS-485, UART). The USB-A host connector is used for communication dongles [9, 14].

USB. Provides two dedicated connections for data management and host interaction. The USB Mini-B connector enables direct Host-PC connection, where the device enumerates and presents the web interface at the IP address `192.168.7.2` on the `usb0` interface. Host - PC connection (USB-Mini-B, IP 192.168.7.2) [9] and mass-storage (USB-A) for bulk transfer of configuration files and firmware [9]. The USB-A Host port accommodates communication dongles (Wi-Fi/Bluetooth) or external mass-storage devices.

This port supports connecting FAT32 mass-storage devices (e.g. USB sticks) for bulk transfer and backup of all saved data (spectra, screenshots, etc.). Connection via USB cable may require special driver software on the host computer, depending on the operating system.

Bluetooth. Functionality requires a Bluetooth dongle plugged into the USB-A Host port of the device. The system supports Bluetooth Class 2.0, which enables it to pair with external devices, typically smartphones, for remote control and data services. Once paired, the device displays the web interface address, for example `http://172.20.10.4`, which facilitates remote access and allows the sharing of the internet connection (tethering). The user interface provides comprehensive control used to manage the Bluetooth status (on/off), initiate the pairing process, and remove known connected devices. Bluetooth is also used for web access and user interface, and for sharing the mobile device's internet connection for data transmission. None of these communication paths employs mutual authentication or encryption, leaving data in transit susceptible to passive eavesdropping and active man-in-the-middle attacks, as documented in established Bluetooth security guidance [17].

Wi-Fi. The device can join an existing Wi-Fi network (SSID supplied by the operator) or activate its built-in hotspot (default SSID `<device serial number>`, password `Astronomy`) as described in the user manual [9]. When operating as a hotspot, the device advertises the address `192.168.0.1` and presents a web UI that displays the current dose-rate, allows real-time spectrum streaming, and provides hotspot management (SSID/password display) [9]. Continuous dose-rate reporting and live spectrum data are transmitted over the Wi-Fi link to the web interface, allowing remote monitoring and data export [9, 14]. While connected to a Wi-Fi network, either an existing network or a hotspot, the device exposes several network services in addition to its HTTP web interface, as confirmed by a port scan. These include an SSH server (port 22), an SMB server (ports 139 and 445), and an Nginx web server listening on nonstandard ports 3000, 4000, 8080, and 8888. The SSH server supports basic username and password authentication, and the device is configured with a default user account whose password matches the username. This allows remote access to the device, as demonstrated by our PoC scripts: they use `nmcli` (or `iwconfig`) to connect to the SSID with the default password, then interact with the device over SSH to make malicious configuration changes.

3.4 OS/Firmware

The RADEAGLE runs an embedded Linux distribution (Debian 8 "jessie", kernel 4.9.10) on a BeagleBoard class ARMv7 processor. Functionality is implemented via a mixture of Python code, Java programs, and C libraries.

The on-device UI is a Python program, running in Python 3.4.2, that launches when the device boots up. This program's source code is stored unobfuscated in the directory `/root/quest`, enabling it to be easily analyzed for vulnerabilities. It stores its configuration options as a Python pickled (serialized) dictionary object at `/home/innouser/data/configuration.ird`, and it reads this file on startup to initialize its configuration. These configuration options include sensitive values such as the password for the "Protected Settings" screen and the list of identifiable nuclides along with their classification as "Threat" or "Innocent." When reading the configuration file, the program performs only minimal validation on it; any configuration file that has all of the expected key names in its dictionary will be accepted.

The embedded web interface, `webEAGLE`, is hosted on an nginx server and uses several JavaScript libraries. Specifically, it exposes the vulnerable JavaScript libraries: Moment.js v2.22.2 (CVE-2022-31129 – ReDoS, CVE-2022-24785 – path traversal) and Vue.js v2.5.17 (CVE-2024-9506 – ReDoS) [14]. Critical HTTP security headers such as Content-Security-Policy, X-Frame-Options, and HSTS are missing [14]. Unfortunately, no transport-level encryption (TLS/VPN) is enforced, leaving all traffic in clear-text and susceptible to eavesdropping or tampering [14]. In addition, firmware-update channels also lack security evaluation [9].

The device's security architecture includes authentication mechanisms, data-at-rest and in-transit protection, secure boot, code signing, update verification, and compliance with standards.

(1) **Authentication**: Remote access to the device through the web interface and Bluetooth generally does not require authentication; the device's Wi-Fi hotspot uses a short, well-known password. The device's SSH server requires a username and password for SSH authentication, but the default user account has a username and password that are easy to guess. This account has root privileges and can execute privileged commands without entering a password.
(2) **Data-at-Rest Encryption**: None. The device's source code and all the data it records are stored unencrypted in the file system.
(3) **Data-in-Transit Encryption**: Almost entirely absent. All protocols used by the device transmit clear-text data (see Section 3.3), except for the SSH server, which uses standard transport encryption for the SSH session.
(4) **Secure Boot / Code Signing**: No cryptographic verification of firmware, binaries, or source code; the system is vulnerable to persistent malicious implants.
(5) **Security Headers**: CSP, HSTS, X-Frame-Options, etc., are missing from the web UI.

3.5 Application/Web Services

We analyzed the device's embedded web interface using OWASP Zed Attack Proxy (ZAP), an open-source web application security scanner [5]. There was a range of alerts categorized by severity visualized in Figure 2, each with implications for the application's security; the identified JavaScript patterns and other findings were mapped to their corresponding potential Common Vulnerabilities and Exposures (CVEs) [13]. These are publicly disclosed, cataloged security vulnerabilities in software or hardware, each identified by a unique CVE ID to standardize reporting and facilitate tracking and remediation. As shown in Table 1, the RADEAGLE JavaScript findings have been mapped to potential CWE categories along with recommended mitigations, sorted by severity. The most critical findings involved the use of vulnerable JavaScript libraries, specifically Moment.js version 2.22.2 and Vue.js version 2.5.17. Moment.js

was found to contain two significant vulnerabilities. CVE-2022-31129 describes a quadratic-time complexity issue in RFC2822 date parsing that can lead to Regular Expression Denial of Service (ReDoS) attacks when processing long user-supplied strings. CVE-2022-24785 exposes a path traversal vulnerability when unsanitized locale strings are used, potentially allowing unauthorized access to files. Vue.js is similarly affected by CVE-2024-9506, where improper regular expression handling in the parseHTML function may also lead to ReDoS. These findings highlight the importance of maintaining up-to-date dependencies and implementing strict input validation.

Medium-severity alerts included the absence of Content Security Policy (CSP) headers, increasing the risk of Cross-Site Scripting (XSS) and data injection attacks. CSP headers are crucial for identifying trusted content sources and mitigating client-side threats. Additional medium alerts identified hidden files accessible on the device/server, which may expose sensitive configuration or credential data. The lack of anti-clickjacking headers such as X-Frame-Options or Content-Security-Policy with frame-ancestors directives may leave the application vulnerable to UI redress attacks. Furthermore, the use of URL rewriting to track session IDs may pose risks of session hijacking through referer leakage and exposure of browser history.

Low-severity alerts focused on cookie security and information leakage. Cookies set without the SameSite attribute are vulnerable to cross-site request forgery (CSRF), while the presence of X-Powered-By and Server headers discloses framework and version information that could aid attackers in fingerprinting and targeted exploitation. The absence of the X-Content-Type-Options: nosniff header enables MIME-sniffing, which may result in misinterpreted content types and unintended script execution. Informational alerts, though not immediately exploitable, revealed potential misconfigurations and areas for further hardening. Cookie poisoning was identified through user-supplied input in query strings and POST data, suggesting improper input validation. Suspicious comments

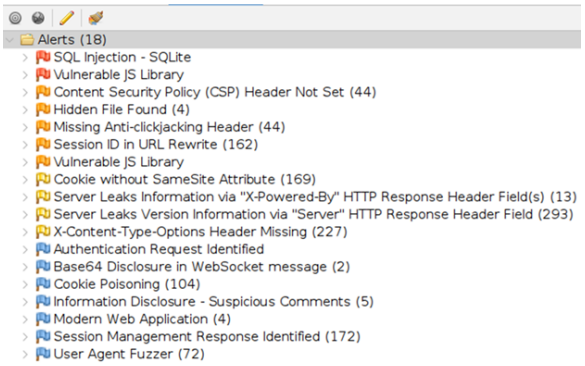


Figure 2: Overview of ZAP Results

and indicators of modern web application frameworks may hint at development artifacts or client-side vulnerabilities. Session management responses and potential susceptibility to user-agent fuzzing further underscore the need for robust session handling and input sanitization.

RADEAGLE Javascript Review. Several objects and functions were revealed that warrant scrutiny due to their potential security implications. The use of document.write in multiple locations is concerning, as it can introduce XSS vulnerabilities if not properly sanitized. Similarly, multiple .innerHTML assignments may expose the application to DOM-based XSS if user input is directly injected into the DOM without escaping. The code also includes dynamic DOM manipulation through createElement and appendChild, which, while common in modern web development, must be carefully managed to avoid injection risks.

Storage mechanisms such as localStorage and session.cookie were observed, indicating possible persistence of client-side data.

Table 1: Mapped ZAP Javascript Findings

CWE ID & Title	Pattern/Finding	Recommended Severity	Risk/Impact	Mitigations
CWE-676; CWE-79 (XSS)	document.write usage	High	Allows injection of attacker-controlled scripts	Avoid document.write; use safe templating; enforce CSP; sanitize input
CWE-79; CWE-116	Assignments to innerHTML	High	DOM-based XSS through HTML/script injection	Use textContent or sanitizers; centralize output encoding; strict CSP
CWE-79; CWE-116	Dynamic DOM creation with untrusted strings	High	Injection via attacker-controlled tag names/attributes	Whitelist elements/attributes; sanitize HTML; use Trusted Types
CWE-312; CWE-565	Client-side storage of sensitive data	High	Credentials/session theft; offline compromise	Avoid secrets in localStorage; use HttpOnly/Secure cookies; rotate tokens
CWE-16	Missing/weak security headers	High	XSS, clickjacking, mixed content exposure	Enable HSTS, CSP, frame-ancestors, nosniff, Referrer/Permissions Policy
CWE-1104	Outdated third-party JS libraries	High	Known CVEs; supply-chain risk	Perform SCA scanning; update dependencies; use SRI
CWE-20	Improper input validation	High	Injection, traversal, parser confusion	Centralize validation; strict schemas; canonicalize input; server-side checks
CWE-915	Prototype manipulation (__proto__)	Medium	Prototype pollution; logic corruption	Avoid __proto__; use Object.create(null); sanitize JSON merges
CWE-601	Unvalidated redirects (window.location)	Medium	Open redirect; phishing; token leakage	Enforce allowlists; validate server-side; block javascript:/data: URLs
CWE-83 (XSS in event handlers)	Inline event handlers (e.g., onclick)	Medium	Triggering attacker scripts through injected attributes	Use addEventListener; avoid inline handlers; disallow inline CSP

These should be used with caution, especially when storing sensitive information, as they are accessible via JavaScript and vulnerable to theft through XSS. The use of __proto__ may introduce prototype pollution risks, leading to unexpected behavior or privilege escalation in JavaScript objects. Navigation functions like window.location.replace and window.location.href were also present, which can be exploited for open redirect attacks if user-controlled input is used. Finally, event handlers such as onclick and onmouse were found embedded in the code. These can be leveraged to execute malicious payloads if tied to unannounced user input or used in conjunction with vulnerable DOM elements. Collectively, these JavaScript features highlight the importance of secure coding practices, especially in client-side logic, where user interaction and dynamic content generation are prevalent. In conclusion, the findings from OWASP ZAP, RADEAGLE, and manual code inspection demonstrate a spectrum of vulnerabilities ranging from critical third-party library flaws to subtle misconfigurations and insecure JavaScript practices. Addressing these issues requires a multi-layered approach, including dependency updates, secure HTTP headers, rigorous input validation, and careful management of client-side scripting.

3.6 Human Interface

3.6.1 Usability, Visual Indicators, UI Design. Interaction is performed via a three-button keypad beneath the screen using visual indicators. As stated in the manual, the status bar shows battery, time, and connectivity; the dose-rate bar uses color-coded WARN (yellow) and ALARM (red) zones. LED indicators show Gamma Alarm (red), optional Neutron Alarm (blue), Battery Charge (green), and Battery Failure (red).

3.6.2 Field Operator Ergonomics and Cognitive Load. UI features reduce cognitive load by using the Easy-ID Mode. This provides for minimal interaction. This identifies up to four isotopes in 30 s with distance-guidance messages (Too Close, Optimum, Too Far).

4 Proof-of-Concept Exploit

To evaluate the security posture of the RADEAGLE, we conducted a supervised, controlled proof-of-concept (PoC) experiment to assess the resilience of its configuration management controls. The aim was to determine whether the device's isotope-classification parameters could be programmatically altered without triggering the embedded safeguard mechanisms.

The PoC exploit was implemented in Python 3.4.2 and executed after making an SSH connection. The goal was to determine whether the configuration file—responsible for isotope-specific detection thresholds—could be modified to suppress identification functions. The PoC successfully demonstrated that the configuration could be altered without protective mechanisms preventing the change.

Specifically, the exploit was delivered with the following steps:

(1) Connect to the device's Wi-Fi network
(2) Make an SSH connection to the device's IP address (which will always be 192.168.0.1 if using the device's Wi-Fi hotspot) with the default administrator username and password
(3) Upload a Python script, written by the attacker, using scp. The script will open the `configuration.ird` configuration file, un-pickle the dictionary, replace the key "NuclideIdConfigurationString" with a new value in which every nuclide is set to disabled, and write a new pickled object back to `configuration.ird`.
(4) Run the attacker's Python script
(5) Restart the UI program, either by rebooting the device or by issuing the command `pkill python; python3 /root/quest/questSplashScreen.py /root/quest/`. This causes the UI program to reload `configuration.ird`.
(6) After the UI restarts, all nuclides will be configured to have detection and alerts disabled.

(a) Cs-137 Container **(b) PoC Experiment Setup**

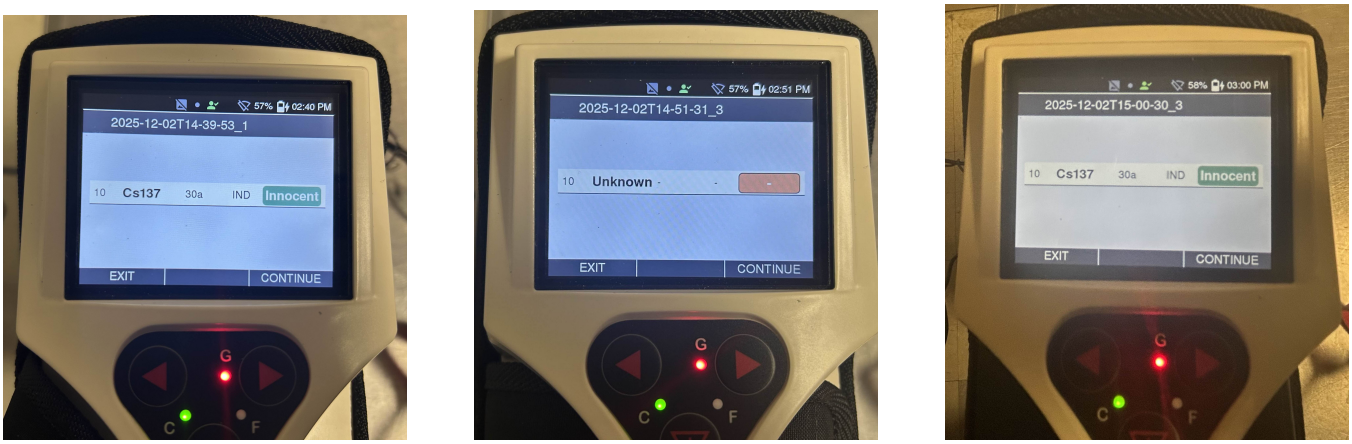
Figure 3: Cs-137 Training Source for Experiment

To assess operational impact, we were granted authorized and supervised access to a training isotope, Cesium-137 (Cs-137). This radioactive isotope is commonly used in industrial gauges, medical calibration sources, and radiation training simulators due to its well-characterized gamma emissions. It is a fission byproduct with a predictable decay profile, making it a safe and controlled choice for laboratory or instructional environments when handled in accordance with proper regulations [18]. As seen in Figure 3a, the Cs-137 container was labeled with an activity of 0.232 mCi (8.58 MBq), indicating the source undergoes approximately 8.58 million decays per second. This measure reflects the radioactive strength of the training isotope used for controlled evaluation.

While conducting the PoC experiment, the Cs-137 isotope was placed in front of the RADEAGLE within the RADEAGLE's detection range, as shown in Figure 3b.

4.1 Experiment Phases

The evaluation occurred in two stages. First, the PoC was conducted via a direct USB connection to confirm feasibility under controlled physical access conditions. Next, the Wi-Fi phase of the assessment was conducted from an independent laptop connected to the device's built-in Wi-Fi access point. This phase consisted of three subphases: Subphase 2a, in which the device was tested under



(a) Correct Identification Before the Exploit **(b) Faulty Identification After the Exploit** **(c) Identification After Original Configuration Restored**

Figure 4: Proof-of-Concept Phases

normal operating conditions to establish a functionality baseline (Figure 4a); Sub-phase 2b, in which the exploit was delivered and executed remotely, causing the device to fail in correctly classifying the isotope Cs-137 (Figure 4b); and Sub-phase 2c, in which the original configuration was restored and the device was retested, confirming that standard detection functionality had returned (Figure 4c).

4.2 Controlled Evaluation Results

Under normal conditions, the RADEAGLE correctly identified the training source. After executing the PoC, however, the device was unable to classify the isotope, confirming that the altered configuration directly impacted detection fidelity.

After the evaluation, the original configuration was restored using the SSH interface, by uploading an unmodified copy of the `configuration.ird` file and restarting the Python UI. This ensured full restoration of operational integrity and compliance with safety and test protocols.

4.3 Experiment Observations

The proof-of-concept exploit revealed a series of severe weaknesses that align with Common Weakness Enumeration (CWE) categories [13], as summarized in Table 2. The most critical issues involved Missing Authentication for Critical Function (CWE-306), in which the device permitted modification of sensitive configuration files without any secondary authentication beyond the initial SSH login, and Improper Access Control (CWE-284), in which SSH access effectively granted unrestricted control over critical operations. Equally critical was the absence of integrity verification for configuration files—categorized as Download of Code Without Integrity Check (CWE-494)—which exposed the system to undetected tampering. The device also relied on static, hard-coded credentials, representing Use of Hard-coded Credentials (CWE-321), a severe and easily exploitable authentication weakness. Additionally, the system demonstrated Protection Mechanism Failure (CWE-693), lacking safeguards to prevent disabling or bypassing essential detection capabilities.

High-severity issues were also present. These included Incorrect Permission Assignment for Critical Resource (CWE-732), where configuration files lacked appropriate permission boundaries, and Insecure Handling of Pickled Data (CWE-1188), as the device relied on Python's pickle mechanism without validation, creating deserialization and potential code execution risks.

Finally, the assessment identified a broader Medium–High severity Configuration Management Flaw (CWE-16), reflecting the absence of structured controls to ensure that configuration changes are safe, verified, and auditable. Addressing these weaknesses requires implementing cryptographically signed configuration files, enforcing strict role-based access controls, eliminating hard-coded credentials, and strengthening configuration management processes to maintain system integrity and operational resilience.

Devices deployed in field or inspection environments must assume the possibility of both physical and wireless access vectors. As demonstrated, insufficient configuration protection can unintentionally suppress key detection capabilities. Enhancements such as cryptographically signed configuration files, integrity attestation, and stricter access controls would mitigate such risks in future deployments.

5 Mitigations

Portable radiological detection systems must be secured as critical cyber-physical devices rather than treated as simple measurement instruments. The mitigation strategies presented below consolidate technical and operational controls into deployment-ready guidance aimed at addressing the vulnerabilities identified in earlier sections. The CVE/CWEs discussed in Sections 3.4 and 4.3 can be effectively reduced by applying established security principles and managing the device as a high-value asset within the operational environment.

(1) **Configuration and Firmware Integrity.** Devices should enforce cryptographic integrity checks on all configuration

Table 2: Mapped Device Weaknesses

CWE ID	Weakness Title	Description of Observed Issue	Severity	Recommended Mitigations
CWE-306	Missing Authentication for Critical Function	Device allowed modification of sensitive configuration files without secondary authentication or privilege validation.	**Critical**	Add multi-factor or secondary authentication; enforce privilege separation.
CWE-284	Improper Access Control	SSH access provided unrestricted control over critical device operations without granular authorization checks.	**Critical**	Implement RBAC; restrict SSH privileges; enforce least privilege.
CWE-494	Download of Code Without Integrity Check	No integrity or authenticity verification performed on configuration files, enabling undetected tampering.	**Critical**	Use cryptographically signed configuration files; enforce checksum/signature validation.
CWE-321	Use of Hard-coded Credentials	Device used a static username and password for authentication, undermining security posture.	**Critical**	Eliminate hard-coded credentials; require secure provisioning; enforce unique credentials.
CWE-693	Protection Mechanism Failure	System lacked safeguards preventing the disabling or bypassing of essential detection and monitoring capabilities.	**Critical**	Implement tamper protection; secure integrity checks; restrict runtime access to protection mechanisms.
CWE-732	Incorrect Permission Assignment for Critical Resource	Configuration files lacked granular permission enforcement, allowing unauthorized or overly broad access.	**High**	Apply strict file permissions; enforce least privilege; isolate critical resources.
CWE-1188	Insecure Handling of Pickled Data	Unvalidated use of Python pickle enabled potential deserialization and code execution risks.	**High**	Use safe serialization formats (JSON/MessagePack); validate inputs; avoid pickle for untrusted data.
CWE-16	Configuration Management Flaw	Lack of structured configuration management controls allowed unsafe modifications to critical system settings.	**Medium–High**	Enforce configuration change controls; require signed configs; maintain audit logging.

files and firmware images. Signature verification must occur at boot and during any update process to prevent unauthorized modification. Incorporating a hardware root of trust further protects against persistent tampering by ensuring the system only boots from validated, manufacturer-approved firmware.

(2) **Access Control.** Eliminating hard-coded and default credentials significantly reduces the risk of unauthorized access. Each device should use unique administrative accounts and implement role-based access control (RBAC) for all maintenance interfaces. Applying least-privilege principles and disabling passwordless privilege escalation limits both accidental misuse and adversarial control pathways.

(3) **Transport Security.** All remote, network-exposed, or web-based interfaces should require TLS to protect credentials and command channels. When remote administration is necessary, mutual-TLS or VPN encapsulation provides additional assurance. Unencrypted services and unused ports should be disabled by default to minimize unnecessary network attack surface.

(4) **Update Mechanisms and Platform Hardening.** A secure, authenticated update channel is essential for maintaining device integrity over time. Firmware and software updates should require signature verification before installation. Embedded web interfaces must use current, supported client-side libraries, and standard security headers – such as HSTS, CSP, X-Frame-Options, and X-Content-Type-Options – should be deployed to provide defense-in-depth against common exploitation vectors.

(5) **File and Process Permission Controls.** Sensitive files, including configuration data and nuclide-identification parameters (e.g., configuration.ird), should be tightly restricted in terms of read/write permissions. System services should operate using isolated, minimally privileged accounts. All changes to configuration or identification logic must be logged and auditable to ensure traceability and support incident response.

(6) **Operational Controls.** Communication interfaces such as Wi-Fi, Bluetooth, and SSH should remain disabled unless operationally required. Agencies should rotate credentials regularly and integrate pre- and post-use checks that verify configuration integrity and confirm proper operation of the nuclide identification engine. These operational measures reinforce technical controls and ensure the device remains in a secure, validated state throughout its use.

6 Conclusion and Future Work

This paper examined a widely deployed handheld radiation detection system–the ORTEC RADEAGLE–through the combined lenses of performance characteristics, embedded-system architecture, and cybersecurity exposure. Our analysis shows that although the device meets established operational and electromagnetic compatibility standards, it does not adequately address contemporary cyber-physical threats. Clear-text communication channels, hard-coded credentials, outdated libraries, and insufficient firmware safeguards illustrate a broader pattern: RIIDs have evolved in sensitivity and interoperability, but their security controls have not kept pace.

The findings presented highlight two critical gaps. First, organizations lack independent, publicly available data to compare commercial RIIDs beyond vendor-supplied specifications. Second, current certification pathways do not incorporate a meaningful cybersecurity assessment, despite the increasing reliance on devices on Linux-based systems, wireless connectivity, and complex web interfaces. As emergency response, border security, and critical infrastructure protection continue to depend on portable radiation-detection systems, the absence of a security-aware design introduces an avoidable operational risk.

While this paper concentrates on an RIID–which may not be a widely deployed device–the techniques and processes demonstrated here are broadly applicable to security assessments of a much wider class of embedded systems and IoT devices. The RADEAGLE itself is built on a single-board computer running a standard Linux distribution with USB ports, network connectivity, and an integrated display, an architecture shared by numerous commercial and industrial devices. As such, the methodology used here can serve as a model for evaluating other instrument families that rely on similar commodity hardware, common operating system components, and software-defined functionality. Many classes of sensors, handheld diagnostic tools, and specialized field-deployed equipment mirror this design pattern, making them equally susceptible to configuration tampering, insecure interfaces, and software-centric attacks.

The next phase of this research will involve performing a full security assessment of an additional RIID device to evaluate whether similar vulnerabilities and attack pathways exist across product generations and vendors. This extended analysis will follow the same structured procedures for vulnerability discovery, attack-surface enumeration, exploit development, and proof-of-concept validation used in the present study. By applying a repeatable and rigorous workflow, the goal is not only to identify weaknesses in individual devices but also to refine and formalize a general security-assessment process that can be adopted by practitioners, researchers, and manufacturers. Ultimately, the intent is to create a portable, reproducible methodology capable of supporting comparative analysis across devices, guiding mitigations, and improving the cybersecurity posture of embedded systems more broadly.

References

[1] American National Standards Institute. 2020. *American National Standard Data Format for Radiation Detectors Used for Homeland Security.* Technical Report. American National Standards Institute. 311 pages. https://doi.org/10.1109/IEEESTD.2020.9264644 ANSI N42.42-2020 (Revision of ANSI N42.42-2012), IEEE Std 324-2020.

[2] American National Standards Institute. 2021. *American National Standard Performance Criteria for Handheld Instruments for the Detection and Identification of Radionuclides.* Technical Report. American National Standards Institute. https://doi.org/10.1109/IEEESTD.2021.9519594 ANSI N42.34-2021 (Revision of ANSI N42.34-2015).

[3] Jorge Barredo, Maialen Eceiza, Jose Luis Flores, and Mikel Iturbe. 2025. CARNYX: A Framework for Vulnerability Detection via Power Consumption Analysis in Embedded Systems. *International Journal of Information Security* 24, 4 (Aug. 2025), 23. https://doi.org/10.1007/s10207-025-01092-2 Accepted: 23 June 2025; Published: 08 July 2025.

[4] Arup Barua, Md Abdullah Al Alamin, Md. Shohrab Hossain, and Ekram Hossain. 2022. Security and Privacy Threats for Bluetooth Low Energy in IoT and Wearable Devices: A Comprehensive Survey. *IEEE Open Journal of the Communications Society* 3 (2022), 251–281. https://doi.org/10.1109/OJCOMS.2022.3149732

[5] Checkmarx. 2025. *ZAP Documentation.* https://www.zaproxy.org/docs/

[6] David L. Chichester, James T. Johnson, Scott M. Watson, Scott J. Thompson, Nick R. Mann, and Kevin P. Carney. 2016. Post-blast Radiological Dispersal Device Source Term Estimation. In *2016 IEEE Nuclear Science Symposium, Medical Imaging Conference and Room-Temperature Semiconductor Detector Workshop (NSS/MIC/RTSD).* IEEE, Strasbourg, France, 1–3. https://doi.org/10.1109/NSSMIC.2016.8069920

[7] Gerald Coley. 2013. *BeagleBone Black System Reference Manual* (revision a5.2 ed.). https://circuitco.com/support/index.php?title=BeagleBoneBlack BeagleBoard.org/CircuitCo.

[8] Benyan Hou, Chenglong Huang, Ting Yang, Gaofei Wu, He Wang, and Yuqing Zhang. 2024. Research on State-Based Bluetooth Multi-Protocol Fuzzing. In *2024 IEEE Annual Congress on Artificial Intelligence of Things (AIoT).* IEEE, Melbourne, Australia, 13–18. https://doi.org/10.1109/AIoT63253.2024.00013

[9] innoRIID GmbH. 2019. *radEAGLE User Manua.* innoRIID GmbH. Software v3.2.12, Document v3.3.0o.

[10] Terry H. Lorier. 2014. *Validation of ANSI N42.34: American National Standard Performance Criteria for Hand-Held Instruments for the Detection and Identification of Radionuclides.* Technical Report SRNL-STI-2014-00360, Revision 0. Savannah River National Laboratory. https://sti.srs.gov/fulltext/SRNL-STI-2014-00360.pdf

[11] John Makoye and Jesu Arockia Rose Mina. 2025. Enhancing Border Security Against Nuclear and Radiological Threats. *Nuclear Science* 10, 2 (2025), 25–38. https://doi.org/10.11648/j.ns.20251002.11

[12] Luis Marques, Alberto Vale, and Pedro Vaz. 2021. State-of-the-Art Mobile Radiation Detection Systems for Different Scenarios. *Sensors* 21, 4 (2021), 1051. https://doi.org/10.3390/s21041051

[13] MITRE Corporation. 2025. *MITRE Security Taxonomies: Common Vulnerabilities and Exposures (CVE) and Common Weakness Enumeration (CWE).* https://cve.mitre.org

[14] MRIGlobal. 2025. *RADEAGLE Specifications, Cost, Training, and Deployment.* https://cbrnetechindex.com/Print/4280/ortec-ametek/radeagle

[15] AMETEK ORTEC. 2025. Vendor Documentation for the RADEAGLE and innoRIID Product Line. Consolidated Vendor Documentation and Product Descriptions. https://www.ortec-online.com/innoriid

[16] ORTEC Products Group. 2015. ORTEC Partners with innoRIID for Release of RADEAGLE Scintillation-Based Hand-Held Radioisotope Identification Device. https://news.cision.com/ametek

[17] John Padgette, Karen Scarfone, and Lily Chen. 2017. *Guide to Bluetooth Security.* Technical Report NIST Special Publication 800-121 Revision 2. National Institute of Standards and Technology. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-121r2.pdf

[18] U.S. Environmental Protection Agency. 2025. *Radionuclide Basics: Cesium-137.* https://www.epa.gov/radiation/radionuclide-basics-cesium-137

[19] Jianliang Wu, Ruoyu Wu, Dongyan Xu, Dave Jing Tian, and Antonio Bianchi. 2022. Formal Model-Driven Discovery of Bluetooth Protocol Design Vulnerabilities. In *2022 IEEE Symposium on Security and Privacy (SP).* IEEE, San Francisco, United States, 2285–2303. https://doi.org/10.1109/SP46214.2022.9833777