ORIGINAL RESEARCH, ANALYSIS, DISCLOSURES AND AUTHORED BY JON “GAINSEC” GAINES*

*Contribution from Joseph “JosephRC” Cohen

Examining the security posture of an Anti-Crime Ecosystem

Version: 1.2-PR

November 11, 2025

Table of Contents

TABLE OF CONTENTS 0

DISCLAIMER 3

ABSTRACT 4

MY BACKGROUND 5

SCOPE 6

FORMAT EXPLANATION 10

EXECUTIVE SUMMARY 11

FINDINGS OVERVIEW 13

DETAILED FINDINGS 18

DEVICE 1: RAVEN GUNSHOT DETECTION SYSTEM 18

- **FINDING 1: Secure Boot is Disabled – Raven Gunshot Detection** 18
- **FINDING 2: Debug (UART) Console Access** 19
- **FINDING 3: Lack of Password Debug (UART) Console Access** 21
- **FINDING 4: Hardcoded Wi-Fi Credentials Auto Connect – Raven Gunshot Detection** 23
- **FINDING 5: Lack of Flash Encryption - Raven** 25
- **FINDING 6: Debug Interface Accessible (JTAG) – Raven Gunshot Detection** 27
- **FINDING 7: Debug Interface Accessible (UART Download) – Raven Gunshot Detection** 28
- **FINDING 8: No Anti-Rollback Protection – Raven Gunshot Detection** 29
- **FINDING 9: Audio ML/AI Model Disclosed – Raven Gunshot Detection** 30
- **FINDING 10: Hardcoded Credentials – API Client Secret – Raven Gunshot Detection** 31
- **FINDING 11: Lack of Server Verification (DNS Spoofing) – Raven Gunshot Detection** 34

DEVICE 2: FALCON/SPARROW/FLEX* LICENSE PLATE READER (LPR) 36

- **FINDING 12: Root Shell - Falcon/Sparrow/Flex* LPR** 36
- **FINDING 13: Secure Boot is Disabled – Falcon/Sparrow/Flex* LPR** 37
- **FINDING 14: Unlocked Bootloader – Falcon/Sparrow/Flex* LPR** 39
- **FINDING 15: Lack of Authentication: EDL/QDL Mode – Falcon/Sparrow/Flex* LPR** 40
- **FINDING 16: Lack of Authentication – Android Debug Bridge - Falcon/Sparrow/Flex* LPR** 42

FINDING 17: Improper Access Control – Android Debug Bridge Sideload - Falcon/Sparrow/Flex* LPR 42
FINDING 18: Lack of Flash/EMMC Encryption - Falcon/Sparrow/Flex* LPR 43
FINDING 19: Use of an Unsupported and End of Life Operating System - Falcon/Sparrow/Flex* LPR 44
FINDING 20: Sensitive Information Disclosed – Development/Test Credential in Production – Falcon/Sparrow/Flex* LPR 45
DEVICE 3: PICARD/BRAVO COMPUTE BOX 46
FINDING 21: Root Shell – Picard/Bravo Compute Box 46
FINDING 22: Secure Boot is Disabled – Compute Box 47
FINDING 23: Unlocked Bootloader – Compute Box 48
FINDING 24: Lack of Authentication: EDL/QDL Mode – Picard/Bravo Compute Box 49
FINDING 25: Lack of Authentication – Android Debug Bridge – Compute Box 50
FINDING 26: Improper Access Control – Android Debug Bridge Sideload– Compute Box 50
FINDING 27: Lack of Flash/UFS Encryption – Compute Box 51
MUTLI-DEVICE 52
FINDING 28: Unauthenticated Administrative API Endpoints 52
FINDING 29: Hidden Hardware Debug Functionality – Hotspot 54
FINDING 30: Wireless Remote Code Execution (RCE) – System* 55
FINDING 31: Incorrect Default Permissions – Media Recordings Directories 56
FINDING 32: Shared Media Library Allows Cross App Data Exposure 57
FINDING 33: Wireless Remote Code Execution (RCE) - Shell 57
FINDING 34: Unauthenticated Debug Broadcast Clears Settings and Shuts off Device 59
FINDING 35: Multiple Privileged System Apps Shipped with Debugging Enabled 60
FINDING 36: Lack of Per File Encryption on Sensitive Media 61
FINDING 37: Sensitive Information Disclosed – Hardcoded Auth0 Secret 61
FINDING 38: Root Command Injection via Data Log Cleanup Service 63
FINDING 39: Excessive Sensitive Media Copies Persist on Disk 64
FINDING 40: Sensitive Information Disclosed – Cleartext API Keys/Credentials 64
FINDING 41: Wireless Remote Code Execution (RCE) - Root 67
FINDING 42: ML/AI Local Model Accessible 68

FINDING 43: Sensitive Information Disclosed – Hardcoded Java Keystore & Password 68
FINDING 44: Data Recording retention relies solely on Disk Capacity 71
FINDING 45: Records are stored on unencrypted external partition 71
FINDING 46: Sensitive Information Disclosed – Datadog API Token 72
PUBLIC APPLICATIONS 73
FINDING 47: Cleartext Communications 73
FINDING 48: Sensitive Information Disclosure – Google API Key 74
FINDING 49: Plaintext HTTP in Logs 74
FINDING 50: Sensitive Information Disclosure – API Keys 75
EXTERNAL CONTRIBUTOR 75
FINDING 51: Remote Code Execution (RCE) – System* 75
TIMELINE 77
CONCLUSION 78
DISTRIBUTABLE FORMAL STATEMENT 79
Disclaimer 79
Background 80
Summary of Probable Cause 80
Material Facts 80
Context of Exposure 81
Technical Detainment/Remediation 81
Declaration & Contact 81
APPENDIX A: TERM GLOSSARY 82
APPENDIX B: METHODOLOGY 83
APPENDIX C: DEFENDERS CHECKLIST 85

DISCLAIMER

This work documents good-faith security research conducted exclusively on vices legally procured and maintained in an isolated research environment under sole researcher custody, without accessing any third-party account, network, or system. All testing avoided circumvention intended to facilitate infringement, avoided trafficking in circumvention tools, and excluded any attempt to access production services. Findings are published to inform defensive remediation and public interest oversight. Nothing herein is an instruction to access systems "without authorization" or to exceed authorized access within the meaning of the Computer Fraud and Abuse Act (18 U.S.C. §1030), nor to engage in circumvention prohibited by 17 U.S.C. §1201 outside applicable exemptions for good-faith security research. The content is intended to inform defenders and vendors; it is not an instruction manual for exploitation. Attempting to reproduce the described tests on devices you do not own, on live/production systems, or without explicit permission may be unlawful, dangerous, and cause unintended harm.

The author affirms that this research was performed independently, without financial support, employment, consultancy, or material benefit from the vendor or its affiliates. No funding, compensation, or third-party direction influenced the selection of targets, the methods used, or the interpretation of results. The devices analyzed were purchased at retail by the author. Tests were confined to offline/lab environments; no interception of third-party communications or content prohibited by ECPA/Title III occurred; no human-subjects' data were collected.

The purpose of this study is to advance public understanding of security posture and responsible disclosure practices, not to promote or discredit any product or company. This document is its current form is intended for defensive security, compliance, and policy evaluation. Redistribution that adds operational detail, live credentials, or working exploits is prohibited.

Descriptions intentionally omit operational steps and live secrets. Any reproduction must be limited to assets owned or explicitly authorized for testing, conducted in controlled lab environments, and performed solely to validate remediation. These methods must not be applied to third parties or production systems. All literal command strings, payloads, and other sensitive technical details have been redacted from this white paper. High level reproduction steps remain included to enable independent validation and further research without enabling misuse. For findings pending full disclosure, limited additional redactions have been applied; these do not affect comprehension or validation of the underlying issues. Parties with a legitimate need for the unredacted material may contact the author directly.

ABSTRACT

The prevalence of anti-crime technology has seen a steep incline in the past few years. Since the introduction of cell phones, the expectation of privacy has gone steeply down. With that in mind, independent security research was conducted into these devices and their underlying technologies. The types of surveillance devices accessed are the following: Gunshot Detection Systems, License Plate Reader (LPRs) and the AI Compute Boxes that support these technologies. This research purposely focuses on one ecosystem, multiple devices from the same vendor, rather than cross-vendor comparison. The selected hardware is actively deployed within the researcher's locality and is obtainable on a modest independent research budget via third-party marketplaces. These devices remain largely unexplored despite their rapid deployment across the United States.

Although the initial focus was on physical access-based findings, the scope quickly expanded. The objective of this research is to raise awareness, strengthen the overall security posture and ensure that technologies used by law enforcement are as secure and resilient as possible, supporting their mission while minimizing risk to the public or national security. This paper details fifty-one (51) findings independently discovered and disclosed, as well as one additional finding discovered and disclosed by a colleague. Whether through hardware interfaces, debug functionality, bootloader misconfigurations, EOS services, or the custom Android application suite included on two of the three types of evaluated devices, a comprehensive list of security issues is included. While additional research remains to be done, this project has reached a stage suitable for formal publication documenting the current state of the independent research into Flock Safety's (The Vendor) anti-crime ecosystem.

My Background

The author is a seasoned offensive security practitioner, leader, and researcher with over a decade of experience as a security consultant across several security firms. Within those organizations, hundreds of offensive security engagements have been performed against entities ranging from start-ups to government entities, from most of the Fortune 10 to non-profits across sectors including finance, software development, and beyond.

These engagements have encompassed a broad range of offensive security assessments, with a primary focus on penetration testing across nearly every major service line. Including but not limited to, red team operations, Internet of Things (IoT), Web Application, Software, Physical, Cloud, Host Based, Internal, External, and other more specialized engagements. This extensive experience has provided a comprehensive understanding of the security posture across a wide range of organizations. It has also fostered deep insight into both offensive and defensive practices, ensuring that this research is grounded in practical, accurate and balanced perspective. Contributions have included service line development, introduction of new offensive cybersecurity offerings, mentorship (both formal and informal), methodology creation, and the design and delivery of training programs. Leadership experience includes managing teams of up to fifteen offensive cybersecurity professionals, guiding both daily execution and long-term career growth.

Beyond professional work, a lifelong interest in exploration, testing, and hacking has shaped much of this career. Over five years ago, this independent research identity was formalized under the handle *GainSec*, which has evolved into a personal brand. Under this handle, the author has lectured at conferences, academic institutions, and industry events, published work in *Phrack* and *Unredacted*, developed and contributed to open-source tooling, mentored both junior and senior peers, volunteered at community security events, and achieved 48 of the 50 currently published CVEs attributed to this research, all disclosed responsibly.

SCOPE

Out of Scope:

1. Any live production deployed devices operating in the wild.
2. Any externally facing, cloud-hosted or third-party service the devices communicate or connect to.
3. Vulnerabilities within the base teal time operating system (RTOS) or embedded operating system (EOS itself, or the manufacturer applications unrelated to the device security operation.

In Scope:

1. Devices acquired legally through third party markets, tested exclusively in a laboratory environment.
2. Any firmware, RTOS, EOS, applications, software, or services operating within the vendor's device ecosystems.
3. Any publicly available mobile application associated with these devices.

Limitations:

1. Devices deployed in production environment may differ in configuration, firmware updates, or operational policy.
2. The vendor has not confirmed whether any of the reported issues have been remediated or are planned for remediation.
3. Certain functionality, including registration functionality and authentication validation could not be tested due to the lack of backend access and vendor confirmation. The exception being Finding 35's API key which the vendor stated is no longer valid.
4. Most of the findings have been already completed through responsible full disclosure, however there are a few that have not reached their full disclosure dates at the time of writing.
5. Communications with the vendor have since degraded due to issues described in the *Timeline* section. Despite this, all disclosure timelines and full disclosure technical write-ups continue to be provided to the vendor prior to publication.

Devices Covered

Device Name – Firmware Hash	Device Type	OS Type	Model #	# of Units
Raven - 8bcdd2fd8042ba91af2e94db044f301a293936980821a23564a85dfae41a7b12	Gunshot Detection	RTOS - ESP	1.2	1
Falcon/Sparrow/Flex* - 08da4991581076e2d0b3be87c377c177d955d55c92be8ecee66e586181293a2f	License Plate Reader	EOS – Android Things	2.2	3
Picard/Bravo Compute Box- dede8a4976eee00e464f6e7c301b291954e7941951fdcf23642613912a94bca7	Edge AI Compute Box	EOS - Android	1.0	2

Multi-Device Applications (Vendors on Device Application Suite)

Application Name	Package Name - Hash	OS	Version #1
Phone Home Service	com.flocksafety.android.phonehomeservice - ccf6fd6e53f49a13ccc623fde766769a00d7f83491c5caf7d836fb0dc0199d97	Android	6.35.35
System Control	com.flocksafety.android.systemcontrol - 54316c1cc5ead339f4561d2de0de059b2b449e5ff20c1e6f533cd4c212ea35e4	Android	6.35.35
Objects(DetectionProcessing)	com.flocksafety.android.objects - 9a737222514143fb03ed8464098913c1857ee76fc2df5ba90fe239d661f69e62	Android	6.35.33
Pisco	com.flocksafety.android.pisco - d046aa8a4d94208b4b133ffec064e8884ac4b9682e6b47f403415113a519bb9e	Android	66.21.11
Peripheral	com.flocksafety.android.peripheral - 2564cc12b4691bf8970cd4bf927f9755361c4ee9454b8ffb86c7c4b0a6de1d0e	Android	6.35.30
Collins	com.flocksafety.android.collins - e015c2934e92564c1855a8f9a61d8986daac83d78e7f87d897d2147d677b9d36	Android	6.35.31
Video Recording	com.flocksafety.android.videorecording - f2f1844cf6410c523b6e7a4f5d98ea693b3f6200c1afcef628c950ce78f525b5	Android	7.38.3

Motion	com.flocksafety.android.motion - 480b6234e1f89b19b97e03c8ef879688a1b6997e050f3343a0bc1c1a289f4ab7	Android	7.38.3
Encoding	com.flocksafety.android.encoding - b64392518e286844eae8c65403e0b2574f9fad36df08d8388b82c7d871b44636	Android	7.38.3
Camera Config	com.flocksafety.android.cameraconfig - 769ef6ce5e7c300384d053c54df0b5a9c17a25fa97396cebfa1a416d501f0268	Android	7.38.5
Video Streaming	com.flocksafety.android.streaming - 5d627be4363c3ce6db1475cf57699f1ddcbfedc1ea75e3f1cb21c7c65563cacd	Android	7.38.3
Application Name	**Package Name - Hash**	**OS**	**Version #2**
Phone Home Service	com.flocksafety.android.phonehomeservice - 33210c78a29c82ccb4c91fb32acb1dc30cd157eb4c1485c23658110a2c6aaf6c	Android	7.38.5
System Control	com.flocksafety.android.systemcontrol - ac3d9d05b5c278bf56086dba0f954c9994bdea8339b831c54bb576e39b571e89	Android	7.38.3
Objects(DetectionProcessing)	com.flocksafety.android.objects - e4e34bf3b7d15f642fe070be52fb19bc545bd3284d4066c3e31eda15a8e0e69c	Android	7.38.3
Peripheral	com.flocksafety.android.peripheral - 9d16c033ce58e9787e3db4c8815ce4050cf943200e84036ff098ec62083aebd4	Android	7.38.3
Collins	com.flocksafety.android.collins - dc10cb9b9da76adfde20e196bc1fa96e6c3c35e81eb60a5ff4b43bfbf68e6c36	Android	7.38.3

Standalone Applications Covered

Application Name	Package Name - Hash	OS	Version #
FSInstaller	com.flocksafety.hazyhiwire - b46ea409d43529de8320ab0dfcc69d27d1040381090d05009c00d5d865a1cda8	Android	2.4.0
Flock Safety	com.flocksafety.sweetwater - 3703c043dfdbd98ad851d91252fcd844364fad1620ad84d81832bbe5d32048a2	Android	1.49.1
Flock On Patrol	com.flocksafety.android.negroni - b54f7a53250f2162e99aae4f09f7ec9d69f581221e3deaaae7f8e97d2a4c8b99	Android	1.2.0

FORMAT EXPLANATION

The following sections have been created in the format I'm most familiar with, a penetration test. With that in mind, it is important to break down what the formatting is for the rest of this paper both to ensure that it is easily digestible and understood.

Executive Summary:

Provides a high-level overview of the most impactful and notable findings, indicators, and exposures identified during research. This section serves as a non-technical explanation of the assessed technologies' overall security posture. It connects technical issues to their underlying categories and explains their significance.

Findings Overview:

Provides a high-level view into all identified issues, divided into six categories: three device specific sections, one section for multi-device findings, one for publicly available mobile applications (distributed through the app store) and one for external contributor findings. This section includes relevant figures and a consolidated table containing key issue data.

Detailed Findings:

Contains an in-depth analysis of each issue, formatted in a classical offensive security style. At the time of writing, some issues remain under responsible full disclosure embargo or have not yet been submitted for formal CVE assignment. Most have completed the responsible full disclosure lifecycle and have self-published full disclosure technical write-ups on the author's site.

Timeline:

Outlines the project chronology, including key milestones, disclosure events and document version information.

Conclusion:

Provides an overall assessment of the current research state and states areas for future research.

Distributable Formal Statement:

A condensed version of this paper that was prepared for public distribution, intended for journalists, privacy advocates, non-profits, regulators, legal professionals, pro-law enforcement entities and legislators. Condensation may utilize AI assisted summarization techniques.

Appendixes: Term Glossary, Methodology, and a Defenders Checklist.

EXECUTIVE SUMMARY

Version 1.2-PR of this paper documents fifty-one (51) findings. Of these, twenty-two (22) have been assigned CVE identifiers and have been responsibly and fully disclosed. Eight (8) findings are pending CVE assignment and full disclosure, and at least four (4) additional findings meet the standard for CVE eligibility but have not yet been submitted.

Elevent (11) findings were identified in the Raven Gunshot Detection System, with seven (7) tied to CVEs. Nine (9) findings were identified in the Falcon/Sparrow/Flex* License Plate Reader, with five (5) assigned CVEs. Seven (7) findings were identified in the Picard/Bravo Compute Box, with four (4) assigned CVEs. Nineteen (19) findings applied across multiple devices, of which fifteen (15) are tied to, or pending, unique CVE assignments. Four (4) findings relate to the vendor's public mobile applications, one (1) of which is pending CVE assignment. One (1) additional finding was contributed by an external researcher.

This research represents the first known comprehensive public vulnerability assessment of these devices and their ecosystem. Prior to this work, no public disclosures or advisories existed for the hardware or software evaluated. The largest concentrations of issues fall within the categories of Cryptographic Failures, Improper Access Control, and Sensitive Information Disclosure, which together account for twenty-five (25) of the total findings.

Cryptographic failures expose a systemic absence of fundamental protections, including secure boot, flash/EMMC/UFS encryption, and key management. This leaves firmware, stored data, and communications susceptible to unauthorized disclosure, modification, replacement, and integrity bypass, ultimately enabling full compromise of the device trust chain.

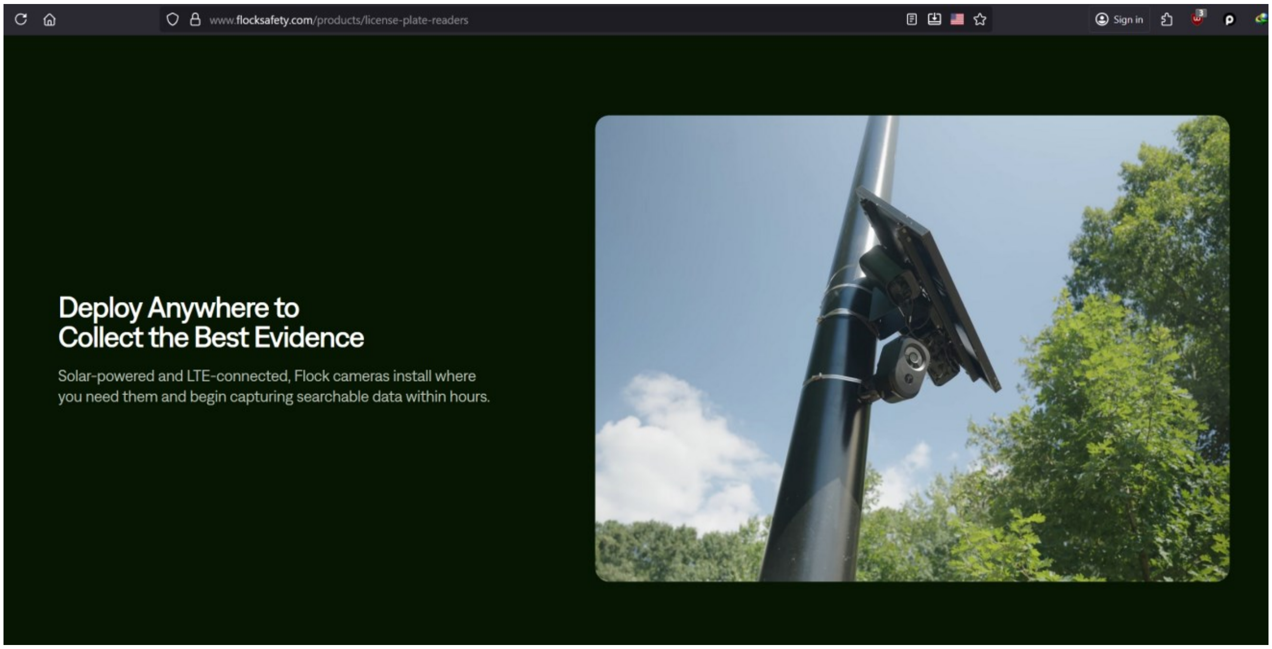
Improper access control findings reveal insufficient hardening of critical interfaces such as UART, ADB, API, and EDL. These weaknesses allow attackers to bypass privilege checks, gain system- or administrative-level access, and manipulate restricted functions or data.

Some device misconfigurations resulted in unintentional security side effects. For instance, the SELinux policy prevents the data log cleanup service from executing, which incidentally blocks an attacker from obtaining a root shell over Wi-Fi or USB. However, it remains unclear whether this policy is active in production-deployed units; no conclusion can be made as to which configuration provides greater protection.

The License Plate Reader units were also found to run Android Things 8.1, a platform long past end-of-life and unsupported by the vendor or Google. In a conventional penetration test, such findings would be classified as high or critical severity due to the inherent risk of unpatched vulnerabilities within the base OS and bundled applications.

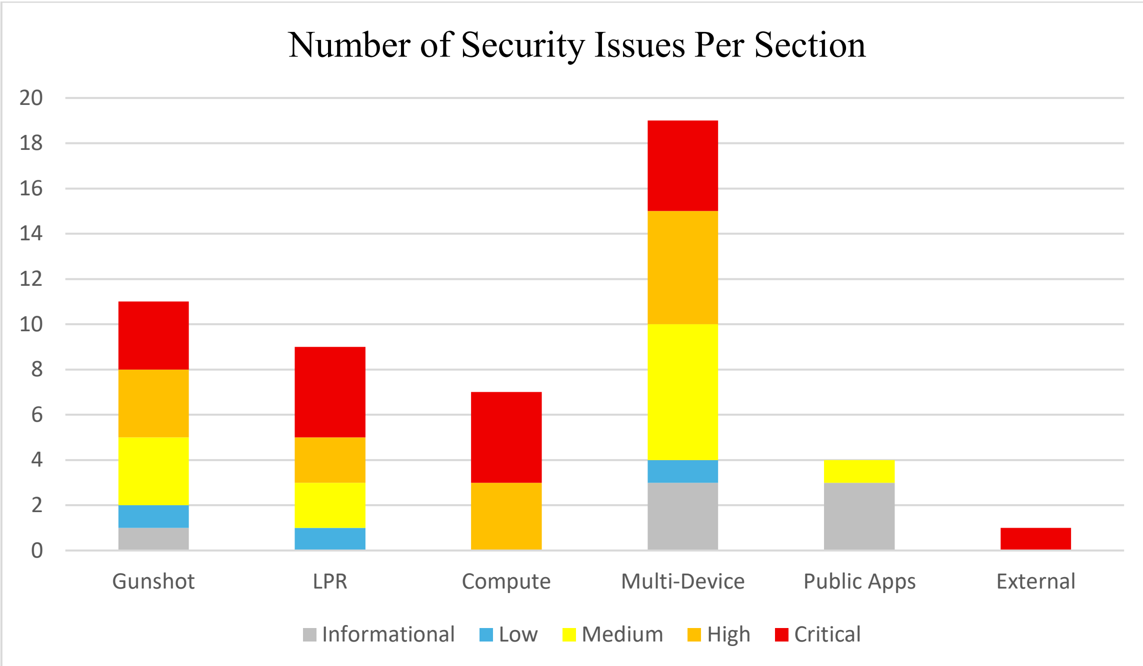
Multiple independent paths to root-level access were confirmed. This is especially concerning given these devices are deployed by law enforcement, positioned in public areas (often roadside on 5−7 ft poles), and physically accessible. In certain models, attackers could obtain a wireless shell via an exposed hardware button or exploit identical weak hotspot credentials. Others, such as the Picard/Bravo Compute Box, could be compromised directly through physical USB-C access and subsequent privilege escalation.

The primary recommendations are to remediate all disclosed vulnerabilities, implement formal patch management and asset tracking across deployed hardware, and enforce stronger operational policy to prevent recurrence of these systemic weaknesses.



An image from the Vendors website as off 11/01/25 showing how the SIM tray, button, and ports of the compute box are all exposed if the Picard/Bravo units are deployed in the wild

FINDINGS OVERVIEW



DEVICE 1: RAVEN GUNSHOT DETECTION

#	Title	Severity	Responsible Disclosure Date	CVE #	Vendor Acknowledged (Y/N)	Remediated (Y/N/U)
1	Secure Boot is Disabled	CRITICAL	6/19/2025	CVE-2025-47819	Y	U
2	Debug (UART) Console Access	CRITICAL	6/19/2025	CVE-2025-47819	Y	U
3	Lack of Password Debug (UART) Console Access	CRITICAL	6/19/2025	CVE-2025-47819	Y	U
4	Hardcoded Wi-Fi Credentials Auto Connect	HIGH	6/19/2025	CVE-2025-47818	Y	U
5	Lack of Flash Encryption	HIGH	6/19/2025	CVE-2025-47820	Y	U
6	Debug Interface Accessible (JTAG)	HIGH	6/19/2025	CVE-2025-47819	Y	U
7	Debug Interface Accessible (UART Download)	MEDIUM	6/19/2025	CVE-2025-47819	Y	U
8	No Anti-Rollback Protection	MEDIUM	6/19/2025	N/A	Y	U

9	Audio ML/AI Model Disclosed	MEDIUM	6/19/2025	N/A	Y	U
10	Hardcoded Credentials – API Client Secret	LOW	6/19/2025	CVE-2025-47821	Y	U
11	Lack of Server Verification (DNS Spoofing)	INFORMATIONAL	6/19/2025	N/A	Y	U

DEVICE 2: FALCON/SPARROW/FLEX* LPR

#	Title	Severity	Responsible Disclosure Date	CVE #	Vendor Acknowledged	Remediated (Y/N/U)
12	Root Shell	CRITICAL	6/19/2025	N/A	Y	U
13	Secure Boot is Disabled	CRITICAL	6/19/2025	CVE-2025-47822	Y	U
14	Unlocked Bootloader	CRITICAL	6/19/2025	CVE-2025-47822	Y	U
15	Lack of Authentication: EDL/QDL Mode	CRITICAL	6/19/2025	CVE-2025-47822	Y	U
16	Lack of Authentication – Android Debug Bridge	HIGH	6/19/2025	N/A	Y	U
17	Improper Access Control – Android Debug Bridge Sideload	HIGH	6/19/2025	N/A	Y	U
18	Lack of Flash/EMMC Encryption	MEDIUM	6/19/2025	CVE-2025-47824	Y	U
19	Use of an Unsupported and End of Life Operating System	MEDIUM	6/19/2025	N/A	Y	U
20	Sensitive Information Disclosed – Development/Test Credential in Production	LOW	01/23/26	CVE-2025-59409	Y	U

DEVICE 3: PICARD/BRAVO COMPUTE BOX

#	Title	Severity	Responsible Disclosure Date	CVE #	Vendor Acknowledged	Remediated (Y/N/U)
21	Root Shell	CRITICAL	9/19/2025	N/A	Y	U
22	Secure Boot is Disabled	CRITICAL	9/19/2025	CVE-2025-59408	Y	U
23	Unlocked Bootloader	CRITICAL	9/19/2025	CVE-2025-59404	Y	U
24	Lack of Authentication: EDL/QDL Mode	CRITICAL	9/19/2025	CVE-2025-59402	Y	U

25	Lack of Authentication – Android Debug Bridge	HIGH	9/19/2025	N/A	Y	U
26	Improper Access Control – Android Debug Bridge Sideload	HIGH	9/19/2025	N/A	Y	U
27	Lack of Flash/UFS Encryption	HIGH	9/19/2025	N/A	Y	U

MULTI-DEVICE

#	Title	Severity	Responsible Disclosure Date	CVE #	Vendor Acknowledged	Remediated (Y/N/U)
28	Unauthenticated Administrative API Endpoints	CRITICAL	9/27/2025	CVE-2025-59403	Y	U
29	Hidden Hardware Debug Functionality – Hotspot	CRITICAL	9/27/2025	PENDING	Y	U
30	Wireless Remote Code Execution (RCE) – System	CRITICAL	01/23/2026	PENDING	Y	U
31	Incorrect Default Permissions – Media Recordings Directories	CRTICAL	02/22/2026	PENDING	N	U
32	Wireless Remote Code Execution (RCE) - Shell	HIGH	9/27/2025	CVE-2025-59403	Y	U
33	Shared Media Library Allows Cross App Data Exposure	HIGH	02/22/2026	PENDING	N	U
34	Unauthenticated Debug Broadcast Clears Settings and Shuts off Device	HIGH	01/23/2026	PENDING	Y	U
35	Multiple Privileged System Apps Shipped with Debugging Enabled	HIGH	9/27/2025	PENDING	Y	U
36	Lack of Per File Encryption on Sensitive Media	HIGH	02/22/2026	PENDING	N	U
37	Sensitive Information Disclosed – Hardcoded Auth0 Secret	MEDIUM	9/27/2025	CVE-2025-59406	Y	U
38	Root Command Injection via Data Log Cleanup Service	MEDIUM	01/23/2026	PENDING	Y	U
39	Excessive Sensitive Media Copies Persist on Disk	MEDIUM	02/22/2026	PENDING	N	U
40	Sensitive Information Disclosed – Cleartext API Keys/Credentials	MEDIUM	6/19/2025	CVE-2025-47823	Y	U

41	Wireless Remote Code Execution (RCE) - Root	MEDIUM	01/23/2026	PENDING	Y	U
42	ML/AI Local Model Accessible	MEDIUM	6/19/2025	N/A	Y	U
43	Sensitive Information Disclosed – Hardcoded Java Keystore & Password	LOW	9/27/2025	CVE-2025-59407	Y	U
44	Data Recording retention relies solely on Disk Capacity	INFORMATIONAL	N/A	N/A	Y	U
45	Records are stored on unencrypted external partition	INFORMATIONAL	N/A	N/A	Y	U
46	Sensitive Information Disclosed – Datadog API Token	INFORMATIONAL	09/27/2025	CVE-2025-59405	N	U

Public applications

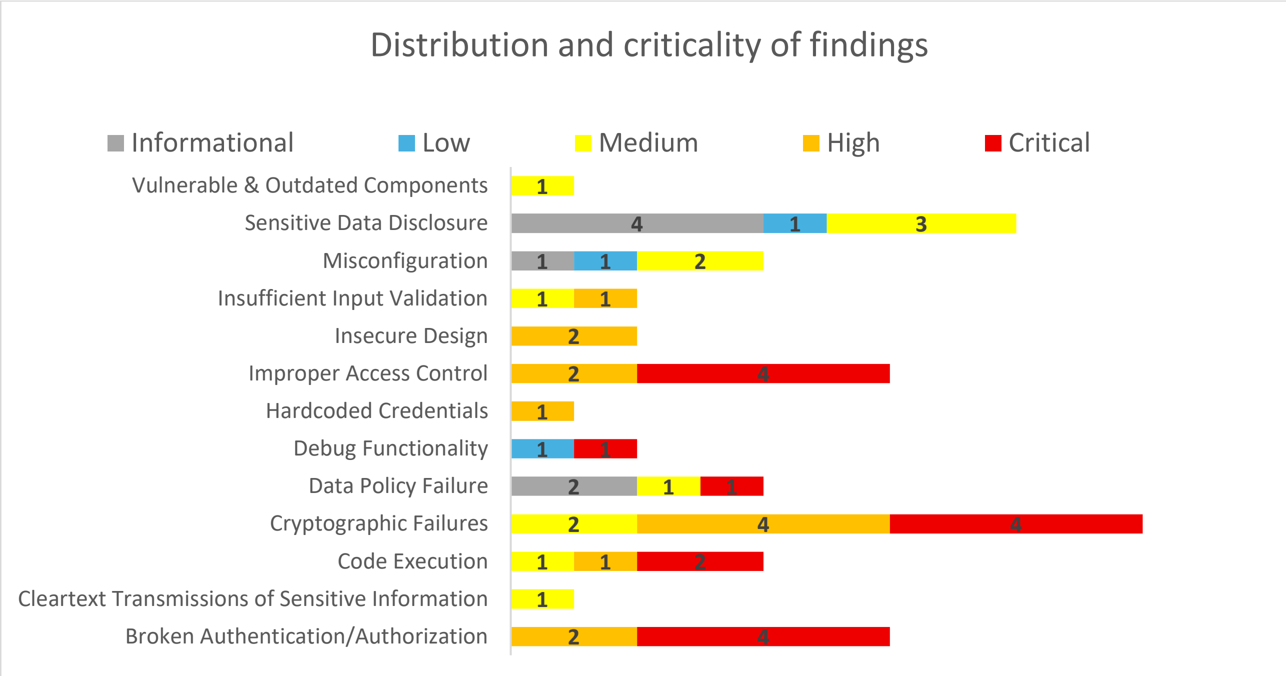
#	Title	Severity	Responsible Disclosure Date	CVE #	Vendor Acknowledged	Remediated (Y/N/U)
47	Cleartext Communications	MEDIUM	N/A	N/A	N/A	U
48	Sensitive Information Disclosure – Google API Key	INFORMATIONAL	N/A	N/A	N/A	U
49	Plaintext HTTP in Logs	INFORMATIONAL	N/A	N/A	N/A	U
50	Sensitive Information Disclosure – API Keys	INFORMATIONAL	N/A	N/A	N/A	U

External contributions

#	Title	Severity	Responsible Disclosure Date	CVE #	Vendor Acknowledged	Remediated (Y/N/U)
51	Remote Code Execution (RCE) – System	CRITICAL	01/23/2026	PENDING	Y	U

Certain sensitive technical details have been redacted for security and disclosure reasons; refer to the *Disclaimer* section for full context.

DETAILED FINDINGS



DEVICE 1: RAVEN GUNSHOT DETECTION SYSTEM

FINDING 1: Secure Boot is Disabled – Raven Gunshot Detection

<table>
<tr><th colspan="3">Description</th></tr>
<tr><td>Type: Cryptographic Failures</td><td rowspan="2">The Raven gunshot detection system was found to have ‘Secure Boot’ disabled. Secure Boot is a security feature that ensures only trusted software runs during the devices startup process.</td><td>CVSS Score: 9.8</td></tr>
<tr><td>Threat Context: Inexperienced Attacker</td><td>Severity: CRTICAL</td></tr>
<tr><td rowspan="2">Public Full Disclosure Date: 06/19/2025</td><td>Impact</td><td rowspan="2">CVE #: CVE-2025-47819</td></tr>
<tr><td>Disabling Secure Boot allows unsigned or malicious bootloaders and kernel-level code to execute during system startup, undermining the trust chain and enabling persistent compromise at the firmware or OS level. This exposes the host to rootkits and pre-boot tampering undetectable by standard security controls.</td></tr>
<tr><td colspan="2">CVSS 4.0 AV:P/AC:L/AT:N/PR:N/UI:N/VC:H/VI:H/VA:H/SC:N/SI:N/SA:N</td><td>CWE: CWE- 1326</td></tr>
<tr><td rowspan="2">Discovered By:
Jon Gaines</td><td>Affected Hardware/Software</td><td rowspan="2">Further Research Recommended: N</td></tr>
<tr><td>Raven Gunshot Detection</td></tr>
<tr><td colspan="3">Notes
This finding was improperly included in CVE-2025-47819 instead of being given its own CVE number when the Vendor submitted the CVE assignment request.</td></tr>
<tr><td colspan="3">Relevant Output:
BLOCK2 (secure_boot_v1 s) [2] read_regs:
00000000 00000000 00000000 00000000</td></tr>
</table>

00000000 00000000 00000000 00000000 BLOCK2 (BLOCK2) Security boot key= 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0000 R/W

Steps to Reproduce:
1. Open the case of the device.
2. Connect probes at the UART pad on the devices mainboard.
3. Connect a TTL/UART adapter to the probes.
4. Use ‘esptool’ to dump the ‘espfuse’ settings and values: <REDACTED_COMMAND>
5. Note the ‘BLOCK2’ value, showing that flash encryption is disabled:
6. Additionally, the following command can also confirm the UART Download Mode Support is enabled: <REDACTED_COMMAND>
7. Note the values of ‘ABS_DONE_0’ and ‘ABS_DONE_1’ showing that secure boot v1 and secure boot v2 is disabled:
   ABS_DONE_0 (BLOCK0) Secure boot V1 is enabled for bootloader image = False R/W (0b0)
   ABS_DONE_1 (BLOCK0) Secure boot V2 is enabled for bootloader image = False R/W (0b0)

Tools Used:
o Esptool
o UART Adapater
o strings

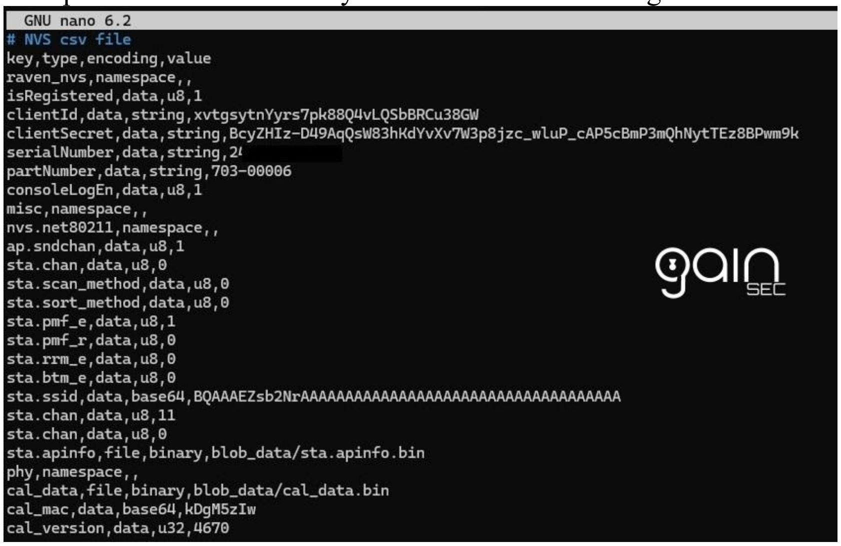
Mitigation
Enable Secure Boot.

FINDING 2: Debug (UART) Console Access

<table>
<tr><th colspan="3">Description</th></tr>
<tr><td>Type: Improper Access Control</td><td rowspan="2">The Raven gunshot detection system was found to have debug (UART) console access disabled. However, it can be reenabled via a single byte modification of its NVS’ partition. This results in control of the device via a ‘shell.’</td><td>CVSS Score: 8.7</td></tr>
<tr><td>Threat Context: Inexperienced Attacker</td><td>Severity: CRTICAL</td></tr>
<tr><td rowspan="2">Public Full Disclosure Date: 06/19/2025</td><td>Impact</td><td rowspan="2">CVE #: CVE-2025-47819</td></tr>
<tr><td>An attacker can leverage this access to run debug commands, view firmware logs and other functionalities</td></tr>
<tr><td colspan="2">CVSS 4.0 AV:P/AC:L/AT:N/PR:N/UI:N/VC:H/VI:H/VA:H/SC:N/SI:N/SA:N</td><td>CWE: CWE- 1191</td></tr>
<tr><td rowspan="2">Discovered By:
Jon Gaines</td><td>Affected Hardware/Software</td><td rowspan="2">Further Research Recommended: N</td></tr>
<tr><td>Raven Gunshot Detection</td></tr>
<tr><td colspan="3">Notes
This finding was improperly included in CVE-2025-47819 instead of being given its own CVE number when the Vendor submitted the CVE assignment request.</td></tr>
<tr><td colspan="3">Relevant Output:
CONSOLE_DEBUG_DISABLE (BLOCK0) Disable ROM BASIC interpreter fallback = True R/W (0b1)</td></tr>
</table>

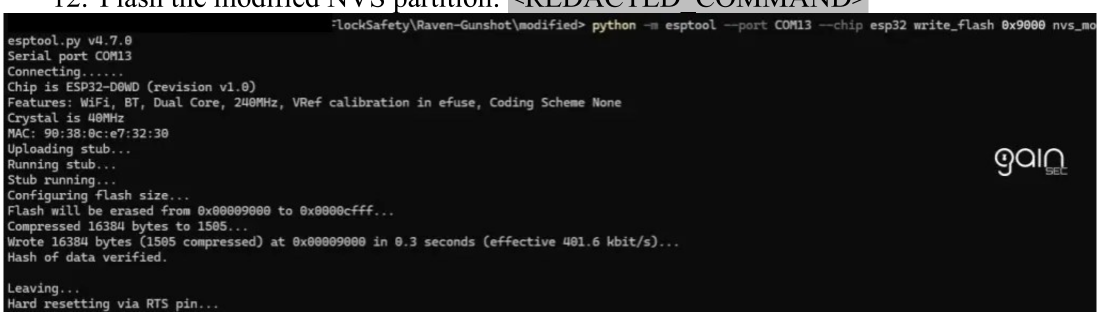
Steps to Reproduce:
1. Open the case of the device.
2. Connect probes at the UART pad on the devices mainboard.
3. Connect a TTL/UART adapter to the probes.

4. Note that the ‘CONSOLE_DEBUG_DISABLE’ is set to true when viewing the ‘ESPEfuse’ summary: <REDACTED_COMMAND>
5. Note the output: CONSOLE_DEBUG_DISABLE (BLOCK0) Disable ROM BASIC interpreter fallback = True R/W (0b1)
6. Hold IO0 and EN pads down
7. Turn it on and let EN float.
8. Dump the NVS partition specifically: <REDACTED_COMMAND>
9. Convert the ‘NVS’ dump to a CSV file: <REDACTED_COMMAND>
10. Open the CSV and modify the value of ‘ConsoleLogEn’ from 0 to 1:



```
  GNU nano 6.2
# NVS csv file
key,type,encoding,value
raven_nvs,namespace,,
isRegistered,data,u8,1
clientId,data,string,xvtgsytnYyrs7pk88Q4vLQSbBRCu38GW
clientSecret,data,string,BcyZHIz-D49AqQsW83hKdYvXv7W3p8jzc_wluP_cAP5cBmP3mQhNytTEz8BPwm9k
serialNumber,data,string,[redacted]
partNumber,data,string,703-00006
consoleLogEn,data,u8,1
misc,namespace,,
nvs.net80211,namespace,,
ap.sndchan,data,u8,1
sta.chan,data,u8,0
sta.scan_method,data,u8,0
sta.sort_method,data,u8,0
sta.pmf_e,data,u8,1
sta.pmf_r,data,u8,0
sta.rrm_e,data,u8,0
sta.btm_e,data,u8,0
sta.ssid,data,base64,BQAAAEZsb2NrAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
sta.chan,data,u8,11
sta.chan,data,u8,0
sta.apinfo,file,binary,blob_data/sta.apinfo.bin
phy,namespace,,
cal_data,file,binary,blob_data/cal_data.bin
cal_mac,data,base64,kDgM5zIw
cal_version,data,u32,4670
```

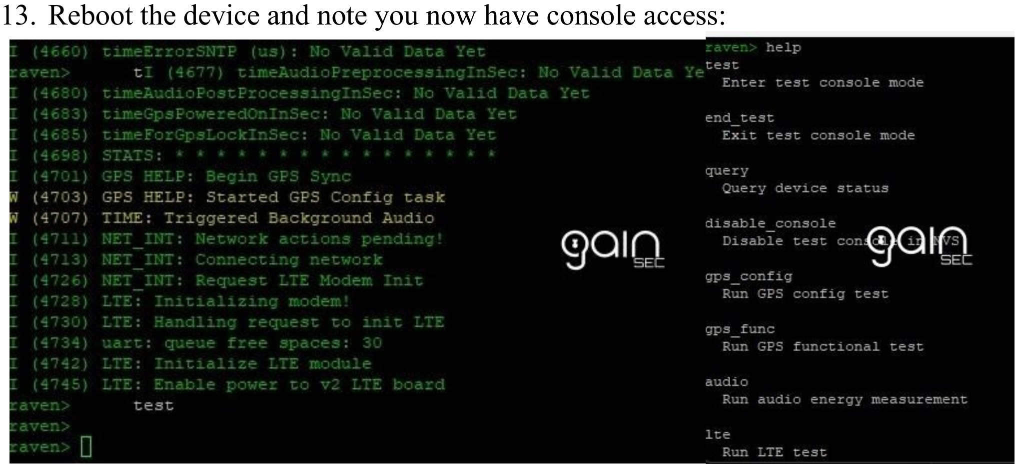
11. Convert the modified CSV to NSV format: <REDACTED_COMMAND>
12. Flash the modified NVS partition: <REDACTED_COMMAND>



```
[redacted]lockSafety\Raven-Gunshot\modified> python -m esptool --port COM13 --chip esp32 write_flash 0x9000 nvs_mo
esptool.py v4.7.0
Serial port COM13
Connecting......
Chip is ESP32-D0WD (revision v1.0)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: 90:38:0c:e7:32:30
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Flash will be erased from 0x00009000 to 0x0000cfff...
Compressed 16384 bytes to 1505...
Wrote 16384 bytes (1505 compressed) at 0x00009000 in 0.3 seconds (effective 401.6 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

13. Reboot the device and note you now have console access:



```
I (4660) timeErrorSNTP (us): No Valid Data Yet
raven>      tI (4677) timeAudioPreprocessingInSec: No Valid Data Ye
I (4680) timeAudioPostProcessingInSec: No Valid Data Yet
I (4683) timeGpsPoweredOnInSec: No Valid Data Yet
I (4685) timeForGpsLockInSec: No Valid Data Yet
I (4698) STATS: * * * * * * * * * * * * * * * *
I (4701) GPS HELP: Begin GPS Sync
W (4703) GPS HELP: Started GPS Config task
W (4707) TIME: Triggered Background Audio
I (4711) NET_INT: Network actions pending!
I (4713) NET_INT: Connecting network
I (4726) NET_INT: Request LTE Modem Init
I (4728) LTE: Initializing modem!
I (4730) LTE: Handling request to init LTE
I (4734) uart: queue free spaces: 30
I (4742) LTE: Initialize LTE module
I (4745) LTE: Enable power to v2 LTE board
raven>      test
raven>
raven>
```

```
raven> help
test
  Enter test console mode

end_test
  Exit test console mode

query
  Query device status

disable_console
  Disable test console in NVS

gps_config
  Run GPS config test

gps_func
  Run GPS functional test

audio
  Run audio energy measurement

lte
  Run LTE test
```

Tools Used:
- Esptool
- Esp32knife
- UART Adapater

Mitigation:
Do not allow UART Console Access when the device is being deployed. Encrypt the firmware.

FINDING 3: Lack of Password Debug (UART) Console Access

<table>
<tr><th colspan="3">Description</th></tr>
<tr><td>Type: Improper Access Control</td><td rowspan="2">The Raven gunshot detection system was found to lack a debug (UART) console access password.</td><td>CVSS Score: 8.7</td></tr>
<tr><td>Threat Context: Inexperienced Attacker</td><td>Severity: CRTICAL</td></tr>
<tr><td rowspan="2">Public Full Disclosure Date: 06/19/2025</td><td>Impact</td><td rowspan="2">CVE #: CVE-2025-47819</td></tr>
<tr><td>An attacker can leverage this access to run debug commands, view firmware logs and other functionalities.</td></tr>
<tr><td colspan="2">CVSS 4.0 AV:P/AC:L/AT:N/PR:N/UI:N/VC:H/VI:H/VA:H/SC:N/SI:N/SA:N</td><td>CWE: CWE- 1191</td></tr>
<tr><td rowspan="2">Discovered By:
Jon Gaines</td><td>Affected Hardware/Software</td><td rowspan="2">Further Research Recommended: N</td></tr>
<tr><td>Raven Gunshot Detection</td></tr>
<tr><td colspan="3">Notes
This finding was improperly included in CVE-2025-47819 instead of being given its own CVE number when the Vendor submitted the CVE assignment request.</td></tr>
<tr><td colspan="3">Relevant Output:</td></tr>
</table>

Steps to Reproduce:
1. Open the case of the device.
2. Connect probes at the UART pad on the devices mainboard.
3. Connect a TTL/UART adapter to the probes.
4. Note that the 'CONSOLE_DEBUG_DISABLE' is set to true when viewing the 'ESPEfuse' summary: <REDACTED_COMMAND>
5. Note the output: CONSOLE_DEBUG_DISABLE (BLOCK0) Disable ROM BASIC interpreter fallback = True R/W (0b1)
6. Hold IO0 and EN pads down
7. Turn it on and let EN float.
8. Dump the NVS partition specifically: <REDACTED_COMMAND>
9. Convert the 'NVS' dump to a CSV file: <REDACTED_COMMAND>
10. Open the CSV and modify the value of 'ConsoleLogEn' from 0 to 1:

```
  GNU nano 6.2
# NVS csv file
key,type,encoding,value
raven_nvs,namespace,,
isRegistered,data,u8,1
clientId,data,string,xvtgsytnYyrs7pk88Q4vLQSbBRCu38GW
clientSecret,data,string,BcyZHIz-D49AqQsW83hKdYvXv7W3p8jzc_wluP_cAP5cBmP3mQhNytTEz8BPwm9k
serialNumber,data,string,2[redacted]
partNumber,data,string,703-00006
consoleLogEn,data,u8,1
misc,namespace,,
nvs.net80211,namespace,,
ap.sndchan,data,u8,1
sta.chan,data,u8,0
sta.scan_method,data,u8,0
sta.sort_method,data,u8,0
sta.pmf_e,data,u8,1
sta.pmf_r,data,u8,0
sta.rrm_e,data,u8,0
sta.btm_e,data,u8,0
sta.ssid,data,base64,BQAAAEZsb2NrAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
sta.chan,data,u8,11
sta.chan,data,u8,0
sta.apinfo,file,binary,blob_data/sta.apinfo.bin
phy,namespace,,
cal_data,file,binary,blob_data/cal_data.bin
cal_mac,data,base64,kDgM5zIw
cal_version,data,u32,4670
```

11. Convert the modified CSV to NSV format: <REDACTED_COMMAND>
12. Flash the modified NVS partition: <REDACTED_COMMAND>

```
[redacted]FlockSafety\Raven-Gunshot\modified> python -m esptool --port COM13 --chip esp32 write_flash 0x9000 nvs_mo
esptool.py v4.7.0
Serial port COM13
Connecting......
Chip is ESP32-D0WD (revision v1.0)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: 90:38:0c:e7:32:30
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Flash will be erased from 0x00009000 to 0x0000cfff...
Compressed 16384 bytes to 1505...
Wrote 16384 bytes (1505 compressed) at 0x00009000 in 0.3 seconds (effective 401.6 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

13. Reboot the device and note you now have console access which does not require authentication:

```
I (4660) timeErrorSNTP (us): No Valid Data Yet
raven>     tI (4677) timeAudioPreprocessingInSec: No Valid Data Ye
I (4680) timeAudioPostProcessingInSec: No Valid Data Yet
I (4683) timeGpsPoweredOnInSec: No Valid Data Yet
I (4685) timeForGpsLockInSec: No Valid Data Yet
I (4698) STATS: * * * * * * * * * * * * * * * *
I (4701) GPS HELP: Begin GPS Sync
W (4703) GPS HELP: Started GPS Config task
W (4707) TIME: Triggered Background Audio
I (4711) NET_INT: Network actions pending!
I (4713) NET_INT: Connecting network
I (4726) NET_INT: Request LTE Modem Init
I (4728) LTE: Initializing modem!
I (4730) LTE: Handling request to init LTE
I (4734) uart: queue free spaces: 30
I (4742) LTE: Initialize LTE module
I (4745) LTE: Enable power to v2 LTE board
raven>     test
raven>
raven>
```

```
raven> help
test
  Enter test console mode

end_test
  Exit test console mode

query
  Query device status

disable_console
  Disable test console in NVS

gps_config
  Run GPS config test

gps_func
  Run GPS functional test

audio
  Run audio energy measurement

lte
  Run LTE test
```

Tools Used:

- Esptool
- Esp32knife
- UART Adapater
- strings

Mitigation: Implement a password that is hashed at rest to debug console access. Add ‘authorized use only’ and other regulatory banners.

FINDING 4: Hardcoded Wi-Fi Credentials Auto Connect – Raven Gunshot Detection

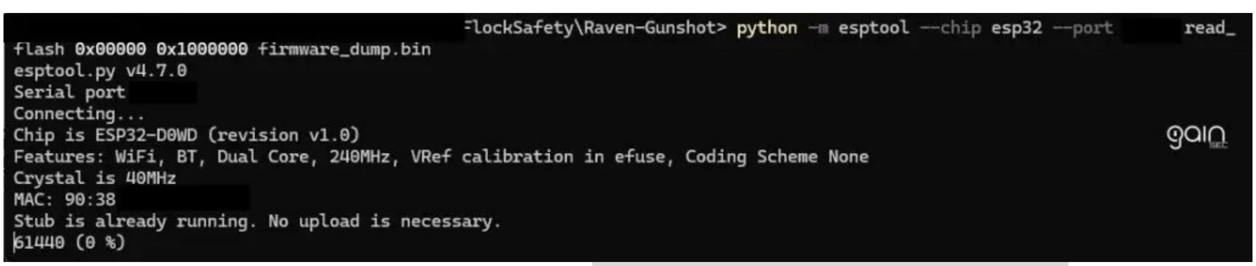
Description		
Type: Hardcoded Credentials	The Raven gunshot detection system was found to store cleartext SSID & Password within its firmware. The device automatically connects to this SSID if the LTE modem is unavailable/not configured.	**CVSS Score:** 7.2
Threat Context: Inexperienced Attacker		**Severity:** HIGH
Public Full Disclosure Date: 06/19/2025	**Impact** An attacker can leverage this to obtain a person-in-the-middle (PiTM) position, allowing intercepting of the devices network traffic.	**CVE #:** [CVE-2025-47818](CVE-2025-47818)
CVSS 4.0 AV:A/AC:L/AT:N/PR:N/UI:N/VC:H/VI:L/VA:L/SC:N/SI:N/SA:N		**CWE:** CWE- 259
Discovered By: Jon Gaines	**Affected Hardware/Software** Raven Gunshot Detection	**Further Research Recommended:** N
Notes This finding was improperly included in CVE-2025-47818 instead of being given its own CVE # when the Vendor submitted the CVE assignment request.		
Relevant Output: I (116066) WIFI: Preferred SSID not set. Using flockApList. I (116072) WIFI: Connecting to SSID Flock I (116088) WIFI: wifi_start finished. I (116093) NET_INT: Network connect to wifi returned ok		

Steps to Reproduce:

1. Open the case of the device.
2. Connect probes at the UART pad on the devices mainboard.
3. Connect a TTL/UART adapter to the probes.
4. Hold <REDACTED_PIN> and EN pads down.
5. Turn it on and let EN float.
6. You are now in 'DOWNLOAD_BOOT' mode:

```
UART> bridge
UART bridge. Press Bus Pirate button to exit.
ets Jun  8 2016 00:22:57
rst:0x1                                   (POWERON_RESET),boot:0x3
(DOWNLOAD_BOOT(UART0/UART1/SDIO_REI_REO_V2))
waiting for download
```

7. Use 'esptool' or similar to dump the firmware: <REDACTED_COMMAND>



```
[redacted]FlockSafety\Raven-Gunshot> python -m esptool --chip esp32 --port [redacted] read_
flash 0x00000 0x1000000 firmware_dump.bin
esptool.py v4.7.0
Serial port [redacted]
Connecting...
Chip is ESP32-D0WD (revision v1.0)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: 90:38[redacted]
Stub is already running. No upload is necessary.
61440 (0 %)
```

8. Search for 'Flock' within the firmware dump: strings firmware_dump.bin | grep 'Flock

Output:

```
Flock
<REDACTED_PASSWORD>
35|[
```

Flock-230503
<REDACTED_PASSWORD>
s>:pW
Flock
<REDACTED_PASSWORD>

9. Set up an access point (AP) with the SSID of 'Flock' or 'Flock-230503.' And the password of '<REDACTED_PASSWORD>' or ''<REDACTED_PASSWORD>" respectively.
10. Boot up the device, ensuring that the LTE modem cannot connect or is unplugged.
11. Note it auto connects to the AP:
12. I (113228) NET_INT: Network actions pending!
13. I (113229) NET_INT: Connecting network
14. I (113294) wifi_init: WiFi/LWIP prefer SPIRAM
15. I (113301) phy_init: phy_version 4670,719f9f6,Feb 18 2021,17:07:07
16. I (115947) WIFI: Found 13 networks
17. I (115948) WIFI: Network found SSID SectorI
18. I (115949) WIFI: Signal Strength = -22
19. I (115960) WIFI: Network found SSID Flock
20. I (115964) WIFI: Signal Strength = -24
21. I (116066) WIFI: Preferred SSID not set. Using flockApList.
22. I (116072) WIFI: Connecting to SSID Flock
23. I (116088) WIFI: wifi_start finished.
24. I (116093) NET_INT: Network connect to wifi returned ok
25. I (116159) WIFI: WiFi Connected to AP
26. I (116985) esp_netif_handlers: sta ip: 192.168.191.60, mask: 255.255.255.0, gw: 192.168.191.1
27. I (116987) NET_INT: got ip:192.168.191.60
28. I (116990) NET_INT: Setting modem sleep mode to WIFI_PS_NONE

Alternative Steps for Reproduction:

1. Carve out the 'NVS' partition from the full firmware dump: <REDACTED_COMMAND>
2. Alternatively, dump the NVS partition specifically: <REDACTED_COMMAND>
3. Convert the 'NVS' dump to a CSV file: <REDACTED_COMMAND>
4. Open the CSV and note the 'sta.apinfo' values (binary blob_data/sta.apinfo.bin):
5. 00000000: 0500 0000 466c 6f63 6b00 0000 0000 0000 0000 0000 0000 0000 :….Flock…………..
6. 00000018: 0000 0000 7365 6375 7269 79 0000 0000 0000 0000 0000 <REDACTED_PASSWORD>
7. 00000030: 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 :……………………
8. 00000048: 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 :……………………
9. 00000060: 0000 0000 00ff ffff ffff ffa3 37ad 0aa6 4ea4 b2c0 fd07 0927 :…………7…N…..'
10. 00000078: 49b8 2b51 3df0 0933 f1d1 28cc ec05 3335 7c5b 850b 0c00 0000 :I.+Q=..3..(…35|[……
11. 00000090: 466c 6f63 6b2d 3233 3035 3033 0000 0000 0000 0000 0000 0000 :Flock-230503…………
12. 000000a8: 0000 0000 0000 0000 7365 6375 7269 7479 0000 0000 <REDACTED_PASSWORD>
13. 000000c0: 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 :……………………
14. 000000d8: 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 :……………………
15. 000000f0: 00ff ffff ffff ff8c 733e 3a70 571b 5a07 45df b2a8 af1c fd43 :……..s>:pW.Z.E……C
16. 00000108: 16d2 e115 df65 25b5 88d7 0d54 99e5 3c0b 0500 0000 466c 6f63 :…..e%….T..<…..Floc
17. 00000120: 6b00 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 :k…………………..

18. 00000138: 0000 0000 4179 3454 776e 0000 0000 0000 0000 0000 0000 :<REDACTED_PASSWORD>

Tools Used:

- Esptool
- Esp32knife
- UART Adapater
- strings

Mitigation: Remove hardcoded Wi-Fi credentials before deployment. Disable Wi-Fi auto connections.

FINDING 5: Lack of Flash Encryption - Raven

Description		
Type: Cryptographic Failures	The Raven gunshot detection system was found to lack flash encryption. This enables an attacker with physical access the ability to read or dump the device's firmware in cleartext.	**CVSS Score:** 7.2
Threat Context: Experienced Attacker		**Severity:** HIGH
Public Full Disclosure Date: 06/19/2025	**Impact** An attacker with physical access can read or dump the devices firmware	**CVE #:** CVE-2025-47820
CVSS 4.0 AV:P/AC:L/AT:N/PR:N/UI:N/VC:H/VI:H/VA:H/SC:N/SI:N/SA:N		**CWE:** CWE- 312
Discovered By: Jon Gaines	**Affected Hardware/Software** Raven Gunshot Detection	**Further Research Recommended:** N
Notes		

Relevant Output:

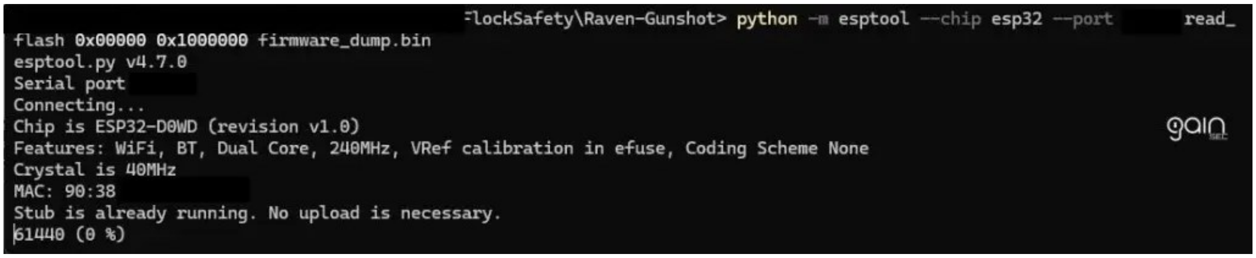
```
Flash fuses: FLASH_CRYPT_CNT (BLOCK0) = 0 R/W (0b0000000)
             FLASH_CRYPT_CONFIG (BLOCK0) = 0 R/W (0x0)
             BLOCK1 (BLOCK1) Flash encryption key= 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 R/W
             DISABLE_DL_ENCRYPT (BLOCK0) = False R/W (0b0)
             DISABLE_DL_DECRYPT (BLOCK0) = False R/W (0b0
```

Steps to Reproduce:

1. Open the case of the device.
2. Connect probes at the UART pad on the devices mainboard.
3. Connect a TTL/UART adapter to the probes.
4. Hold <REDACTED_PIN> and EN pads down.
5. Turn it on and let EN float.
6. You are now in 'DOWNLOAD_BOOT' mode.

```
UART> bridge
UART bridge. Press Bus Pirate button to exit.
ets Jun  8 2016 00:22:57
rst:0x1 (POWERON_RESET),boot:0x3 (DOWNLOAD_BOOT(UART0/UART1/SDIO_REI_REO_V2))
waiting for download
```

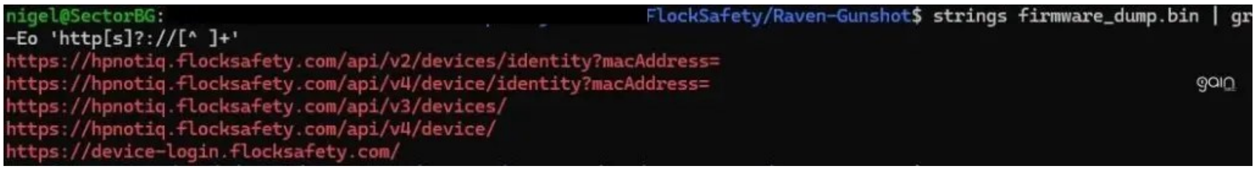
7. Use 'esptool' or similar to dump the firmware: <REDACTED_COMMAND>



```
[redacted]FlockSafety\Raven-Gunshot> python -m esptool --chip esp32 --port [redacted] read_
flash 0x00000 0x1000000 firmware_dump.bin
esptool.py v4.7.0
Serial port [redacted]
Connecting...
Chip is ESP32-D0WD (revision v1.0)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: 90:38[redacted]
Stub is already running. No upload is necessary.
61440 (0 %)
```

8. Confirm the firmware is in cleartext:

strings firmware_dump.bin | grep -Eo ‘http[s]?://[*]+’



```
nigel@SectorBG:[redacted]FlockSafety/Raven-Gunshot$ strings firmware_dump.bin | gr
-Eo 'http[s]?://[^ ]+'
https://hpnotiq.flocksafety.com/api/v2/devices/identity?macAddress=
https://hpnotiq.flocksafety.com/api/v4/device/identity?macAddress=
https://hpnotiq.flocksafety.com/api/v3/devices/
https://hpnotiq.flocksafety.com/api/v4/device/
https://device-login.flocksafety.com/
```

Alternative Steps to Reproduce:

1. Follow steps 1-3 of the previous instance.
2. Use ‘esptool’ to dump the ‘espfuse’ settings and values:
   <REDACTED_COMMAND>

3. Note the ‘BLOCK1’ value, showing that flash encryption is disabled:
   BLOCK1 (flash_encryption) [1] read_regs: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

4. Additionally, the following command can also confirm the lack of flash encryption: <REDACTED_COMMAND>
5. Output:
   FLASH_CRYPT_CNT (BLOCK0) = 0 R/W (0b0000000)
   FLASH_CRYPT_CONFIG (BLOCK0) = 0 R/W (0x0)
   BLOCK1 (BLOCK1) Flash encryption key
   = 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 000 R/W
   DISABLE_DL_ENCRYPT (BLOCK0) = False R/W (0b0)
   DISABLE_DL_DECRYPT (BLOCK0) = False R/W (0b0)

Tools Used:

- Esptool
- UART Adapater
- strings

Mitigation: Enable flash encryption

FINDING 6: Debug Interface Accessible (JTAG) – Raven Gunshot Detection

Description		
Type: Improper Access Control	The Raven gunshot detection system was found to have JTAG enabled. This enables an attacker with physical access to access this debug interface.	**CVSS Score:** 7.2
Threat Context: Inexperienced Attacker		**Severity:** HIGH
Public Full Disclosure Date: 06/19/2025	**Impact** An attacker with physical access can interface with the JTAG interface which can result in the following: unauthorized access, firmware extraction, and potential code manipulation. This could lead to intellectual property theft, device cloning, or attackers bypassing security protections.	**CVE #:** CVE-2025-47819
CVSS 4.0 AV:P/AC:L/AT:N/PR:N/UI:N/VC:H/VI:H/VA:H/SC:N/SI:N/SA:N		**CWE:** CWE- 1191
Discovered By: Jon Gaines	**Affected Hardware/Software** Raven Gunshot Detection	**Further Research Recommended:** N
Notes		
Relevant Output:		

```
Flash fuses: FLASH_CRYPT_CNT (BLOCK0) = 0 R/W (0b0000000)
              FLASH_CRYPT_CONFIG (BLOCK0) = 0 R/W (0x0)
             BLOCK1 (BLOCK1) Flash encryption key= 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 R/W
             DISABLE_DL_ENCRYPT (BLOCK0) = False R/W (0b0)
             DISABLE_DL_DECRYPT (BLOCK0) = False R/W (0b0
```

Steps to Reproduce:

1. Open the case of the device.
2. Point probesat the UART pad on the devices mainboard.
3. Connect a TTL/UART adapter to the probes.
4. Use ‘esptool’ to dump the ‘espfuse’ settings and values:
   ```
   <REDACTED_COMMAND>
   ```
5. Note the ‘BLOCK0’ value, showing that JTAG is not disabled:
   ```
   JTAG_DISABLE (BLOCK0)   Disable JTAG= False R/W (0b0)
   ```

Tools Used:

- Esptool
- UART Adapater
- strings

Mitigation: Disable the JTAG interface.

FINDING 7: Debug Interface Accessible (UART Download) – Raven Gunshot Detection

<table>
<tr><th colspan="3">Description</th></tr>
<tr><td>Type: Misconfiguration</td><td rowspan="2">The Raven gunshot detection system was found to have JTAG enabled. This enables an attacker with physical access to access this debug interface.</td><td>CVSS Score: 5.3</td></tr>
<tr><td>Threat Context: Inexperienced Attacker</td><td>Severity: MEDIUM</td></tr>
<tr><td rowspan="2">Public Full Disclosure Date: 06/19/2025</td><td>Impact</td><td rowspan="2">CVE #: CVE-2025-47819</td></tr>
<tr><td>An attacker with physical access can interface with the JTAG interface which can result in the following: unauthorized access, firmware extraction, and potential code manipulation. This could lead to intellectual property theft, device cloning, or attackers bypassing security protections.</td></tr>
<tr><td colspan="2">CVSS 4.0 AV:P/AC:L/AT:N/PR:N/UI:N/VC:L/VI:H/VA:H/SC:N/SI:N/SA:N</td><td>CWE: CWE- 1299</td></tr>
<tr><td rowspan="2">Discovered By:
Jon Gaines</td><td>Affected Hardware/Software</td><td rowspan="2">Further Research Recommended: N</td></tr>
<tr><td>Raven Gunshot Detection</td></tr>
<tr><td colspan="3">Notes
This finding was improperly included in CVE-2025-47819 instead of being given its own CVE number when the Vendor submitted the CVE number assignment request.</td></tr>
<tr><td colspan="3">Relevant Output:
UART_DOWNLOAD_DIS (BLOCK0) = False R/W (0b0)</td></tr>
</table>

Steps to Reproduce:

1. Open the case of the device.
2. Connect probes at the UART pad on the devices mainboard.
3. Connect a TTL/UART adapter to the probes.
4. Hold <REDACTED_PIN> and EN pads down.
5. Turn it on and let EN float.
6. You are now in UART 'DOWNLOAD_BOOT' mode.

```
UART> bridge
UART bridge. Press Bus Pirate button to exit.
ets Jun  8 2016 00:22:57
rst:0x1
(POWERON_RESET),boot:0x3 (DOWNLOAD_BOOT(UART0/UART1/SDIO_REI_REO_V2))
waiting for download
```

Alternative Steps to Reproduce:

1. Follow steps 1-3 of the previous instance.
2. Use 'esptool' to dump the 'espfuse' settings and values:
   <REDACTED_COMMAND>
3. Note the 'BLOCK0' value, showing that flash encryption is disabled:
   UART_DOWNLOAD_DIS (BLOCK0) = False R/W (0b0)
4. Additionally, the following command can also confirm the UART Download Mode Support is enabled:
   <REDACTED_COMMAND>

Tools Used:

- Esptool
- UART Adapater
- strings

Mitigation: Disable the UART Download Mode support.

FINDING 8: No Anti-Rollback Protection – Raven Gunshot Detection

Description		
Type: Misconfiguration	The Raven gunshot detection system was found to have ‘Rollback Protection’ disabled. Rollback protection is a security feature that prevents a system from being reverted to an earlier, potentially vulnerable version of its firmware.	**CVSS Score:** 5.3
Threat Context: Inexperienced Attacker		**Severity:** MEDIUM
Public Full Disclosure Date: 06/19/2025	**Impact**	**CVE #:** N/A
	An attacker with physical access can install older and vulnerable firmware onto the device.	
CVSS 4.0 AV:P/AC:L/AT:N/PR:N/UI:N/VC:L/VI:H/VA:H/SC:N/SI:N/SA:N		**CWE:** CWE- 1299
Discovered By:	**Affected Hardware/Software**	**Further Research Recommended:** N
Jon Gaines	Raven Gunshot Detection	
Notes		
This finding was improperly included in CVE-2025-47819 instead of being given its own CVE number when the Vendor submitted the CVE number assignment request.		
Relevant Output:		
SECURE_VERSION (BLOCK3) Secure version for anti-rollback = 0 R/W (0x00000000)		

Steps to Reproduce:

1. Open the case of the device.
2. Connect probes at the UART pad on the devices mainboard.
3. Connect a TTL/UART adapter to the probes.
4. Use ‘esptool’ to dump the ‘espfuse’ settings and values:
   <REDACTED_COMMAND>
5. Note the ‘BLOCK2’ value, showing that flash encryption is disabled:
   SECURE_VERSION (BLOCK3) Secure version for anti-rollback = 0 R/W (0x00000000)
6. Additionally, the following command can also confirm the UART Download Mode Support is enabled:
   <REDACTED_COMMAND>

Tools Used:

- Esptool
- UART Adapater
- strings

Mitigation: Enable Rollback Protection.

FINDING 9: Audio ML/AI Model Disclosed – Raven Gunshot Detection

<table>
<tr><th colspan="3">Description</th></tr>
<tr><td>Type: Cryptographic Failure</td><td rowspan="2">The Raven gunshot detection system was found to lack flash encryption. This resulted in the devices gunshot recognition model to be accessible.</td><td>CVSS Score: 5.3</td></tr>
<tr><td>Threat Context: Inexperienced Attacker</td><td>Severity: MEDIUM</td></tr>
<tr><td rowspan="2">Public Full Disclosure Date: 06/19/2025</td><td>Impact</td><td rowspan="2">CVE #: N/A</td></tr>
<tr><td>Plaintext AI/ML binaries let any local or remote foothold copy, reverse, or tamper with inference logic, enabling model plagiarism, rapid bypass of decision thresholds, targeted poisoning of detections, and seamless chaining into the already-documented vulnerabilities.</td></tr>
<tr><td colspan="2">CVSS 4.0 AV:P/AC:L/AT:N/PR:N/UI:N/VC:L/VI:H/VA:H/SC:N/SI:N/SA:N</td><td>CWE: CWE- 1299</td></tr>
<tr><td rowspan="2">Discovered By:
Jon Gaines</td><td>Affected Hardware/Software</td><td rowspan="2">Further Research Recommended: N</td></tr>
<tr><td>Raven Gunshot Detection</td></tr>
<tr><td colspan="3">Notes
This finding is a affect of Finding 5.</td></tr>
<tr><td colspan="3">Relevant Output:</td></tr>
</table>

Steps to Reproduce:

1. Open the case of the device.
2. Connect probes at the UART pad on the devices mainboard.
3. Connect a TTL/UART adapter to the probes.
4. Hold <REDACTED_PIN> and EN pads down.
5. Turn it on and let EN float.
6. You are now in ‘DOWNLOAD_BOOT’ mode.

```
UART> bridge
UART bridge. Press Bus Pirate button to exit.
ets Jun  8 2016 00:22:57
rst:0x1 (POWERON_RESET),boot:0x3 (DOWNLOAD_BOOT(UART0/UART1/SDIO_REI_REO_V2))
waiting for download
```

7. Use ‘esptool’ or similar to dump the firmware: <REDACTED_COMMAND>
8. Extract the ML/AI Audio model that runs on the unit within the <REDACTED_PARTITION> by using the following command: <REDACTED_COMMAND>
9. Confirm its validity by checking the ‘audio_model.bin’ for Syntiant File Signatures.

Tools Used:

- Esptool
- UART Adapater
- strings
- file

Mitigation: Enable Encryption.

FINDING 10: Hardcoded Credentials – API Client Secret – Raven Gunshot Detection

Description		
Type: Misconfiguration	The Raven gunshot detection system was found to store cleartext API client ID and client secret in cleartext.	**CVSS Score:** 2.3
Threat Context: Inexperienced Attacker		**Severity:** LOW
Public Full Disclosure Date: 06/19/2025	**Impact** An attacker can leverage these API credentials to flood, access or otherwise compromise the devices Cloud API.	**CVE #:** [CVE-2025-47821](CVE-2025-47821)
CVSS 4.0 AV:A/AC:H/AT:N/PR:N/UI:N/VC:L/VI:L/VA:N/SC:N/SI:N/SA:N		**CWE:** CWE- 259
Discovered By: Jon Gaines	**Affected Hardware/Software** Raven Gunshot Detection	**Further Research Recommended:** N
Notes		
Relevant Output: clientId data string xvtgsytnYyrs7pk88Q4vLQSb*BREDACTED* clientSecret data string BcyZHIz-D49AqQsW83hKdYvXv7W3p8jzc_*REDACTED*		

Steps to Reproduce:

1. Open the case of the device.
2. Connect probes at the UART pad on the devices mainboard.
3. Connect a TTL/UART adapter to the probes.
4. Hold <REDACTED_PIN> and EN pads down.
5. Turn it on and let EN float.
6. You are now in 'DOWNLOAD_BOOT' mode**.**

```
UART> bridge
UART bridge. Press Bus Pirate button to exit.
ets Jun  8 2016 00:22:57
rst:0x1 (POWERON_RESET),boot:0x3 (DOWNLOAD_BOOT(UART0/UART1/SDIO_REI_REO_V2))
waiting for download
```

```
[redacted]FlockSafety\Raven-Gunshot> python -m esptool --chip esp32 --port [redacted] read_flash 0x00000 0x1000000 firmware_dump.bin
esptool.py v4.7.0
Serial port [redacted]
Connecting...
Chip is ESP32-D0WD (revision v1.0)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: 90:38[redacted]
Stub is already running. No upload is necessary.
61440 (0 %)
```

7. Dump the NVS partition.
8. Convert the 'NVS' dump to a CSV file.
9. Open the CSV and note the 'clientId' and 'clientSecret' disclosed:

```
GNU nano 6.2
# NVS csv file
key,type,encoding,value
raven_nvs,namespace,,
isRegistered,data,u8,1
clientId,data,string,xvtgsytnYyrs7pk88Q4vLQSbBRCu38GW
clientSecret,data,string,BcyZHIz-D49AqQsW83hKdYvXv7W3p8jzc_wluP_cAP5cBmP3mQhNytTEz8BPwm9k
serialNumber,data,string,2[redacted]
partNumber,data,string,703-00006
consoleLogEn,data,u8,1
misc,namespace,,
nvs.net80211,namespace,,
ap.sndchan,data,u8,1
sta.chan,data,u8,0
sta.scan_method,data,u8,0
sta.sort_method,data,u8,0
sta.pmf_e,data,u8,1
sta.pmf_r,data,u8,0
sta.rrm_e,data,u8,0
sta.btm_e,data,u8,0
sta.ssid,data,base64,BQAAAEZsb2NrAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
sta.chan,data,u8,11
sta.chan,data,u8,0
sta.apinfo,file,binary,blob_data/sta.apinfo.bin
phy,namespace,,
cal_data,file,binary,blob_data/cal_data.bin
cal_mac,data,base64,kDgM5zIw
cal_version,data,u32,4670
```

```
[0;31mE (%u) %s: unsupported frequency configuration
rtc_clk_init
[0;33mW (%u) %s: Potentially bogus XTAL frequency: %d MHz, guessing 26 MHz
[0;33mW (%u) %s: Potentially bogus XTAL frequency: %d MHz, guessing 40 MHz
[0;33mW (%u) %s: Bogus XTAL frequency: %d MHz
[0;33mW (%u) %s: Can't estimate XTAL frequency, assuming 26MHz
[0;33mW (%u) %s: Possibly invalid CONFIG_ESP32_XTAL_FREQ setting (%dMHz). Detected %d MHz.
[0;31mE (%u) %s: invalid CPU frequency value
rtc_time
[0;31mE (%u) %s: slowclk_cycles value too large, possible overflow
[0;31mE (%u) %s: Range of data does not match the coding scheme
jcg8
@:cA
&clientId
clientSecret
|BcyZHIz-D49AqQsW83hKdYvXv7W3p8jzc_wluP_cAP5cBmP3mQhNytTEz8BPwm9k
misc
ap.sndchan
Flock
tsecurity
:sta.chan
sta.scan_method
Flock
security
Flock
security
gsta.chan
Flock
security
Flock-230503
:sta.chan
Flock
security
Flock-230503
security
Flock
security
Flock-230503
security
gsta.chan
Flock
```

10. Relevant Output:

```
CORD: Confidence Level above threshold. Found ML data to tag
E (848920) esp-tls: [sock=60] select() timeout
E (848927) esp-tls: Failed to open new connection
E (848927) TRANSPORT_BASE: Failed to open a new connection
E (848931) HTTP_CLIENT: Connection failed, sock < 0
E (849036) AUTH_HELPER: HTTP POST Fetch Auth Token request failed: ESP_ERR_HTTP_CONNECT, response code: 0
I (849038) HTTP_HELP: HTTP_EVENT_DISCONNECTED
I (849041) HTTP_HELP: HTTP_EVENT_DISCONNECTED
E (849047) AUTH_HELPER: Failed to cleanup http client
E (849057) AUTH_HELPER: Failed to initialize auth0 auth token: ESP_FAIL
E (849060) HPNOTIQ_HELP: Could not set authorization header in execute_call_to_hpnotiq: ESP_FAIL
I (849069) HTTP_HELP: HTTP_EVENT_DISCONNECTED
I (849074) HTTP_HELP: HTTP_EVENT_DISCONNECTED
W (849080) HPNOTIQ_HELP: Failed to authenticate through auth0 with hpnotiq, falling back to hardcoded api key
I (849111) HPNOTIQ_HELP: Hpnotiq host: https://hpnotiq.flocksafety.com/api/v2/devices/identity?macAddress=<REDACTED>, ip: <REDACTED>
I (849116) HPNOTIQ_HELP: Executing call to hpnotiq with deprecated authentication
I (851446) NDP_TASK: Finished recording.
```

Tools Used:

- Esptool
- Esp32knife
- UART Adapater
- strings

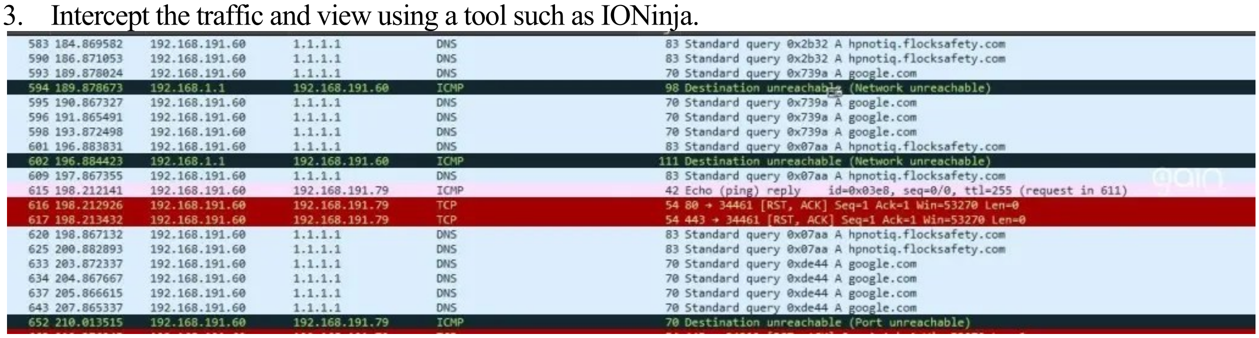
Mitigation: Do not hardcode credentials. Do not use static client secrets. Use rotating client secrets per authentication attempt and device.

FINDING 11: Lack of Server Verification (DNS Spoofing) – Raven Gunshot Detection

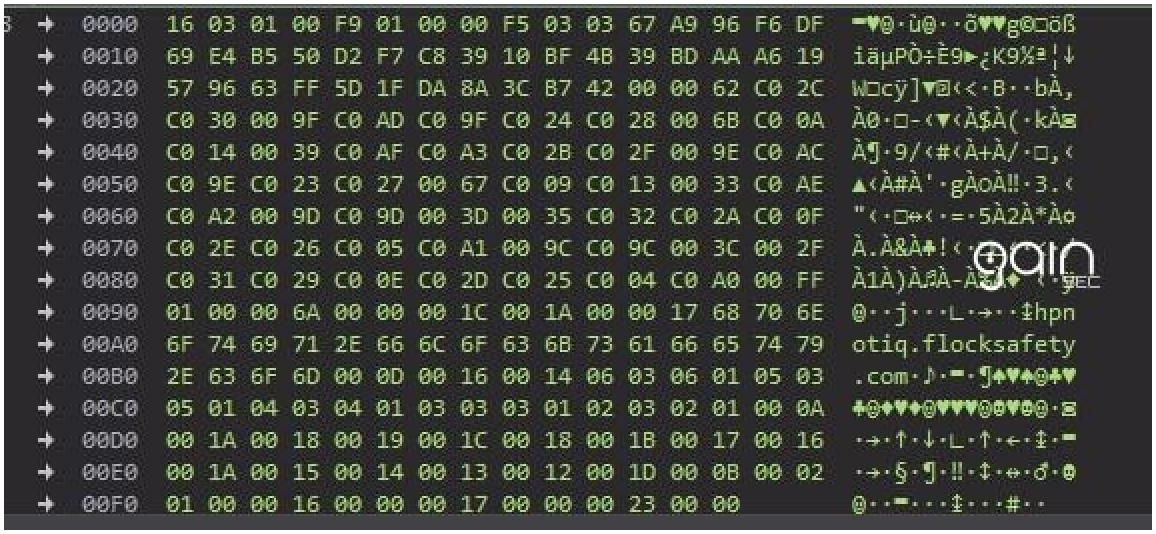
Description		
Type: Misconfiguration	The Raven gunshot detection system was found to store cleartext API client ID and client secret in cleartext.	**CVSS Score:** 2.3
Threat Context: Experienced Attacker		**Severity:** INFORMATIONAL
Public Full Disclosure Date: 06/19/2025	**Impact** An attacker on the same W/LAN can intercept encrypted communications via DNS spoofing.	**CVE #:** N/A
CVSS 4.0 AV:A/AC:H/AT:N/PR:N/UI:N/VC:N/VI:N/VA:N/SC:N/SI:N/SA:N		**CWE:** CWE- 295
Discovered By: Jon Gaines	**Affected Hardware/Software** Raven Gunshot Detection	**Further Research Recommended:** Y
Notes This finding requires further research.		
Relevant Output: The following subdomains were susceptible: device-login.flocksafety.com hpnotiq.flocksafety.com		

Steps to Reproduce:

1. Use DNSChef & and MITMRouter (Such as GainSec-in-the-middle) impelementation while on the same W/LAN as the device.
2. Ensure that the two subdomains are pointed at your own server.
3. Intercept the traffic and view using a tool such as IONinja.



4. View the encrypted traffic:




```
0000  16 03 01 00 F9 01 00 00 F5 03 03 67 A9 96 F6 DF
0010  69 E4 B5 50 D2 F7 C8 39 10 BF 4B 39 BD AA A6 19
0020  57 96 63 FF 5D 1F DA 8A 3C B7 42 00 00 62 C0 2C
0030  C0 30 00 9F C0 AD C0 9F C0 24 C0 28 00 6B C0 0A
0040  C0 14 00 39 C0 AF C0 A3 C0 2B C0 2F 00 9E C0 AC
0050  C0 9E C0 23 C0 27 00 67 C0 09 C0 13 00 33 C0 AE
0060  C0 A2 00 9D C0 9D 00 3D 00 35 C0 32 C0 2A C0 0F
0070  C0 2E C0 26 C0 05 C0 A1 00 9C C0 9C 00 3C 00 2F
0080  C0 31 C0 29 C0 0E C0 2D C0 25 C0 04 C0 A0 00 FF
0090  01 00 00 6A 00 00 00 1C 00 1A 00 00 17 68 70 6E   ···j···········hpn
00A0  6F 74 69 71 2E 66 6C 6F 63 6B 73 61 66 65 74 79   otiq.flocksafety
00B0  2E 63 6F 6D 00 0D 00 16 00 14 06 03 06 01 05 03   .com············
00C0  05 01 04 03 04 01 03 03 03 01 02 03 02 01 00 0A
00D0  00 1A 00 18 00 19 00 1C 00 18 00 1B 00 17 00 16
00E0  00 1A 00 15 00 14 00 13 00 12 00 1D 00 0B 00 02
00F0  01 00 00 16 00 00 00 17 00 00 00 23 00 00
```


Tools Used:

- IONinja
- GainSec-in-the-Middle

Mitigation: Implement a server validation method before the client connects

DEVICE 2: FALCON/SPARROW/FLEX* LICENSE PLATE READER (LPR)

FINDING 12: Root Shell - Falcon/Sparrow/Flex* LPR

<table>
<tr><th colspan="3">Description</th></tr>
<tr><td>Type: Broken Authentication/Authorization</td><td rowspan="2">The Falcon/Sparrow/Flex* LPR failed to prevent a root shell from being achieved. Root access results in complete device compromise.</td><td>CVSS Score: 9.8</td></tr>
<tr><td>Threat Context: Experienced Attacker</td><td>Severity: CRITICAL</td></tr>
<tr><td rowspan="2">Public Full Disclosure Date: 06/19/2025</td><td>Impact</td><td rowspan="2">CVE #: N/A</td></tr>
<tr><td>An attacker with physical access can get root access to the device.</td></tr>
<tr><td colspan="2">CVSS 4.0 AV:A/AC:L/AT:P/PR:N/UI:N/VC:H/VI:H/VA:H/SC:N/SI:N/SA:N</td><td>CWE: CWE- 306</td></tr>
<tr><td rowspan="2">Discovered By:
Jon Gaines</td><td>Affected Hardware/Software</td><td rowspan="2">Further Research Recommended: N</td></tr>
<tr><td>Falcon/Sparrow/Flex* License Plate Readers</td></tr>
<tr><td colspan="3">Notes</td></tr>
<tr><td colspan="3">Relevant Output:</td></tr>
</table>

Steps to Reproduce:

1. Download Magisk version 23.
2. Unzip Magisk and then copy the device's stock boot image to the Magisk directory.
3. Flip the dip switch off, turn on the device, flip the dip switch to on and connect USB.
4. Push the Magisk directory to the device.
5. Use 'adb shell' to access a shell on the device.
6. Navigate to the 32 bit magisk binary directory.
7. Copy or move the following binaries remove the '.so' extensions.
8. Use 'chmod' to be marked as executable.
9. Now 'chmod' the 'boot_patch.sh' script within the Magisk directory and then run it, patching the stock 'boot.img.'
10. Pull the patched image off the device.
11. Reboot into edl mode.
12. Flash the modified image.
13. Reboot and follow the same process to get adb access.
14. Uninstall the currently installed Magisk APK that was installed with the patched 'boot.img.'
15. Install the proper APK downloaded in step 1.
16. Use a tool like 'scrcpy' to mirror the LPR's screen and grant superuser privileges to terminal when executing 'su' for the first time.
17. Set selinux to permissive.

```
┌──(root㉿SectorTL)-[/home/nigel/magisk23]
└─# adb shell
msm8953_32:/ $ su
msm8953_32:/ #
msm8953_32:/ #
msm8953_32:/ # whoami
root
msm8953_32:/ # getenforce
Enforcing
msm8953_32:/ # setenforce 0
msm8953_32:/ # getenforce
Permissive
msm8953_32:/ #
```

Tools Used:

- edl
- MicroUSB cord
- adb
- Magisk

Mitigation: Apply the other findings mitigations.

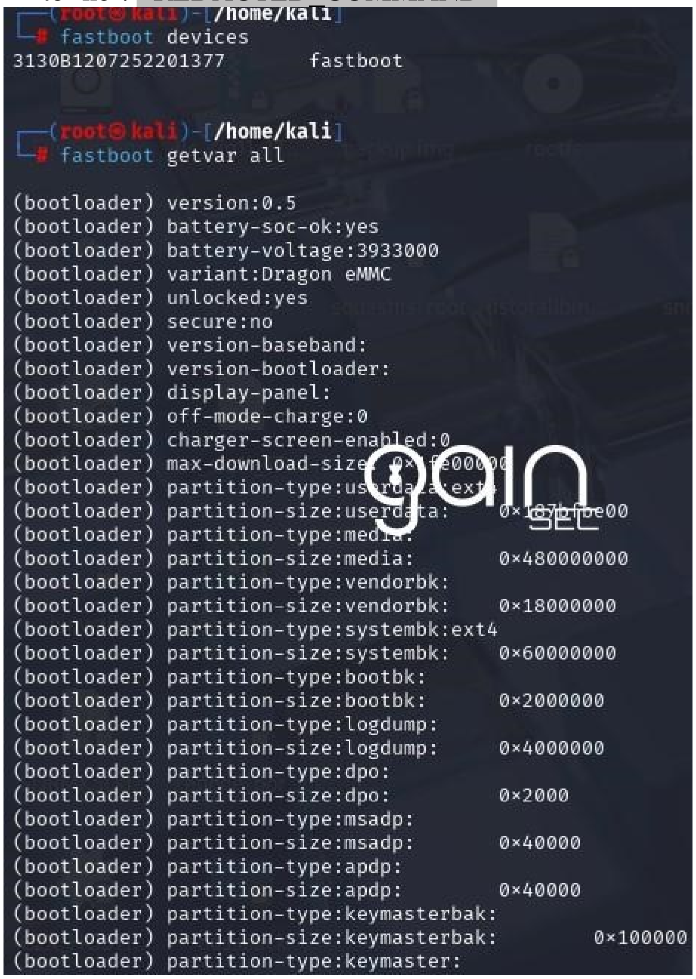
FINDING 13: Secure Boot is Disabled – Falcon/Sparrow/Flex* LPR

Description		
Type: Cryptographic Failures	The Falcon/Sparrow/Flex* LPR was found to have 'Secure Boot' disabled. Secure Boot is a security feature that ensures only trusted software runs during the device's startup process.	**CVSS Score:** 9.8
Threat Context: Inexperienced Attacker		**Severity:** CRITICAL
Public Full Disclosure Date: 06/19/2025	**Impact** Disabling Secure Boot allows unsigned or malicious bootloaders and kernel-level code to execute during system startup, undermining the trust chain and enabling persistent compromise at the firmware or OS level. This exposes the host to rootkits and pre-boot tampering undetectable by standard security controls.	**CVE #:** CVE-2025-47822
CVSS 4.0 AV:P/AC:L/AT:N/PR:N/UI:N/VC:H/VI:H/VA:H/SC:N/SI:N/SA:N		**CWE:** CWE- 1104
Discovered By: Jon Gaines	**Affected Hardware/Software** Falcon/Sparrow/Flex* License Plate Readers	**Further Research Recommended:** N
Notes		
Relevant Output:		

Steps to Reproduce:

1. Open the case of the device.
2. Flip the dip switch to off.
3. Hold down the volume down button and turn on the device.
4. Flip the dip switch to on.
5. Plug in a micro-usb to its port.
6. Note the device is now accessible in 'fastboot' mode.

7. Alternatively, if connected via Android Debug Bridge (ADB), use the following command: <REDACTED_COMMAND>
8. Use the following command to view the variables of the device and note the 'secure' variable is set to 'no': <REDACTED_COMMAND>



```
┌──(root㉿kali)-[/home/kali]
└─# fastboot devices
3130B1207252201377      fastboot

┌──(root㉿kali)-[/home/kali]
└─# fastboot getvar all

(bootloader) version:0.5
(bootloader) battery-soc-ok:yes
(bootloader) battery-voltage:3933000
(bootloader) variant:Dragon eMMC
(bootloader) unlocked:yes
(bootloader) secure:no
(bootloader) version-baseband:
(bootloader) version-bootloader:
(bootloader) display-panel:
(bootloader) off-mode-charge:0
(bootloader) charger-screen-enabled:0
(bootloader) max-download-size:0x[illegible]e00000
(bootloader) partition-type:userdata:ext4
(bootloader) partition-size:userdata:    0x[illegible]e00
(bootloader) partition-type:media:
(bootloader) partition-size:media:       0x480000000
(bootloader) partition-type:vendorbk:
(bootloader) partition-size:vendorbk:    0x18000000
(bootloader) partition-type:systembk:ext4
(bootloader) partition-size:systembk:    0x60000000
(bootloader) partition-type:bootbk:
(bootloader) partition-size:bootbk:      0x2000000
(bootloader) partition-type:logdump:
(bootloader) partition-size:logdump:     0x4000000
(bootloader) partition-type:dpo:
(bootloader) partition-size:dpo:         0x2000
(bootloader) partition-type:msadp:
(bootloader) partition-size:msadp:       0x40000
(bootloader) partition-type:apdp:
(bootloader) partition-size:apdp:        0x40000
(bootloader) partition-type:keymasterbak:
(bootloader) partition-size:keymasterbak:        0x100000
(bootloader) partition-type:keymaster:
```

Alternative Steps to Reproduce:

1. Put the device in EDL mode by pressing the 'Force USB' button when turning it on.
2. Plug a Micro-USB cord into its port.
3. Use the following command to confirm Secure Boot is disabled: <REDACTED_COMMAND>

```
Qualcomm Sahara / Firehose Client V3.62 (c) B.Kerler 2018-2024.
main - Waiting for the device
main - Using loader ALPR-DDR-FIREHOUSE.mbn ...
main - Device detected :)
main - Mode detected: firehose
Sec_Boot0 PKHash-Index:0 OEM_PKHash: False Auth_Enabled: FalseUse_Serial: False
Sec_Boot1 PKHash-Index:0 OEM_PKHash: False Auth_Enabled: FalseUse_Serial: False
Sec_Boot2 PKHash-Index:0 OEM_PKHash: False Auth_Enabled: FalseUse_Serial: False
Sec_Boot3 PKHash-Index:0 OEM_PKHash: False Auth_Enabled: FalseUse_Serial: False
Secure boot disabled.
```

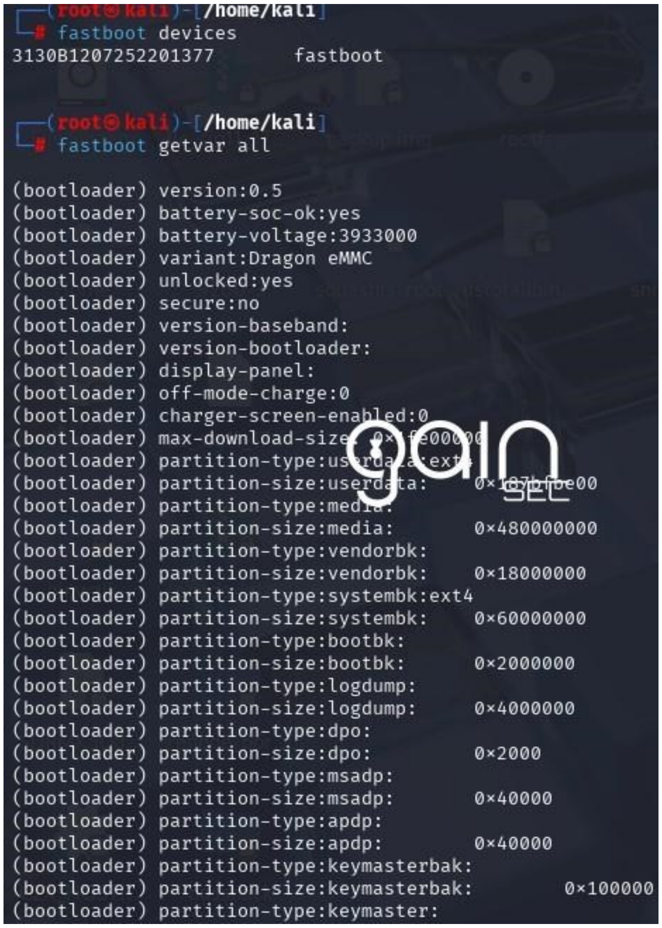
Tools Used:
- MicroUSB Cord
- fastboot

Mitigation: Enable Secure Boot

FINDING 14: Unlocked Bootloader – Falcon/Sparrow/Flex* LPR

Description		
Type: Improper Access Control	The Falcon/Sparrow/Flex* LPR Bootloader was found to be unlocked allowing unauthorized firmware to be installed.	**CVSS Score:** 9.8
Threat Context: Experienced Attacker		**Severity:** CRITICAL
Public Full Disclosure Date: 06/19/2025	**Impact**	**CVE #:** CVE-2025-47822
	An unlocked bootloader permits arbitrary unsigned firmware to be installed and executed on the device, effectively bypassing the device's root of trust. This yields full compromise of the device's security properties.	
CVSS 4.0 AV:P/AC:L/AT:N/PR:N/UI:N/VC:H/VI:H/VA:H/SC:N/SI:N/SA:N		**CWE:** CWE- 1299
Discovered By:	**Affected Hardware/Software**	**Further Research Recommended:** N
Jon Gaines	Falcon/Sparrow/Flex* License Plate Readers	
Notes		
Relevant Output:		

Steps to Reproduce:
1. Open the case of the device.
2. Flip the dip switch to off.
3. Hold down the volume down button and turn on the device.
4. Flip the dip switch to on.
5. Plug in a micro-usb to its port.
6. Note the device is now accessible in 'fastboot' mode.
7. Alternatively, if connected via Android Debug Bridge (ADB), use the following command: <REDACTED_COMMAND>
8. Use the following command to view the variables of the device and note the 'unlocked' variable is set to 'yes': <REDACTED_COMMAND>



Tools Used:
- Micro USB Cord
- fastboot

Mitigation: Lock Bootloader after installing firmware.

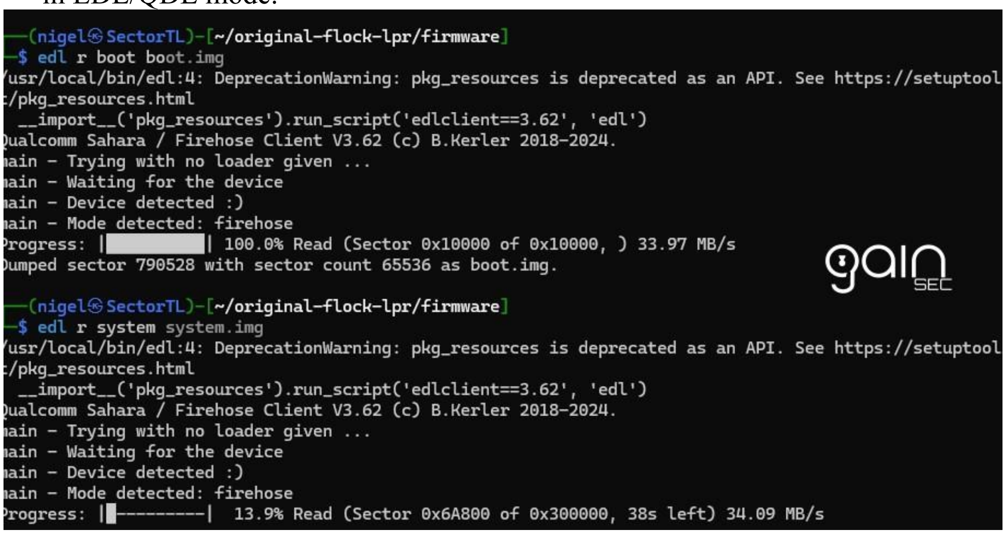
FINDING 15: Lack of Authentication: EDL/QDL Mode – Falcon/Sparrow/Flex* LPR

Description		
Type: Broken Authentication/Authorization	The Falcon/Sparrow/Flex* LPR EDL/QDL mode was found to lack any type of authentication or access control.	**CVSS Score:** 9.8
Threat Context: Inexperienced Attacker		**Severity:** CRITICAL
Public Full Disclosure Date: 06/19/2025	**Impact**	**CVE #:** [CVE-2025-47822]
	Attackers can exploit publicly known vulnerabilities that remain unpatched, enabling privilege escalation, remote code execution, or denial-of-service. Continued operation on an obsolete platform increases overall attack surface and compromises system integrity, confidentiality, and availability.	
CVSS 4.0 AV:P/AC:L/AT:N/PR:N/UI:N/VC:H/VI:H/VA:H/SC:N/SI:N/SA:N		**CWE:** CWE- 1299
Discovered By:	**Affected Hardware/Software**	

Jon Gaines	Falcon/Sparrow/Flex* License Plate Readers	**Further Research Recommended:** N
Notes: This finding was improperly included with CVE-2025-47822 instead of being given its own CVE # when the Vendor submitted the CVE assignment request.		
Relevant Output:		

Steps to Reproduce:

1. Put the device in EDL mode by pressing the ‘Force USB’ button when turning it on.
2. Plug a Micro-USB cord into its port.
3. Note the default provided ‘firehose’ or the default ‘msm8956’ firehose works with the device when in EDL/QDL mode:



Tools Used:

- Micro USB Cord
- edl

Mitigation: Implement a custom signed firehose that isn’t publicly available.

FINDING 16: Lack of Authentication – Android Debug Bridge - Falcon/Sparrow/Flex* LPR

Description		
Type: Broken Authentication/Authorization	The Falcon/Sparrow/Flex* LPR is configured to not require authentication (approval) when accessing the device via Android Debug Bridge (ADB).	**CVSS Score:** 8.8
Threat Context: Inexperienced Attacker		**Severity:** HIGH
Public Full Disclosure Date: 06/19/2025	**Impact** An attacker with physical access can get 'shell' access to the device.	**CVE #:** N/A
CVSS 4.0 AV:A/AC:L/AT:N/PR:N/UI:N/VC:L/VI:H/VA:H/SC:N/SI:N/SA:N		**CWE:** CWE- 287
Discovered By: Jon Gaines	**Affected Hardware/Software** Falcon/Sparrow/Flex* License Plate Readers	**Further Research Recommended:** N
Notes:		
Relevant Output:		

Steps to Reproduce:

1. Flip the dip switch off, turn on the device, flip the dip switch to on and connect USB.
2. Use 'adb shell' to access a shell on the device.
3. Note that 'developer options' were not required to be enabled and there was no required approval prompt on the device to approve ADB access. Lastly, specific ADB server keys are not preconfigured.

Tools Used:

- MicroUSB cord
- adb

Mitigation: Disable unauthenticated and unauthorized ADB access.

FINDING 17: Improper Access Control – Android Debug Bridge Sideload - Falcon/Sparrow/Flex* LPR

Description		
Type: Broken Authentication/Authorization	The Falcon/Sparrow/Flex* LPR is configured to allow sideloading apps via ADB.	**CVSS Score:** 8.8
Threat Context: Inexperienced Attacker		**Severity:** HIGH
Public Full Disclosure Date: 06/19/2025	**Impact** An attacker with physical access can install whatever application they want onto the device.	**CVE #:** N/A
CVSS 4.0 AV:A/AC:L/AT:N/PR:N/UI:N/VC:L/VI:H/VA:H/SC:N/SI:N/SA:N		**CWE:** CWE- 284
Discovered By: Jon Gaines	**Affected Hardware/Software** Falcon/Sparrow/Flex* License Plate Readers	**Further Research Recommended:** N
Notes:		
Relevant Output:		

Steps to Reproduce:

1. Flip the dip switch off, turn on the device, flip the dip switch to on and connect USB.
2. Use 'adb install example.apk to sideload an app. Note it is successful.

Tools Used:

- MicroUSB cord
- adb

Mitigation: Disable unauthenticated and unauthorized ADB access. Disallow sideloading via ADB.

FINDING 18: Lack of Flash/EMMC Encryption - Falcon/Sparrow/Flex* LPR

<table>
<tr><th colspan="3">Description</th></tr>
<tr><td>Type: Cryptographic Failures</td><td rowspan="2">The Falcon/Sparrow/Flex* LPR was found to lack flash/EMMC encryption. This encryption ensures that if the firmware is dumped from the device, it is unreadable</td><td>CVSS Score: 5.2</td></tr>
<tr><td>Threat Context: Inexperienced Attacker</td><td>Severity: MEDIUM</td></tr>
<tr><td rowspan="2">Public Full Disclosure Date: 06/19/2025</td><td>Impact</td><td rowspan="2">CVE #: CVE-2025-47824</td></tr>
<tr><td>An attacker with physical access can read or dump the device's firmware in cleartext.</td></tr>
<tr><td colspan="2">CVSS 4.0 AV:P/AC:L/AT:N/PR:N/UI:N/VC:H/VI:L/VA:N/SC:N/SI:N/SA:N</td><td>CWE: CWE- 312</td></tr>
<tr><td rowspan="2">Discovered By:
Jon Gaines</td><td>Affected Hardware/Software</td><td rowspan="2">Further Research Recommended: N</td></tr>
<tr><td>Falcon/Sparrow/Flex* License Plate Readers</td></tr>
<tr><td colspan="3">Notes:</td></tr>
<tr><td colspan="3">Relevant Output:</td></tr>
</table>

Steps to Reproduce:

1. Open the case of the device.
2. Put the device in Emergency Download Mode (EDL) by holding it's 'Force USB' button down while turning it on or by using a EDL flash cable.
3. Using the 'edl' tool, dump the firmware:
   <REDACTED_COMMAND>
4. Run 'strings' on the firmware dump and note cleartext.

Alternative Steps to Reproduce:

1. Dump a specific partition such as 'system' or 'userdata' and use a tool like 'debugfs' to view its contents:
   <REDACTED_COMMAND>
2. Note its contents are accessible:

```
2  (12) .    2  (12) ..    11  (20) lost+found    12  (12) app
327  (12) bin    723  (20) build.prop    724  (32) compatibility_matrix.xml
725  (12) etc    1100  (20) fake-libs    1102  (16) fonts
1109  (20) framework    1330  (12) lib    1912  (20) manifest.xml
1913  (16) priv-app    2069  (28) recovery-from-boot.p    2070  (12) usr
2131  (16) vendor    2132  (3804) xbin
(END)
```

Tools Used:

- edl
- MicroUSB cord
- debugfs

Mitigation: Implement Flash/EMMC encryption.

FINDING 19: Use of an Unsupported and End of Life Operating System - Falcon/Sparrow/Flex* LPR

<table>
<tr><th colspan="3">Description</th></tr>
<tr><td>Type: Vulnerable & Outdated Components</td><td rowspan="2">The Falcon/Sparrow/Flex* LPR was found to run Android Things v8.1, an OS that reached its end-of-life (EOL) in 2022. Post-EOL, the vendor ceases delivering security updates, leaving the system exposed to known and emerging vulnerabilities. Using unsupported software in production violates secure lifecycle management principles and undermines compliance with most cybersecurity baselines (e.g., CIS Controls, NIST 800-53 SI-2). Devices on deprecated OS versions are more susceptible to exploitation, as vulnerabilities remain unpatched and publicly documented exploit code often exists.</td><td>CVSS Score: 5.3</td></tr>
<tr><td>Threat Context: Experienced Attacker</td><td>Severity: MEDIUM</td></tr>
<tr><td rowspan="2">Public Full Disclosure Date: 06/19/2025</td><td>Impact</td><td rowspan="2">CVE #: N/A</td></tr>
<tr><td>Attackers can exploit publicly known vulnerabilities that remain unpatched, enabling privilege escalation, remote code execution, or denial-of-service. Continued operation on an obsolete platform increases overall attack surface and compromises system integrity, confidentiality, and availability.</td></tr>
<tr><td colspan="2">CVSS 4.0 AV:N/AC:L/AT:N/PR:N/UI:N/VC:L/VI:L/VA:N/SC:N/SI:N/SA:N</td><td>CWE: CWE- 1104</td></tr>
<tr><td rowspan="2">Discovered By:
Jon Gaines</td><td>Affected Hardware/Software</td><td rowspan="2">Further Research Recommended: N</td></tr>
<tr><td>Falcon/Sparrow/Flex* License Plate Readers</td></tr>
<tr><td colspan="3">Notes:</td></tr>
<tr><td colspan="3">Relevant Output:</td></tr>
</table>

Steps to Reproduce:
1. Boot up the device and connect via ADB.
2. Use commands such as getprop to view the variant and version of Android the device is running.

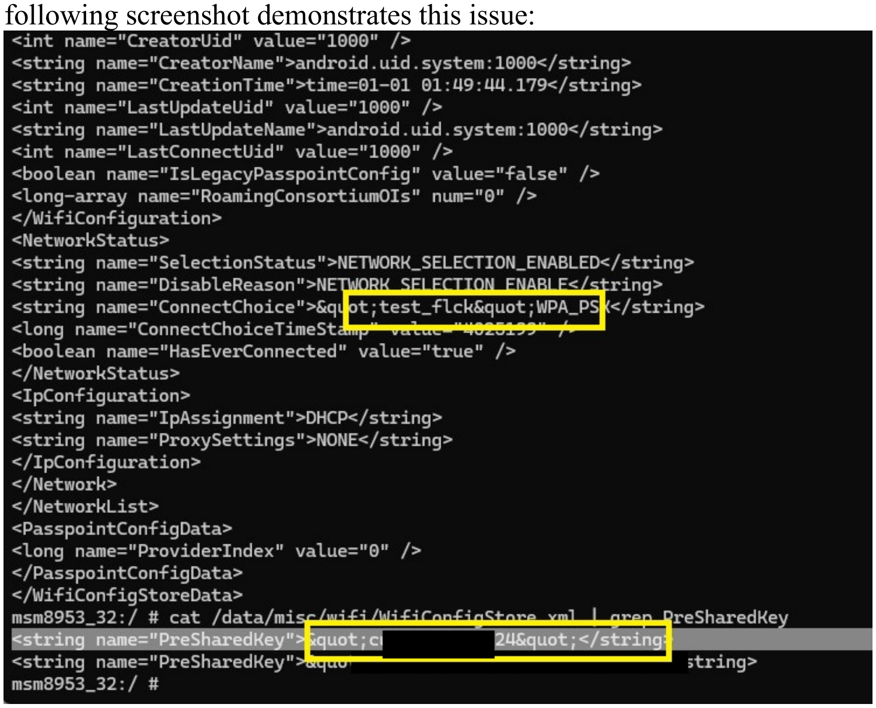
Tools Used:
- adb
- MicroUSB cord

Mitigation: Apply the other findings mitigations. Use hardware that runs an actively supported EOS.

FINDING 20: Sensitive Information Disclosed – Development/Test Credential in Production – Falcon/Sparrow/Flex* LPR

Description		
Type: Debug Functionality	One of the Falcon/Sparrow/Flex* LPR units was found to contain development/test credentials in clear text. In this case, this was for the 'test_flck' Wi-Fi network. It was found that if the wireless interface was brought up or if the modem could not connect, the device would automatically connect to any AP with that name and password.	**CVSS Score:** 3.5
Threat Context: Inexperienced Attacker		**Severity:** LOW
Public Full Disclosure Date: 09/27/2025	**Impact** An attacker with physical or local access can steal this password. Additionally, an attacker cam set up a malicious AP (evil twin) positioning themselves a 'person-in-the-middle' PiTM when the device auto connects too when its Wi-Fi is enabled.	**CVE #:** CVE-2025-59409
CVSS 4.0 AV:L/AC:L/AT:N/PR:N/UI:N/VC:H/VI:N/VA:N/SC:N/SI:N/SA:N		**CWE:** CWE- 1299
Discovered By: Jon Gaines	**Affected Hardware/Software** Falcon/Sparrow/Flex* License Plate Readers	**Further Research Recommended:** N
Notes: Used in an attack chain with Findings such as 29 or 30, this finding's severity would greatly increase.		
Relevant Output:		

The following screenshot demonstrates this issue:



```
<int name="CreatorUid" value="1000" />
<string name="CreatorName">android.uid.system:1000</string>
<string name="CreationTime">time=01-01 01:49:44.179</string>
<int name="LastUpdateUid" value="1000" />
<string name="LastUpdateName">android.uid.system:1000</string>
<int name="LastConnectUid" value="1000" />
<boolean name="IsLegacyPasspointConfig" value="false" />
<long-array name="RoamingConsortiumOIs" num="0" />
</WifiConfiguration>
<NetworkStatus>
<string name="SelectionStatus">NETWORK_SELECTION_ENABLED</string>
<string name="DisableReason">NETWORK_SELECTION_ENABLE</string>
<string name="ConnectChoice">&quot;test_flck&quot;WPA_PSK</string>
<long name="ConnectChoiceTimeStamp" value="4025199" />
<boolean name="HasEverConnected" value="true" />
</NetworkStatus>
<IpConfiguration>
<string name="IpAssignment">DHCP</string>
<string name="ProxySettings">NONE</string>
</IpConfiguration>
</Network>
</NetworkList>
<PasspointConfigData>
<long name="ProviderIndex" value="0" />
</PasspointConfigData>
</WifiConfigStoreData>
msm8953_32:/ # cat /data/misc/wifi/WifiConfigStore.xml | grep PreSharedKey
<string name="PreSharedKey">&quot;c[redacted]24&quot;</string>
<string name="PreSharedKey">&quot[redacted]string>
msm8953_32:/ #
```

Tools Used:

- Micro USB Cord
- adb
- cat

Mitigation: Do not deploy devices with test or development configurations on them.

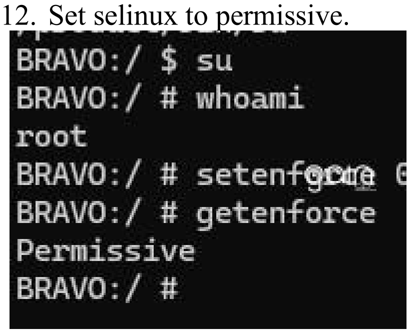
DEVICE 3: PICARD/BRAVO COMPUTE BOX

FINDING 21: Root Shell – Picard/Bravo Compute Box

Description		
Type: Broken Authentication/Authorization	The Compute Box failed to prevent a root shell from being achieved. Root access results in complete device compromise	**CVSS Score:** 9.8
Threat Context: Inexperienced Attacker		**Severity:** CRITICAL
Public Full Disclosure Date: 09/19/2025	**Impact**	**CVE #:** N/A
	An attacker with physical access can get root access to the device.	
CVSS 4.0 AV:P/AC:L/AT:N/PR:N/UI:N/VC:H/VI:H/VA:H/SC:N/SI:N/SA:N		**CWE:** CWE- 306
Discovered By: Jon Gaines	**Affected Hardware/Software** Picard/Bravo Compute Box	**Further Research Recommended:** N
Notes:		
Relevant Output:		

Steps to Reproduce:

1. Null out AVB by generating a custom ‘vbmeta_a’ partition that follows the proper boot order: <REDACTED_COMMAND>
2. Generate an empty ‘vbmeta_system_a’ image: <REDACTED_COMMAND>
3. Boot into EDL mode and write the two new partitions: <REDACTED_COMMAND>
4. Download Magisk version 29 or newer.
5. Sideload Magisk to install it.
6. Push a copy of the devices ‘boot_a’ partition.
7. Use a tool like ‘scrcpy’ to open Magisk and patch the ‘boot_a’ partition.
8. Pull the ‘boot_a’ partition off the device and boot into EDL mode.
9. Flash the ‘boot_a’ partition: <REDACTED_COMMAND>
10. Reboot.
11. Use a tool like ‘scrcpy’ to mirror the LPR’s screen and grant superuser privileges to terminal when executing ‘su’ for the first time.
12. Set selinux to permissive.



FINDING 22: Secure Boot is Disabled – Compute Box

<table>
<tr><th colspan="3">Description</th></tr>
<tr><td>Type: Cryptographic Failures</td><td rowspan="2">The Picard/Bravo Compute Box was found to have 'Secure Boot' disabled. Secure Boot is a security feature that ensures only trusted software runs during the device's startup process.</td><td>CVSS Score: 9.8</td></tr>
<tr><td>Threat Context: Inexperienced Attacker</td><td>Severity: CRITICAL</td></tr>
<tr><td rowspan="2">Public Full Disclosure Date: 09/19/2025</td><td>Impact</td><td rowspan="2">CVE #: CVE-2025-59408</td></tr>
<tr><td>Disabling Secure Boot allows unsigned or malicious bootloaders and kernel-level code to execute during system startup, undermining the trust chain and enabling persistent compromise at the firmware or OS level. This exposes the host to rootkits and pre-boot tampering undetectable by standard security controls.</td></tr>
<tr><td colspan="2">CVSS 4.0 AV:P/AC:L/AT:N/PR:N/UI:N/VC:H/VI:H/VA:H/SC:N/SI:N/SA:N</td><td>CWE: CWE- 1326</td></tr>
<tr><td rowspan="2">Discovered By:
Jon Gaines</td><td>Affected Hardware/Software</td><td rowspan="2">Further Research Recommended: N</td></tr>
<tr><td>Picard/Bravo Compute Box</td></tr>
<tr><td colspan="3">Notes:</td></tr>
<tr><td colspan="3">Relevant Output:
(bootloader) secure:no</td></tr>
</table>

Steps to Reproduce:

1. Plug into the black USB-C of the device and push the button to turn it on.
2. Once the blue light appears use adb to boot into fastboot.
3. Use the following command to confirm that Secure Boot is off: <REDACTED_COMMAND>

Output:

```
(bootloader) parallel-download-flash:yes
(bootloader) hw-revision:10000
(bootloader) unlocked:yes
…
(bootloader) erase-block-size: 0x1000
…
(bootloader) secure:no
(bootloader) serialno:REDACTED
(bootloader) product:lahaina
…
all:
Finished. Total time: 0.012s
```

Tools Used:

- USB-C Cord
- adb
- fastboot

Mitigation: Enable Secure Boot

FINDING 23: Unlocked Bootloader – Compute Box

Description		
Type: Cryptographic Failures	The Picard/Bravo Compute Box's bootloader was found to be unlocked allowing unauthorized firmware to be installed.	**CVSS Score:** 9.8
Threat Context: Inexperienced Attacker		**Severity:** CRITICAL
Public Full Disclosure Date: 09/19/2025	**Impact**	**CVE #:** CVE-2025-59404
	An attacker with physical access can flash modified or malicious firmware onto the device trivially.	
CVSS 4.0 AV:P/AC:L/AT:N/PR:N/UI:N/VC:H/VI:H/VA:H/SC:N/SI:N/SA:N		**CWE:** CWE- 1299
Discovered By: Jon Gaines	**Affected Hardware/Software**	**Further Research Recommended:** N
	Picard/Bravo Compute Box	
Notes:		
Relevant Output: (bootloader) unlocked:yes		

Steps to Reproduce:

1. Plug into the black USB-C of the device and push the button to turn it on.
2. Once the blue light appears use adb to boot into fastboot.
3. Use the following command to confirm that the bootloader is unlocked: <REDACTED_COMMAND>

Output:

```
(bootloader) hw-revision:10000
(bootloader) unlocked:yes
(bootloader) off-mode-charge:0
(bootloader) charger-screen-enabled:0
…
(bootloader) serialno:REDACTED
(bootloader) product:lahaina
(bootloader) snapshot-update-status:none
(bootloader) is-userspace:no
(bootloader) max-download-size:805306368
(bootloader) kernel:uefi
all:
Finished. Total time: 0.012s
```

Tools Used:

- USB-C Cord
- adb
- fastboot

Mitigation: Lock Bootloader after installing firmware.

FINDING 24: Lack of Authentication: EDL/QDL Mode – Picard/Bravo Compute Box

<table>
<tr><th colspan="3">Description</th></tr>
<tr><td>Type: Broken Authentication/Authorization</td><td rowspan="2">The Picard/Bravo Compute Box EDL/QDL mode was found to lack any type of authentication or access control.</td><td>CVSS Score: 9.8</td></tr>
<tr><td>Threat Context: Inexperienced Attacker</td><td>Severity: CRITICAL</td></tr>
<tr><td rowspan="2">Public Full Disclosure Date: 09/19/2025</td><td>Impact</td><td rowspan="2">CVE #: CVE-2025-59402</td></tr>
<tr><td>An attacker with physical access can access device memory, firmware dumping, reading and flashing. In this case it results in a full compromise of the system's integrity.</td></tr>
<tr><td colspan="2">CVSS 4.0 AV:P/AC:L/AT:N/PR:N/UI:N/VC:H/VI:H/VA:H/SC:N/SI:N/SA:N</td><td>CWE: CWE- 1299</td></tr>
<tr><td rowspan="2">Discovered By:
Jon Gaines</td><td>Affected Hardware/Software</td><td rowspan="2">Further Research Recommended: N</td></tr>
<tr><td>Picard/Bravo Compute Box</td></tr>
<tr><td colspan="3">Notes:</td></tr>
<tr><td colspan="3">Relevant Output:</td></tr>
</table>

Steps to Reproduce:

4. Put the device in EDL mode by pressing the 'Force USB' button when turning it on or using a EDL flash cable.
5. Plug in a USB-C cord to its black USB-C port.
6. Note the default firehose for its chipset provided by the manufacturer works:

```
Qualcomm Sahara / Firehose Client V3.62 (c) B.Kerler 2018-2024.
main - Using loader prog_firehose_ddr.elf ...
main - Waiting for the device
main - Device detected :)
main - Mode detected: firehose

Parsing Lun 0:

GPT Table:
-------------
ssd:          Offset 0x0000000000006000, Length 0x0000000000002000, Flags 0x0000000000000000,
UUID 9bc13cdc-82e0-88d5-c693-103191f3d2a9, Type 0x2c86e742, Active False
persist:      Offset 0x0000000000008000, Length 0x0000000002000000, Flags 0x0000000000000000,
UUID 8902fc35-5b77-4647-e84b-8da793dff88c, Type 0x6c95e238, Active False
misc:         Offset 0x0000000002008000, Length 0x0000000000100000, Flags 0x0000000000000000,
UUID 6eb751a5-1ae1-1088-0027-860b563d12e5, Type 0x82acc91f, Active False
…
```

Tools Used:

- USB-C Cable
- edl

Mitigation: Implement a custom signed firehose that isn't publicly available.

FINDING 25: Lack of Authentication – Android Debug Bridge – Compute Box

Description		
Type: Cryptographic Failures	The Picard/Bravo Compute Box was found to not require authentication (approval) when accessing the device via Android Debug Bridge (ADB).	**CVSS Score:** 8.2
Threat Context: Inexperienced Attacker		**Severity:** HIGH
Public Full Disclosure Date: 09/19/2025	**Impact** An attacker with physical access can get ‘shell’ access to the device.	**CVE #:** N/A
CVSS 4.0 AV:P/AC:L/AT:N/PR:N/UI:N/VC:L/VI:H/VA:H/SC:N/SI:N/SA:N		**CWE:** CWE- 312
Discovered By: Jon Gaines	**Affected Hardware/Software** Picard/Bravo Compute Box	**Further Research Recommended:** N
Notes:		
Relevant Output:		

Steps to Reproduce:

1. Plug into the black USB-C of the device and push the button to turn it on.
2. Once the blue light appears connect to the device using ‘adb’ confirming that developer options and on-device approval/pre-shared ADB keys are not being utilized:

```
nigel@SectorTO ~ % adb shell
BRAVO:/ $ uname -a
Linux localhost 5.4.180-20220619-1-qgki-g72005ae422eb #1 SMP PREEMPT Thu Oct 17 02:34:13 CST 2024 aarch64 Toybox
BRAVO:/ $
```

Tools Used:

- USB-C Cord
- adb

Mitigation: Disable unauthenticated and unauthorized ADB access.

FINDING 26: Improper Access Control – Android Debug Bridge Sideload– Compute Box

Description		
Type: Improper Access Control	The Picard/Bravo Compute Box was found to allow sideloading apps via ADB.	**CVSS Score:** 8.2
Threat Context: Inexperienced Attacker		**Severity:** HIGH
Public Full Disclosure Date: 09/19/2025	**Impact** An attacker with physical access can install whatever application they want onto the device.	**CVE #:** N/A
CVSS 4.0 AV:P/AC:L/AT:N/PR:N/UI:N/VC:L/VI:H/VA:H/SC:N/SI:N/SA:N		**CWE:** CWE- 284
Discovered By: Jon Gaines	**Affected Hardware/Software** Picard/Bravo Compute Box	**Further Research Recommended:** N
Notes:		
Relevant Output:		

Steps to Reproduce:

1. Turn the device on and plug in a USB-C cable to its black USB-C port.
2. Use ‘adb install example.apk to sideload an app. Note it is successful.

Tools Used:
- USB-C Cable
- adb

Mitigation: Disable unauthenticated and unauthorized ADB access. Disallow sideloading via ADB.

FINDING 27: Lack of Flash/UFS Encryption – Compute Box

Description		
Type: Cryptographic Failures	The Picard/Bravo Compute Box was found to lack Flash/UFS encryption. This encryption ensures that if the firmware is dumped from the device, it is unreadable	**CVSS Score:** 7.8
Threat Context: Inexperienced Attacker		**Severity:** HIGH
Public Full Disclosure Date: 09/19/2025	**Impact** An attacker with physical access can read or dump the device's firmware in cleartext.	**CVE #:** N/A
CVSS 4.0 AV:P/AC:L/AT:N/PR:N/UI:N/VC:H/VI:L/VA:N/SC:N/SI:N/SA:N		**CWE:** CWE- 312
Discovered By: Jon Gaines	**Affected Hardware/Software** Picard/Bravo Compute Box	**Further Research Recommended:** N
Notes:		
Relevant Output:		

Steps to Reproduce:
1. Use a tool such as 'edl' to dump the firmware.
2. Note it is in clear text.

Tools Used:
- USB-C Cable
- edl

Mitigation: Enable full disk encryption, enable app-level encryption where possible.

MUTLI-DEVICE

FINDING 28: Unauthenticated Administrative API Endpoints

Description		
Type: Improper Access Control **Threat Context:** Inexperienced Attacker	The 'Collins' application used to run and manage the LPR image/video stream installed on multiple devices was found to contain a API web service that lacked any form of authentication or authorization.	**CVSS Score:** 9.8 **Severity:** CRITICAL
Public Full Disclosure Date: 09/27/2025	**Impact** An attacker with adjacent access can request sensitive information, perform a Denial of Service (DoS), enable wireless command access, enable or disable the camera feed and other sensitive operations In conjunction with other findings in this paper, it results in complete device compromise.	**CVE #:** CVE-2025-59403
CVSS 4.0 AV:P/AC:L/AT:N/PR:N/UI:N/VC:L/VI:H/VA:H/SC:N/SI:N/SA:N		**CWE:** CWE- 1299
Discovered By: Jon Gaines	**Affected Hardware/Software** Collins Application (com.flocksafety.android.collins) Picard/Bravo Compute Box & Falcon/Sparrow/Flex* LPR	**Further Research Recommended:** N
Notes:		
Relevant Output:		

Steps to Reproduce:

1. Below is a incomplete list of the Collins API Web Service Functionality:

```
PUT /api/v1/liveView/enable → activates JPEG or MPJEG streaming (startLiveView(true))
PUT /api/v1/liveView/disable → deactivates stream
PUT /api/v1/system/reboot → triggers reboot handler
PUT /api/v1/system/switch/enable → toggles internal system state
GET /api/v1/system/modem → modem stats
GET /api/v1/system/battery → battery info
GET /api/v1/system/os → build/version info
GET /api/v1/system/apps → app version report
GET /api/v1/system/logs → diagnostics dump
GET /api/v1/system/crashpack → crash report bundle
PUT /api/v1/system/battery/disable_internal → disables battery internally (likely shutdown governor)
PUT /api/v1/system/battery/shutdown_delay → modifies auto-shutdown delay
REDACTED → enables adb over TCP without adb authentication (Remote Control)
```

2. The following screenshots demonstrate that the 'collins' application listens on all interfaces:

```
130|msm8953_32:/media # netstat -tulpn
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program Name
tcp        0      0 127.0.0.1:53            0.0.0.0:*               LISTEN      2892/dnsmasq
tcp        0      0 192.168.43.1:53         0.0.0.0:*               LISTEN      2892/dnsmasq
tcp6       0      0 ::ffff:192.168.43.:1040 :::*                    LISTEN      1978/com.flocksafety.android.collins
tcp6       0      0 :::8080                 :::*                    LISTEN      1978/com.flocksafety.android.collins
tcp6       0      0 :::1234                 :::*                    LISTEN      1978/com.flocksafety.android.collins
tcp6       0      0 :::5555                 :::*                    LISTEN      3856/adbd
tcp6       0      0 ::1:53                  :::*                    LISTEN      2892/dnsmasq
tcp6       0      0 fe80::764c:a1ff:fe7e:53 :::*                    LISTEN      2892/dnsmasq
tcp6       0     56 ::ffff:192.168.43.:5555 ::ffff:192.168.43:64748 ESTABLISHED 3856/adbd
tcp6      1372686040 ::ffff:192.168.43.:1234 ::ffff:192.168.43:64760 ESTABLISHED 1978/com.flocksafety.android.collins
udp        0      0 127.0.0.1:53            0.0.0.0:*                           2892/dnsmasq
udp        0      0 192.168.43.1:53         0.0.0.0:*                           2892/dnsmasq
udp        0      0 0.0.0.0:67              0.0.0.0:*                           2892/dnsmasq
udp6       0      0 ::1:53                  :::*                                2892/dnsmasq
udp6       0      0 fe80::764c:a1ff:fe7e:53 :::*                                2892/dnsmasq
```

Tools Used:

- Wireless NIC

Mitigation: Implement Authentication and Authorization. Listen on loopback, disable especially sensitive endpoints.

FINDING 29: Hidden Hardware Debug Functionality – Hotspot

<table>
<tr><th colspan="3">Description</th></tr>
<tr><td>Type: Debug Functionality</td><td rowspan="2">The Picard/Bravo/Falcon/Flex* LPR and Compute Box were found to contain hidden debug functionality. In this case, by pressing the button on any of the devices 3 times in quick succession, the device's hotspot is enabled. Furthermore, by default all device's weak default hotspot passwords are '<REDACTED_WEAK_PASSWORD>.'</td><td>CVSS Score: 9.8</td></tr>
<tr><td>Threat Context: Inexperienced Attacker</td><td>Severity: CRITICAL</td></tr>
<tr><td rowspan="2">Public Full Disclosure Date: 09/27/2025</td><td>Impact</td><td rowspan="2">CVE #: N/A</td></tr>
<tr><td>An attacker with brief physical access can enable the devices to enable their hotspot and then wirelessly connect to them. Chained with other vulnerabilities it greatly increases the risk.</td></tr>
<tr><td colspan="2">CVSS 4.0 AV:P/AC:L/AT:N/PR:N/UI:N/VC:H/VI:H/VA:H/SC:H/SI:H/SA:H</td><td>CWE: CWE- 78</td></tr>
<tr><td rowspan="2">Discovered By:
Kajer (Button Press Sequence)
Jon Gaines</td><td>Affected Hardware/Software</td><td rowspan="2">Further Research Recommended: N</td></tr>
<tr><td>Collins Application (com.flocksafety.android.collins)
Picard/Bravo Compute Box & Falcon/Sparrow/Flex* LPR</td></tr>
<tr><td colspan="3">Notes:</td></tr>
<tr><td colspan="3">Relevant Output:</td></tr>
</table>

Steps to Reproduce:

1. Perform the basic button press sequence to enable the device's Hotspot.
2. Wait for a Flock-* SSID to appear.
3. Connect to it using the weak hardcoded hotspot password '<REDACTED_HARDCODED_WEAK_PASSWORD>.'

Tools Used:

- Wireless NIC

Mitigation: Do not include debug functionality in production deployments.

FINDING 30: Wireless Remote Code Execution (RCE) – System*

Description		
Type: Code Execution	The Falcon/Sparrow/Flex* LPR and Picard/Bravo Compute Box were found to enable the chaining of multiple vulnerabilities disclosed in this paper together resulting in wireless control of devices with system permissions.	**CVSS Score:** 9.8
Threat Context: Experienced Attacker		**Severity:** CRITICAL
Public Full Disclosure Date: 01/23/2026	**Impact**	**CVE #:** PENDING
	An attacker with adjacent access can leverage unauthenticated API requests to enable and then connect to the device wirelessly. Additionally, since the Android applications are installed with debugging enabled, an attacker can leverage that access to execute commands as system.	
CVSS 4.0 AV:A/AC:L/AT:P/PR:N/UI:N/VC:H/VI:H/VA:H/SC:H/SI:H/SA:H		**CWE:** CWE- 78
***Discovered By:**	**Affected Hardware/Software**	**Further Research Recommended:** N
Jon Gaines (System Injection by JosephRC)	Collins Application (com.flocksafety.android.collins) Picard/Bravo Compute Box & Falcon/Sparrow/Flex* LPR	
Notes:		
Relevant Output:		

Steps to Reproduce:

1. Send the following 'PUT' HTTP request when on the same W/LAN of the deivce to enable ADB over TCP without authentication: <REDACTED_COMMAND>
2. Use adb to wirelessly connect to device as the 'shell' user.
3. Use debug access along with a 'trigger' to execute commands as system.

Tools Used:

- Wireless NIC
- adb

Mitigation: Implement Authentication and Authorization. Do not ship application with debug enabled in production.

FINDING 31: Incorrect Default Permissions – Media Recordings Directories

Description		
Type: Data Policy Failure	The Falcon/Sparrow/Flex* LPR and Picard/Bravo Compute Box underlying recording app suite was found to store media recording it takes and processes in directories with insecure permissions. In this case, the <REDACTED_MEDIA_DIRECTORY> and <REDACTED_MEDIA_DIRECTORY> were found to have overly permissive access control permissions (0774).	**CVSS Score:** 9.8
Threat Context: Experienced Attacker		**Severity:** CRITICAL
Public Full Disclosure Date: 02/11/2026	**Impact**	**CVE #:** PENDING
	An attacker with shell or physical access to a unit can mount or view the adoptable partition and read every stage of the recording lifecycle from ‘capturing’ to ‘encoded’ in cleartext.	
CVSS 4.0 AV:A/AC:L/AT:P/PR:N/UI:N/VC:H/VI:H/VA:H/SC:H/SI:H/SA:H		**CWE:** CWE-922
Discovered By: Jon Gaines	**Affected Hardware/Software**	**Further Research Recommended:** N
	Flock Safety Recording App Suite: com.flocksafety.android.videorecording, com.flocksafety.android.motion, com.flocksafety.android.objects, com.flocksafety.android.encoding, com.flocksafety.android.cameraconfig, com.flocksafety.android.collins, com.flocksafety.android.streaming Picard/Bravo Compute Box & Falcon/Sparrow/Flex* LPR	
Notes:		
Relevant Output:		

Steps to Reproduce:

1. Obtain MediaFileUtil.java and view the following method which sets the insecure permissions: <REDACTED_METHOD>
2. 2. On hardware, run `adb shell ls -ld <REDACTED_MEDIA_DIRECTORY> and note the drwxrwxr-- mode.
3. From a secondary process sharing `media_rw` group membership (<REDACTED_COMMAND>), open any file inside `captured/` or `encoded/`; read succeeds due to the overly broad ACLs.

Tools Used:

- Wireless NIC
- adb

Mitigation: Tighten ACLs, isolate groups, local SELinux contexts per stage, mount the media volume with ‘nodev,nosuid,noexec’.

FINDING 32: Shared Media Library Allows Cross App Data Exposure

Description.		
Type: Insecure Design	The Flock Safety Recording App Suite (including at least seven Flock Safety Custom APKs) used by the Falcon/Sparrow/Flex** LPRs and Picard/Bravo Compute Box was found to embed the identical MediaFileUtil and MediaSession code, mounting the same adoptable path; a privilege escalation in any non-recording app immediately exposes the entire media library, dramatically expanding the blast radius of otherwise isolated components.	**CVSS Score:** 8.8
Threat Context: Inexperienced Attacker		**Severity:** HIGH
Public Full Disclosure Date: 02/11/2026	**Impact** An attacker who compromises any auxiliary app (installer, live view, streaming) inherits full read/write access to the recording tree because every package bundles the same storage helper bound to <REDACTED_MEDIA_PATH> or <REDACTED_MEDIA_PATH>.	**CVE #:** PENDING
CVSS 4.0 AV:A/AC:L/AT:P/PR:N/UI:N/VC:H/VI:H/VA:H/SC:N/SI:N/SA:N		**CWE:** CWE- 925
Discovered By: Jon Gaines	**Affected Hardware/Software** Flock Safety Recording App Suite: com.flocksafety.android.videorecording, com.flocksafety.android.motion, com.flocksafety.android.objects, com.flocksafety.android.encoding, com.flocksafety.android.cameraconfig, com.flocksafety.android.collins, com.flocksafety.android.streaming Picard/Bravo Compute Box & Falcon/Sparrow/Flex* LPR	**Further Research Recommended:** N
Notes:		
Relevant Output:		

Steps to Reproduce:

1. Run adb shell dumpsys package com.flocksafety.android.* | grep versionName to confirm all seven packages are installed and share the system UID.

2. Decompile `flock-collins.apk` or `flock-video-streaming.apk` and inspect MediaFileUtil. Note it resolves paths under <REDACTED_MEDIA_PATH> or <REDACTED_MEDIA_PATH>.3.

Repeat for `flock-cameraconfig.apk` to show unrelated apps ship the same media helpers, proving a compromise in anyone yields full read/write access to the shared recording tree.

Tools Used:

- adb

Mitigation: Isolate per app storage roots or SELinux labels, restrict MediaFileUtil access to the owning service, and enforce ACLs that prevent unrelated packages from reading/writing the recording tree.

FINDING 33: Wireless Remote Code Execution (RCE) - Shell

Description		
Type: Code Execution	The Falcon/Sparrow/Flex* LPR and Picard/Bravo Compute Box were found to enable the chaining of multiple vulnerabilities disclosed in this paper together resulting in wireless control of devices.	**CVSS Score:** 8.8
Threat Context: Inexperienced Attacker		**Severity:** HIGH
Public Full Disclosure Date: 09/27/2025	**Impact** An attacker with adjacent access can leverage unauthenticated API request to enable and then connect to the device wirelessly.	**CVE #:** CVE-2025-59403

CVSS 4.0 AV:A/AC:L/AT:P/PR:N/UI:N/VC:H/VI:H/VA:H/SC:N/SI:N/SA:N		**CWE:** CWE- 78
Discovered By: Jon Gaines	**Affected Hardware/Software** Collins Application (com.flocksafety.android.collins) Picard/Bravo Compute Box & Falcon/Sparrow/Flex* LPR	**Further Research Recommended:** N
Notes:		
Relevant Output:		

Steps to Reproduce:

1. Send the following 'PUT' HTTP request when on the same W/LAN of the deivce to enable ADB over TCP without authentication: <REDACTED_COMMAND>
2. Use adb to wirelessly connect to device as the 'shell' user.

Tools Used:

- Wireless NIC
- adb

Mitigation: Implement Authentication and Authorization. Do not ship application with debug enabled in production. Additionally, follow migrations of the other findings.

FINDING 34: Unauthenticated Debug Broadcast Clears Settings and Shuts off Device

Description		
Type: Insufficient Input Validation	The PhoneHomeService Application registers a system-wide debug broadcast with no permission gate; sending type=update with metadata.type=clear drives clearSettingsAndPowerOff, wiping camera settings and issuing a privileged shutdown.	**CVSS Score:** 8.2
Threat Context: Experienced Attacker		**Severity:** HIGH
Public Full Disclosure Date: 01/23/2026	**Impact** An attacker with physical, local access or a malicious installed application can issue a broadcast that will wipe and shut off the device.	**CVE #:** PENDING
CVSS 4.0 AV:A/AC:L/AT:P/PR:N/UI:N/VC:L/VI:H/VA:H/SC:N/SI:N/SA:N		**CWE:** CWE- 925
Discovered By: Jon Gaines	**Affected Hardware/Software** Phone Home Service Application(com.flocksafety.android.phonehomeservice) Picard/Bravo Compute Box & Falcon/Sparrow/Flex* LPR	**Further Research Recommended:** N
Notes:		
Relevant Output:		

Steps to Reproduce:

1. Use the following command after connecting to the device: <REDACTED_COMMAND>
2. Observe logcat or device behavior, within seconds the service wipes camera settings and calls ‘PowerHandler.shutDownWithReason’, causing an enforced shutdown.

Tools Used:

- adb

Mitigation: Remove the debug receiver from production builds or gate it behind a signature-level permission and strict caller validation before invoking clearSettingsAndPowerOff.

FINDING 35: Multiple Privileged System Apps Shipped with Debugging Enabled

<table>
<tr><th colspan="3">Description</th></tr>
<tr><td>Type: Insecure Design</td><td rowspan="2">The Falcon/Sparrow/Flex* LPR and Picard/Bravo Compute Box were found to be deployed with a custom Android application suite all of which had debugging enabled.</td><td>CVSS Score: 7.9</td></tr>
<tr><td>Threat Context: Inexperienced Attacker</td><td>Severity: HIGH</td></tr>
<tr><td rowspan="2">Public Full Disclosure Date: 09/27/2025</td><td>Impact</td><td rowspan="2">CVE #: N/A</td></tr>
<tr><td>An attacker with shell access can tamper with the application during runtime. Additionally, this issue was leveraged in other findings.</td></tr>
<tr><td colspan="2">CVSS 4.0 AV:L/AC:L/AT:N/PR:L/UI:N/VC:H/VI:H/VA:H/SC:N/SI:N/SA:N</td><td>CWE: CWE- 925</td></tr>
<tr><td rowspan="2">Discovered By:
Jon Gaines</td><td>Affected Hardware/Software</td><td rowspan="2">Further Research Recommended: N</td></tr>
<tr><td>Multiple Applications
Picard/Bravo Compute Box & Falcon/Sparrow/Flex* LPR</td></tr>
<tr><td colspan="3">Notes:</td></tr>
<tr><td colspan="3">Relevant Output:</td></tr>
</table>

Steps to Reproduce:

1. Use the following command to confirm the application is debuggable: <REDACTED_COMMAND>
2. Prep the application for debugging: <REDACTED_COMMAND>
3. Attach with a debugger to inspect and tamper with the app in a privileged state.

Tools Used:

- adb

Mitigation: Build production application with the 'android:debuggable="false"' property.

FINDING 36: Lack of Per File Encryption on Sensitive Media

Description		
Type: Cryptographic Failures	The Flock Safety Recording App Suite used by the Falcon/Sparrow/Flex** LPRs and Picard/Bravo Compute Box were found to utilize services with insecure run time data policies. Specifically the Capture, motion, ML, and encoding services persist all intermediates and finals directly onto the adoptable <REDACTED_MEDIA_DIRECTORY> tree without any per file encryption; once <REDACTED_PROP> completes and the LUKS volume is mounted, every JPEG/YUV/MP4 remains readable, exposing raw evidence to anyone who can access the partition.	**CVSS Score:** 7.9
Threat Context: Inexperienced Attacker		**Severity:** HIGH
Public Full Disclosure Date: 02/11/2026	**Impact**	**CVE #:** PENDING
	An attacker with shell or physical access to an unit can mount the adoptable partition and read every stage from ‘capturing’ to ‘encoded’ in cleartext.	
CVSS 4.0 AV:L/AC:L/AT:N/PR:L/UI:N/VC:H/VI:H/VA:H/SC:N/SI:N/SA:N		**CWE:** CWE- 925
Discovered By: Jon Gaines	**Affected Hardware/Software**	**Further Research Recommended:** N
	Flock Safety Recording App Suite: com.flocksafety.android.videorecording, com.flocksafety.android.motion, com.flocksafety.android.objects, com.flocksafety.android.encoding, com.flocksafety.android.cameraconfig, com.flocksafety.android.collins, com.flocksafety.android.streaming Picard/Bravo Compute Box & Falcon/Sparrow/Flex* LPR	
Notes:		
Relevant Output:		

Steps to Reproduce:

1. Access any of the affected devices and wait for getprop <REDACTED_PROP> to return `true`.
2. Execute `adb shell ls -R <REDACTED_MEDIA_PATH>` and note the plain JPEG/YUV/MP4 files under `captured/`, `motionProcessed/`, `detectionProcessed/`, and `encoded/`.
3. 4. Pull any file with `adb pull <REDACTED_MEDIA_PATH>/<file>.mp4` and open it locally confirming no keys or decrypt step required.

Tools Used:

- adb

Mitigation: Implement per file envelope encryption (AES-GCM) using TEE/HSM derived KEKs, rotate DEKs per session, gate every writer on “encrypted and healthy” volume state, encrypt crashpacks and staging folders.

FINDING 37: Sensitive Information Disclosed – Hardcoded Auth0 Secret

Description		
Type: Sensitive Data Disclosure	The Falcon/Sparrow & Picard/Bravo Compute Box were found to use custom Android apps across devices. In this case, the ‘Pisco’ application installed on multiple devices was found to hardcode a static Auth0 client secret as well as store the Auth0 token and JWT in cleartext.	**CVSS Score:** 6.6
Threat Context: Inexperienced Attacker		**Severity:** MEDIUM
Public Full Disclosure Date: 09/27/2025	**Impact**	**CVE #:** CVE-2025-59406
	An attacker with local access can dump the APKs and extract the hardcoded sensitive information from their APKs.	
CVSS 4.0 AV:L/AC:L/AT:N/PR:N/UI:N/VC:H/VI:N/VA:N/SC:N/SI:N/SA:N		**CWE:** CWE-319

Discovered By:	Affected Hardware/Software	Further Research Recommended: N
Jon Gaines	Pisco (com.flocksafety.android.pisco) Android Application Picard/Bravo Compute Box & Falcon/Falcon/Flex* LPR	
Notes:		
The severity of this finding has been significantly reduced as per the scope, testing the validity of the Auth0_client, secret, JWT and token was not performed.		
Relevant Output:		

Steps to Reproduce:

▼ Showing all **5** secrets

"auth0_client_secret" : "[redacted]"

Tools Used:

- Micro USB Cord/USB-C Cord
- adb
- cat

Mitigation: Do not hardcode static sensitive information across devices. Utilize hashing and encryption where applicable

FINDING 38: Root Command Injection via Data Log Cleanup Service

Description		
Type: Insufficient Input Validation **Threat Context:** Experienced Attacker	The ‘SystemControlService service was found to be vulnerable to command injection that is executed with root privileges. In this case, one or more properties are used within the execution of the <REDACTED_RC_NAME> and its bash script without input validation. It can also be triggered manually by modifying <REDACTED_PROP_NAME_3> value.	**CVSS Score:** 5.4 **Severity:** MEDIUM
Public Full Disclosure Date: 01/23/2025	**Impact** An attacker with system level permissions can insert a specifically crafted payload within a specific property that results in root command execution. This results in full device compromise. Additionally an attacker with control of another application within the ‘Flock’ SELinux context can also trigger this vulnerability.	**CVE #:** PENDING
CVSS 4.0 AV:A/AC:H/AT:P/PR:N/UI:N/VC:H/VI:L/VA:N/SC:N/SI:N/SA:N		**CWE:** CWE- 78
Discovered By: Jon Gaines	**Affected Hardware/Software** DataLog Cleanup Service (flock.clean_data_partition.sh) Picard/Bravo Compute Box & Falcon/Falcon/Flex* LPR	**Further Research Recommended:** N
Notes: By default, the Selinux Policy prevents the root commands from being executed, therefore reducing the severity significantly. However, the underlying vulnerability is still there. It is unclear if any of the paths to root, such as the data log cleanup service is used in units currently deployed in the wild.		
Relevant Output:		

Steps to Reproduce:

1. Set the <REDACTED_PROP_NAME_1> or <REDACTED_PROP_NAME_2> value to a specifically crafted payload using system command injection disclosed in previous findings.
2. This prop’s value is inserted directly into a bash script automatically that is used by the Data Log Cleanup service that is always executed as root.
3. Wait for the cleanup service to run for the payload to be executed or set the <REDACTED_PROP_NAME> to 1 using the system command injection to have the service run immediately.

Tools Used:

- adb

Mitigation: Implement Authentication and Authorization. Do not ship application with debug enabled in production. Validate and sanitize user input before inserting it into anything that is executed, especially as root.

FINDING 39: Excessive Sensitive Media Copies Persist on Disk

Description		
Type: Data Policy Failure	The Flock Safety Recording App Suite used by the Falcon/Sparrow/Flex** LPRs and Picard/Bravo Compute Box was found to serialize every session through up to seven directory hops (\`capturing/\`, \`captured/\`, \`motionProcessed/\`, \`detectionProcessed/\`, \`encodedStaging/\`, \`encoded/\`, \`discarded/\`, plus \`crashpack/\` spillover), creating numerous long-lived copies of the same evidence; absent prompt deletion, the expanded footprint makes local exfiltration trivial even if one directory is cleaned.	**CVSS Score:** 5.4
Threat Context: Inexperienced Attacker		**Severity:** MEDIUM
Public Full Disclosure Date: 02/11/2026	**Impact**	**CVE #:** PENDING
	An attacker with access to the adoptable partition can harvest multiple redundant copies (raw frames, motion-filtered sets, ML outputs, staging encodes, crashpacks) that persist until manual purge, greatly increasing the available data set for exfiltration.	
CVSS 4.0 AV:A/AC:H/AT:P/PR:N/UI:N/VC:H/VI:L/VA:N/SC:N/SI:N/SA:N		**CWE:** CWE- 925
Discovered By: Jon Gaines	**Affected Hardware/Software**	**Further Research Recommended:** Y
	Flock Safety Recording App Suite: com.flocksafety.android.videorecording, com.flocksafety.android.motion, com.flocksafety.android.objects, com.flocksafety.android.encoding, com.flocksafety.android.cameraconfig, com.flocksafety.android.collins, com.flocksafety.android.streaming Picard/Bravo Compute Box & Falcon/Sparrow/Flex* LPR	
Notes:		
Relevant Output:		

Steps to Reproduce:

1. Trigger a recording and allow motion, ML, and encoding services to process it.
2. Run <REDACTED_COMMAND> and note the same session identifier present under \`captured/\`, \`motionProcessed/\`, \`detectionProcessed/\`, \`encodedStaging/\`, \`encoded/\`, \`discarded/\`, and \`crashpack/\`.
3. Confirm files exist in each directory even after the final encode is completed, demonstrating redundant plaintext copies.

Tools Used:

- adb

Mitigation: Collapse intermediates where possible, encrypt crashpacks and transient folders, ensure deleteSessionFilesFromAllDirs() executes on every success/failure path, disable ML streaming outputs to world readable trees, integrity tag artifacts, and enforce strict retention windows with verified purge.

FINDING 40: Sensitive Information Disclosed – Cleartext API Keys/Credentials

Description		
Type: Sensitive Data Disclosure	The Falcon/Sparrow/Flex* LPR and Picard/Bravo Compute Box were found to use custom Android apps across devices. In this case, there are multiple instances of hardcoded and clear text sensitive information, including but not limited to API keys and credentials.	**CVSS Score:** 6.6
Threat Context: Experienced Attacker		**Severity:** MEDIUM
	Impact	

Public Full Disclosure Date: 06/19/2025	An attacker with local access can dump the APKs and extract the hardcoded sensitive information from their APKs.	**CVE #:** CVE-2025-47823
CVSS 4.0 AV:L/AC:L/AT:N/PR:N/UI:N/VC:H/VI:N/VA:N/SC:N/SI:N/SA:N		**CWE:** CWE- 798
Discovered By: Jon Gaines	**Affected Hardware/Software** **Multiple Applications** Picard/Bravo Compute Box & Falcon/Falcon/Flex* LPR & Raven Gunshot Detection System	**Further Research Recommended:** N
Notes: This finding was improperly included in CVE-2025-47823 instead of being given its own CVE # when the Vendor submitted the CVE assignment request.		
Relevant Output:		

Steps to Reproduce:

Affected File 1: <REDACTED_FILE_PATH>/CameraSettings.java

Code Snippet:

```
static {
        CameraSettings cameraSettings = new CameraSettings();
        INSTANCE = cameraSettings;
        defaultCoreValues = new CoreValues("cereal", CoreValues.DEFAULT_API_KEY, "https://dev-gimlet.flocksafety.com/", "https://dev-gimlet.flocksafety.com/", null, "", "", 16, null);
        SETTINGS_URI = Uri.parse("content://com.flocksafety.android.settingsservice.provider/settings");
        CORE_VALUES_URI =
Uri.parse("content://com.flocksafety.android.settingsservice.provider/core_values");
        logger = new TimberTagWrapper(cameraSettings.getClass());
    }
    public final String getHpnotiqApiKey() {
        return "<REDACTED_API_KEY>";
```

Affected File 2: <REDACTED_FILE_PATH>/CoreValues.java

```
/* loaded from: classes.dex */
public final /* data */ class CoreValues {
    public static final String DEFAULT_API_KEY = "<REDACTED_API_KEY>";
    public static final String TABLE_NAME = "core_values";
    private final String authToken;
    private final String mediaInfoUrl;
    private final String partNumber;
    private final String serialNumber;
    private final String statusUrl;
    private final Date updatedAt;
    private final String uploadUrl;
```

Affected File 3: <REDACTED_FILE_PATH>/SSL.java

```
/* loaded from: classes.dex */
public interface SSL {
    public static final String DEFAULT_KEYSTORE_PASSWORD =
"<REDACTED_KEYSTORE_PASSWORD>";
    public static final String DEFAULT_KEYSTORE_TYPE = "JKS";
    public static final String DEFAULT_PROTOCOL = "SSL";
    public static final String DEFAULT_SECURE_RANDOM_ALGORITHM = "SHA1PRNG";
}
```

Affected File 4: wpa_supplicant.conf

```
# WPA pre-shared keys for WPA-PSK. This can be either entered as a 256-bit
# secret in hex format (64 hex digits), wpa_psk, or as an ASCII passphrase
# (8..63 characters) that will be converted to PSK. This conversion uses SSID
# so the PSK changes when ASCII passphrase is used and the SSID is changed.
# wpa_psk (dot11RSNAConfigPSKValue)
# wpa_passphrase (dot11RSNAConfigPSKPassPhrase)
#wpa_psk=01234567[redacted]abcdef
wpa_passphrase=s[redacted]y

# Optionally, WPA PSKs can be read from a separate text file (containing list
# of (PSK,MAC address) pairs. This allows more than one PSK to be configured.
```

Tools Used:

- Micro USB Cord/USB-C Cord
- adb

Mitigation

Do not hardcode sensitive information, do not reuse API Keys or credenitals across installs and devices. Always implement hashing or encryption.

FINDING 41: Wireless Remote Code Execution (RCE) - Root

<table>
<tr><th colspan="3">Description</th></tr>
<tr><td>Type: Code Execution</td><td rowspan="2">The Falcon/Sparrow/Flex* LPR and Picard/Bravo Compute Box were found to enable the chaining of multiple vulnerabilities disclosed in this paper together resulting in wireless control of devices with root permissions.</td><td>CVSS Score: 5.4</td></tr>
<tr><td>Threat Context:
Experienced Attacker</td><td>Severity: MEDIUM</td></tr>
<tr><td rowspan="2">Public Full Disclosure Date: 01/23/2026</td><td>Impact</td><td rowspan="2">CVE #: PENDING</td></tr>
<tr><td>An attacker with adjacent access can leverage unauthenticated API requests to enable and then connect to the device wirelessly. Additionally, since the Android applications are installed with debugging enabled, an attacker can leverage that access to execute commands as root.</td></tr>
<tr><td colspan="2">CVSS 4.0 AV:A/AC:H/AT:P/PR:N/UI:N/VC:H/VI:L/VA:N/SC:N/SI:N/SA:N</td><td>CWE: CWE- 78</td></tr>
<tr><td rowspan="2">Discovered By:
Jon Gaines</td><td>Affected Hardware/Software</td><td rowspan="2">Further Research Recommended: N</td></tr>
<tr><td>Collins Application (com.flocksafety.android.collins)
DataLog Cleanup Service
Picard/Bravo Compute Box & Falcon/Falcon/Flex* LPR</td></tr>
<tr><th colspan="3">Notes:</th></tr>
<tr><td colspan="3">By default, the Selinux Policy prevents the root commands from being executed, therefore reducing the severity significantly. However, the underlying vulnerability is still there. It is unclear if any of the paths to root, such as the data log cleanup service is used in units currently deployed in the wild.</td></tr>
<tr><th colspan="3">Relevant Output:</th></tr>
</table>

Steps to Reproduce:

1. Send the following 'PUT' HTTP request when on the same W/LAN of the deivce to enable ADB over TCP without authentication: <REDACTED_COMMAND>
2. Use adb to wirelessly connect to device as the 'shell' user.
3. Leverage one of the paths for root command injection, such as via the Data-Log Cleanup service by injection a specially crafted payload into it's property.

Tools Used:

- Wireless NIC

Mitigation: Implement Authentication and Authorization. Do not ship application with debug enabled in production. Validate and sanitize user input before inserting it into anything that is executed, especially as root.

FINDING 42: ML/AI Local Model Accessible

Description		
Type: Sensitive Data Disclosure	The Falcon/Sparrow/Flex* LPR and Picard/Bravo Compute Box store their AI/ML local inference modules in cleartext, leaving the models fully exposed.	**CVSS Score:** 5.4
Threat Context: Experienced Attacker		**Severity:** MEDIUM
Public Full Disclosure Date: N/A	**Impact**	**CVE #:** N/A
	Plaintext AI/ML binaries let any local or remote foothold copy, reverse, or tamper with inference logic, enabling model plagiarism, rapid bypass of decision thresholds, targeted poisoning of detections, and seamless chaining into the already-documented vulnerabilities.	
CVSS 4.0 AV:A/AC:H/AT:P/PR:N/UI:N/VC:H/VI:L/VA:N/SC:N/SI:N/SA:N		**CWE:** CWE-1299
*Discovered By: Unknown	**Affected Hardware/Software**	**Further Research Recommended:** N
	DetectionProcessing (com.flocksafety.android.objects) Android Application Picard/Bravo Compute Box & Falcon/Falcon/Flex* LPR	
Notes:		
I'm only aware that the Vendor has been told that the models are accessible. I am unsure who originally discovered them and disclosed them to the Vendor.		
Relevant Output:		

Steps to Reproduce:

1. Attach to any affected unit with a existing shell or obtain a copy of its filesystem.
4. List the adoptable media tree to confirm cleartext ML payloads using the following command: <REDACTED_COMMAND>.
5. Pull or Extract the NativeML artifacts using the following command: <REDACTED_COMMAND>
6. Repeat for bundles inside each installed APK.
7. Confirm the existence of models.json, label_map*.json, and every *.tflite, demonstrating ML model access.

Tools Used:

- Wireless NIC
- edl
- USB-C/Micro USB Cord

Mitigation: Implement Encryption.

FINDING 43: Sensitive Information Disclosed – Hardcoded Java Keystore & Password

Description		
Type: Sensitive Data Disclosure	The Falcon/Sparrow & Picard/Bravo Compute Box were found to use custom Android apps across devices. In this case, the 'Flock DetectionProcessing application was found to contain a cleartext password for a Java Keystore. This keystore contains the mutual TLS (mTLS) certificate the device uses when communicating with the cloud infrastructure.	**CVSS Score:** 3.2
Threat Context: Inexperienced Attacker		**Severity:** LOW
Public Full Disclosure Date: 09/27/2025	**Impact**	**CVE #:** CVE-2025-59407
	An attacker with local access can dump the APKs and extract the hardcoded sensitive information from their APKs.	
CVSS 4.0 AV:L/AC:H/AT:N/PR:N/UI:N/VC:H/VI:N/VA:N/SC:N/SI:N/SA:N		**CWE:** CWE- 1299

Discovered By:	Affected Hardware/Software	Further Research Recommended: N
Jon Gaines	DetectionProcessing (com.flocksafety.android.objects) Android Application Picard/Bravo Compute Box & Falcon/Falcon/Flex* LPR	
Notes:		
The severity of this finding has been significantly reduced as per the scope, testing of the validity of this mTLS certificate was not performed.		
Relevant Output:		

Steps to Reproduce:

Affected File 1: ‘<REDACTED_FILE_PATH_AND_FILE>ConnectionClient’

```
        }
    }

    private final SSLContext getSSLContext() {
        Resources resource = this.context.getResources();
        int resourceId = resource.getIdentifier("flock_rye", "raw", this.context.getPackageName());
        InputStream resourceStream = resource.openRawResource(resourceId);
        SSLContext createServerSSLContext = SSLUtil.createServerSSLContext(resourceStream, "flockhibiki17");
        Intrinsics.checkNotNullExpressionValue(createServerSSLContext, "createServerSSLContext(...)");
        return createServerSSLContext;
    }

    /* compiled from: ConnectionClient.kt */
```

1. The keystore is stored within the ‘DetectionProcessing’ Application.
2. You can then download the proper bouncycastle library and extract the ‘cert.pem’ from the keystore using that hardcoded password.

```
bject/res/raw$ keytool -exportcert  -alias selfsigned  -keystore flock_rye.bks  -storetype BKS  -providerclass org.b
ouncycastle.jce.provider.BouncyCastleProvider  -providerpath bcprov.jar  -file cert.pem
Enter keystore password:
Certificate stored in file <cert.pem>
```

3. Import the ‘cert.pem.’

```
bject/res/raw$ keytool  -importkeystore  -srckeystore flock_rye.bks  -srcstoretype BKS  -srcstorepass flockhibiki17
 -srcalias selfsigned  -destkeystore output.p12  -deststoretype PKCS12  -deststorepass flockhibiki17  -providerclas
s org.bouncycastle.jce.provider.BouncyCastleProvider  -providerpath bcprov.jar
Importing keystore flock_rye.bks to output.p12...
```

4. Use ‘openssl’ to extract the private key.

```
bject/res/raw$ openssl pkcs12 -in output.p12 -nocerts -nodes -out key.pem
Enter Import Password:
```

5. Confirm it’s validity by using ‘cat’ or similar:

```
bject/res/raw$ cat key.pem
Bag Attributes
    friendlyName: selfsigned
    localKeyID: 54 69 6D 65 20 31 37 35 30 37 33 37 36 37 36 37 36 38
Key Attributes: <No Attributes>
-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQDKUKeYBmFiq+T/
ekoPtDR+VTnqyD7RWc0UeiP19GDn2bhF0927ZStsCbX1b7Ua8b/L9+xh/Hyah/i0
j6M1YzZ1iApnkhtOQ7OSMELTxmYfff33oWW9vwQ3kPkJAC+vf1xK7yicX7[illegible]
otQf8bvlGHNWBYBYvTX7zHfMqXygKrjvjSPeNC/I/k6AVEa9VHfckuSkx3jJs3fe
y2hlmLCL1xVJZwhMsHS7O5rz+22WfSiXJFONZMh1nkqMUHLl4XXBISO3Fs1R/57d
k4y43aItt15VuB/V9Uht3xoiyBIoNAf+y+keMvnoI/U+TTs/bmCAB9cg9pIk+Jtf
q4IFPFYTAgMBAAECggEANFJ5lWqJAxC7j4rb3oXXUZjO+rZA8LmNvS79X92Mtklr
S7vIS2+VtEIqqcfzCX3eAW0j2A+JVhm2K1Q+fE8lRW4iarixkSMXLMaIceaMgLZ/
2OwHuPPOrmQy064VpwiEIOkLEq36tNtG94M84uoMkuoo4eVYVRrR/EXb8bUfgvgz
rDfyD7ftsHWmuEtkvjDYj/C+Xx7mcYc7zR9LtnWKWVwg8JG5miE1u+puAA9jcmaP
qUzXY7dPAOV3h3ObEks1fzSR1c2oA5I+M5AbJhAVLUFe0oa/88iSuR/BI+vekAmk
```

Tools Used:

- Micro USB Cord/USB-C Cord
- adb
- Bouncycastle
- openssl
- cat

- keytool

Mitigation: Do not use the same mTLS certifcates/private keys across devices and installations. Rotate them out.

FINDING 44: Data Recording retention relies solely on Disk Capacity

Description		
Type: Data Policy Failure	The only default automatic deletion policy prunes oldest files when disk usage exceeds TARGET_DISK_PERCENTAGE (85% by default); there is no age-based purge, so irrelevant footage persists indefinitely until storage is almost full.	**CVSS Score:** 0.0
Threat Context: Inexperienced Attacker		**Severity:** INFORMATIONAL
Public Full Disclosure Date: N/A	**Impact** An attacker with physical access can view, tamper or steal the recordings and AI output from the word-readable partition that lacks app-level encryption.	**CVE #:** N/A
CVSS 4.0 AV:L/AC:L/AT:N/PR:L/UI:N/VC:N/VI:N/VA:N/SC:N/SI:N/SA:N		**CWE:** N/A
Discovered By: Jon Gaines	**Affected Hardware/Software** Falcon/Sparrow/Flex* LPR & ‘Picard/Bravo’ Compute Box	**Further Research Recommended:** Y
Notes: It is unclear if the devices that are deployed in the wild have different data storage policies.		
Relevant Output:		

Steps to Reproduce:

1. Fill ‘/storage/emulated/0/flockMedia/media’ beyond 85% capacity.
2. Monitor cleanup activity using the following command and observe the oldest clips being removed only under disk pressure: `adb logcat -s MediaManagement | grep "Deleting file”`

Tools Used:

- adb

Mitigation: Implement time-/event-based retention. Fix the issue with the data-log cleanup service if that is to be implemented.

FINDING 45: Records are stored on unencrypted external partition

Description		
Type: Data Policy Failure	The Falcon/Sparrow & Picard/Bravo Compute Box were found to capture sessions write raw media into the ‘/storage/emulated/0/flockMedia/…’ directory via Environment.getExternalStoragePublicDirectory when configured.	**CVSS Score:** 0.0
Threat Context: Inexperienced Attacker		**Severity:** INFORMATIONAL
Public Full Disclosure Date: N/A	**Impact** An attacker with physical access can view, tamper or steal the recordings and AI output from the word-readable partition that lacks app-level encryption	**CVE #:** N/A
CVSS 4.0 AV:L/AC:L/AT:N/PR:L/UI:N/VC:N/VI:N/VA:N/SC:N/SI:N/SA:N		**CWE:** N/A
Discovered By: Jon Gaines	**Affected Hardware/Software** Picard/Bravo Compute Box & Falcon/Falcon/Flex* LPR	**Further Research Recommended:** Y
Notes: The severity of this finding has been reduced significantly as it is unclear if units deployed in the wild are configured with this policy.		
Relevant Output:		

Steps to Reproduce:

1. Trigger a recording on a test camera running this stack (motion event or manual command).
2. Use the following command to confirm a media file exists: <REDACTED_COMMAND>
2. Use ‘adb pull’ a file to verify it opens without credentials.

Tools Used:

- adb

Mitigation: Store recordings within a file based encrypted app-private directory or encrypt media blobs before writing to external storage. Enforce access control whens exporting clips.

FINDING 46: Sensitive Information Disclosed – Datadog API Token

<table>
<tr><th colspan="3">Description</th></tr>
<tr><td>Type: Sensitive Data Disclosure</td><td rowspan="2">The ‘Peripheral application installed on multiple devices was found to hardcode a static Datadog API token.</td><td>CVSS Score: 0.0</td></tr>
<tr><td>Threat Context: Inexperienced Attacker</td><td>Severity: INFORMATIONAL</td></tr>
<tr><td rowspan="2">Public Full Disclosure Date: 09/27/2025</td><td>Impact</td><td rowspan="2">CVE #: CVE-2025-59405</td></tr>
<tr><td>An attacker with physical or local access can issue a broadcast that will wipe and shut off the device. Additionally, another application on the device may be able to as well.</td></tr>
<tr><td colspan="2">CVSS 4.0 AV:L/AC:L/AT:N/PR:L/UI:N/VC:N/VI:N/VA:N/SC:N/SI:N/SA:N</td><td>CWE: CWE-312</td></tr>
<tr><td rowspan="2">Discovered By:
Jon Gaines</td><td>Affected Hardware/Software</td><td rowspan="2">Further Research Recommended: N</td></tr>
<tr><td>Peripheral (com.flocksafety.android.peripheral)application
Falcon/Sparrow/Flex* LPR & ‘Picard/Bravo’ Compute Box</td></tr>
<tr><td colspan="3">Notes:</td></tr>
<tr><td colspan="3">Relevant Output:</td></tr>
</table>

Steps to Reproduce:

Affected File: <REDACTED_FILE_PATH>/BuildConfig.java

BuildConfig.java

```
package com.flocksafety.android.common.lib;

/* loaded from: classes.dex */
public final class BuildConfig {
    public static final String BUILD_TYPE = "release";
    public static final String DATADOG_TOKEN = "pub0[illegible]3";
    public static final boolean DEBUG = Boolean.parseBoolean("true");
    public static final String LIBRARY_PACKAGE_NAME = "com.flocksafety.android.common.lib";
}
```

Tools Used:

- Micro USB Cord/USB-C Cord
- adb

Mitigation: Do not hardcode static sensitive information across devices. Utilize hashing and encryption where applicable.

PUBLIC APPLICATIONS

FINDING 47: Cleartext Communications

<table>
<tr><th colspan="3">Description</th></tr>
<tr><td>Type: Cleartext Transmissions of Sensitive Information</td><td rowspan="2">The FSInstaller Android application was found to allow cleartext communications. In this case the application’s manifest contained ‘android:usersCleartextTraffic=”true” as well as hardcoded references to: ‘ http://192.168.43.1:8080/ and http://%s:8081/LAPI/V1.0/.’</td><td>CVSS Score: 6.9</td></tr>
<tr><td>Threat Context: Inexperienced Attacker</td><td>Severity: MEDIUM</td></tr>
<tr><td rowspan="2">Public Full Disclosure Date: N/A</td><td>Impact</td><td rowspan="2">CVE #: N/A</td></tr>
<tr><td>Using cleartext communications makes it trivial for an attacker to intercept the application’s traffic.</td></tr>
<tr><td colspan="2">CVSS 4.0 AV:A/AC:L/AT:N/PR:N/UI:N/VC:H/VI:L/VA:N/SC:N/SI:N/SA:N</td><td>CWE: CWE-319</td></tr>
<tr><td rowspan="2">Discovered By:
Jon Gaines</td><td>Affected Hardware/Software</td><td rowspan="2">Further Research Recommended: N</td></tr>
<tr><td>FSInstaller Application (com.flocksafety.hazyhiwire)</td></tr>
<tr><td colspan="3">Notes:</td></tr>
<tr><td colspan="3">Relevant Output:
android:usesCleartextTraffic="true"</td></tr>
</table>

Steps to Reproduce:

1. View the application’s AndroidManifest.xml file and note the inclusion of: android:usesCleartextTraffic="true"
2. Decompile and then use a tool like ‘strings’ to search for hardcoded URLs that utilize cleartext HTTP communications.

Tools Used:

- adb
- strings

Mitigation: Build production application with the ‘android:usersCleartextTraffic=”false”’ property.

FINDING 48: Sensitive Information Disclosure – Google API Key

Description		
Type: Sensitive Data Disclosure	The Flock Safety Android application was found to contain a hardcoded Google API key.	**CVSS Score:** 0.0
Threat Context: Inexperienced Attacker		**Severity:** INFORMATIONAL
Public Full Disclosure Date: N/A	**Impact** An attacker can download the application, extract the API keys and use them to access their backend APIs if they are valid.	**CVE #:** N/A
CVSS 4.0 AV:A/AC:L/AT:N/PR:N/UI:N/VC:N/VI:N/VA:N/SC:N/SI:N/SA:N		**CWE:** CWE-319
Discovered By: Jon Gaines	**Affected Hardware/Software** Flock Safety (com.flocksafety.sweetwater)	**Further Research Recommended:** N
Notes:		
Relevant Output:		

Steps to Reproduce:
Affected File: <REDACTED_FILE_PATH>/strings.xml
Value: <REDACTED_API_KEY>
Mitigation: Do not include API keys client-side. Implement hashing and encryption where possible.

FINDING 49: Plaintext HTTP in Logs

Description		
Type: Sensitive Data Disclosure	The FlockOnPatrol Android application was found to leak plaintext HTTP requests and responses into logcat logs.	**CVSS Score:** 0.0
Threat Context: Inexperienced Attacker		**Severity:** INFORMATIONAL
Public Full Disclosure Date: N/A	**Impact** An attacker can download the application, extract the API keys and use them to access their backend APIs if they are valid.	**CVE #:** N/A
CVSS 4.0 AV:A/AC:L/AT:N/PR:N/UI:N/VC:N/VI:N/VA:N/SC:N/SI:N/SA:N		**CWE:** CWE-319
Discovered By: Jon Gaines	**Affected Hardware/Software** FlockOnPatrol (com.flocksafety.android.negroni)	**Further Research Recommended:** N
Notes: This application is likely past its End of Life (EOL)		
Relevant Output:		

Steps to Reproduce:
1. Install the production APK on an Android 10+ test device.
2. Authenticate and trigger the "Run Plate" workflow to send a localization request.
3. From a workstation with adb access, execute `adb logcat -s OkHttp` and observe logged request headers/bodies containing Authorization values.

Tools Used:
- adb

Mitigation: Do not log HTTP requests.

FINDING 50: Sensitive Information Disclosure – API Keys

Description		
Type: Sensitive Data Disclosure	The FlockOnPatrol Android application was found to contain multiple hardcoded API keys.	**CVSS Score:** 0.0
Threat Context: Inexperienced Attacker		**Severity:** INFORMATIONAL
Public Full Disclosure Date: N/A	**Impact** An attacker can download the application, extract the API keys and use them to access their backend APIs if they are valid.	**CVE #:** N/A
CVSS 4.0 AV:A/AC:L/AT:N/PR:N/UI:N/VC:N/VI:N/VA:N/SC:N/SI:N/SA:N		**CWE:** CWE-319
Discovered By: Jon Gaines	**Affected Hardware/Software** FlockOnPatrol (com.flocksafety.android.negroni)	**Further Research Recommended:** N
Notes: This application is likely past its End of Life (EOL)		
Relevant Output:		

Steps to Reproduce:

1. The following API keys were found to be contained within the application:

```
bugsnag_key=<REDACTED_API_KEY>
MIXPANEL_TOKEN=<REDACTED_API_KEY>
```

Tools Used:

- adb

Mitigation: Do not include API keys client-side. Implement hashing and encryption where possible.

External Contributor

FINDING 51: Remote Code Execution (RCE) – System*

Description		
Type: Code Execution	The Falcon/Sparrow/Flex* LPR and Picard/Bravo Compute Box were found to enable the chaining of multiple vulnerabilities disclosed in this paper together resulting in control of devices with system permissions.	**CVSS Score:** 9.8
Threat Context: Experienced Attacker		**Severity:** CRITICAL
Public Full Disclosure Date: 01/23/2026	**Impact** An attacker with adjacent or physical access can leverage the Android applications installed with debugging enabled by the Vendor to achieve system command injection.	**CVE #:** PENDING
CVSS 4.0 AV:A/AC:L/AT:P/PR:N/UI:N/VC:H/VI:H/VA:H/SC:H/SI:H/SA:H		**CWE:** CWE- 78
*Discovered By: Joseph “JosephRC” Cohen	**Affected Hardware/Software** Picard/Bravo Compute Box & Falcon/Sparrow/Flex* LPR	**Further Research Recommended:** N
Notes:		
Relevant Output:		

Steps to Reproduce:

1. Connect to one of the affected devices via their USB port.
2. Use adb to wirelessly connect to device as the ‘shell’ user.
3. Use debug access along with a ‘trigger’ to execute commands as system.

Tools Used:

- Micro USB Cable/USB-C Cable
- adb

Mitigation: Implement Authentication and Authorization. Do not ship applications with debug enabled in production.

TIMELINE

Vendor Communications Overview	
Event	**Date**
Initial Contact to Vendor – Part 1	02/08/25
First Response from Vendor	02/10/25
Vendor Submitted Request for CVE Numbers for 10 of the vulnerabilities	03/07/25
Vendor confirmation of submission and explanation on what they chose to submit to MITRE	03/07/25
Vendor PR Statement about Part 1 Disclosures - Link	05/05/25
Full Disclosure – Part 1	06/19/25
Further vulnerabilities disclosed to Vendor – Part 2	06/19/25
First batch of CVE Published	06/27/25
Further vulnerabilities disclosed to Vendor – Part 3	06/27/25
Followed up, provided disclosure deadline	06/27/25
Vendor confirmed validation/triage in progress	06/27/25
Vendor responded that existing CVEs 2025-47823 and 2025-47824 apply	09/03/25
Replied clarifying CVEs do not apply to the Compute Box (Picard/Bravo) or the Android application vulnerabilities; notified Vendor of intent to submit directly to MITRE	09/03/25
Full Disclosure Part 2	09/19/25
Full Disclosure Part 3	09/27/25
Further vulnerabilities disclosed to Vendor Part 4	10/23/25
White paper and Formal Statement Published Publicly	11/05/25
White paper public release update	11/06/25
Vendor PR Statement about White Paper - Link	11/06/25
Further vulnerabilities disclosed to Vendor Part 5	11/11/25
White paper public release update	11/11/25
Full Disclosure Part 4 - Pending	01/23/26
Full Disclosure Part 5 – Pending	02/11/26

Document Timeline			
Document State	**Date**	**Version**	**Author**
Draft	10/27/25	0.4	Jon Gaines
Direct Release	11/4/25	1.0-DR	Jon Gaines
Public Release	11/5/25	1.0-PR	Jon Gaines
Public Release Update	11/6/25	1.1-PR	Jon Gaines
Public Release Update	11/11/25	1.2-PR	Jon Gaines

CONCLUSION

Although this report is extensive, its goal was neither to identify every possible security issue nor to portray the Vendor negatively. The broader state of hardware security represents an industry wide concern, one defined more by systemic weaknesses than isolated defects. This research offers a focused glimpse into that reality. All manufacturers and vendors share a responsibility to strengthen the security of the products they deploy.

There are also positives to note, most of the identified issues can be remediated through consolidated fixes rather than unique patches, demonstrating that meaningful improvement is achievable without excessive complexity. More importantly, even in its current form, this work aims to raise awareness of the largely unexplored field of anti-crime device security posture and to encourage further research in this area.

Regardless of individual perspectives on the use or deployment of these technologies, new standards and minimum baselines must be established if anti-crime devices are here to stay. Most importantly, equipment utilized in public safety technology deployments must strive to be as secure and resilient as possible, ensuring that both the devices and the data they collect cannot be compromised, manipulated, or weaponized by malicious actors, whether domestic or foreign.

FOR IMMEDIATE RELEASE: Distributable Formal Statement by Jon "GainSec" Gaines, the Independent Security Researcher who discovered, disclosed, conducted the analysis and authored the white paper: *Examining the security posture of an Anti-Crime Ecosystem*

Date: 11/05/2025
Current Version Release Date: 11/06/2025
Current Version Release Date: 11/11/2025

Contact for verification or follow-up: whitepaper@gainsecmail.com

DISTRIBUTABLE FORMAL STATEMENT

Disclaimer

This statement documents good faith, independently conducted security research performed exclusively on lawfully acquired hardware under the researcher's control. No testing involved unauthorized access to any network, account, or production environment. All methods complied with 18 U.S.C. § 1030 and 17 U.S.C. § 1201 (g) exemptions for good-faith security testing.

The content is intended to inform defenders and vendors; it is not an instruction manual for exploitation. Replication on systems not under explicit authorization violates U.S. and international computer-misuse law.

The author affirms that this research was performed independently, without financial support, employment, consultancy, or material benefit from the vendor or its affiliates. No funding, compensation, or third-party direction influenced the selection of targets, the methods used, or the interpretation of results. The devices analyzed were purchased by the author. Tests were confined to offline/lab environments; no interception of third-party communications or content prohibited by ECPA/Title III occurred; no human-subjects' data were collected.

The purpose of this study is to advance public understanding of security posture and responsible disclosure practices, not to promote or discredit any product or company. This document, in its current version, is intended for defensive-security evaluation, compliance verification, and policy development. Redistribution that adds operational detail, live credentials, or working exploits is prohibited.

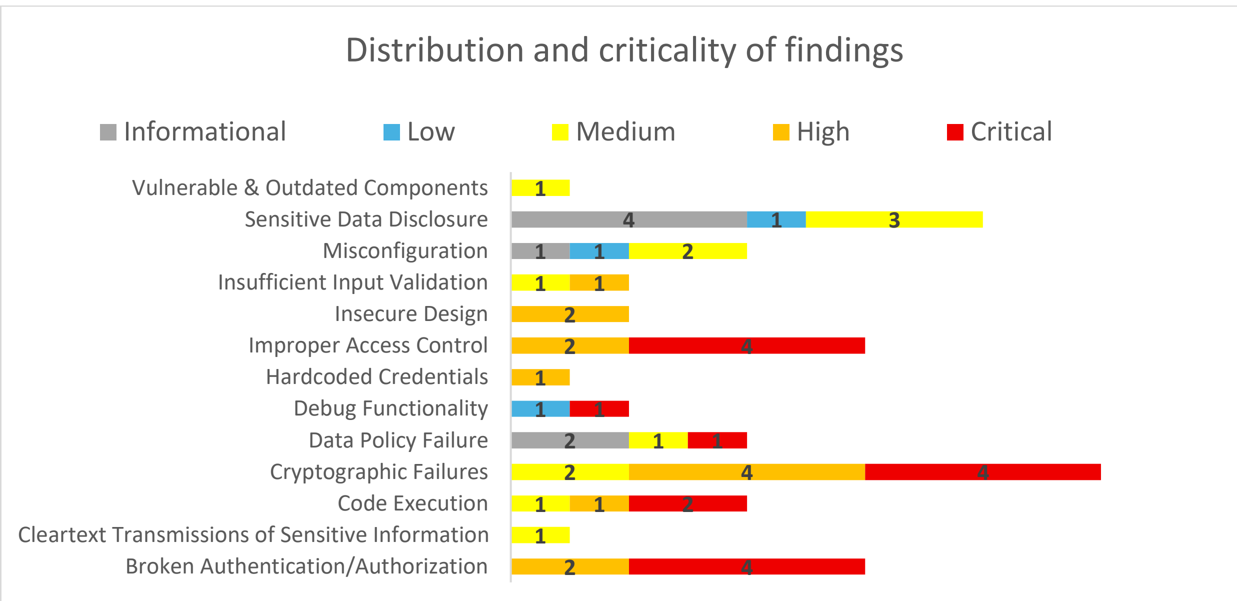
Descriptions omit operational steps and live secrets by design. Any reproduction must be limited to assets the reader owns or is authorized to assess, in controlled labs, and solely to validate remediation. Do not apply to third-party or deployed systems.

Background

I submit this statement as an independent security researcher with over a decade of professional offensive security experience. This statement condenses my whitepaper which consolidated findings from my multipart independent research into Flock Safety's hardware (Raven Gunshot Detection; Falcon/Sparrow/Flex* LPR; Picard/Bravo Compute Box), the Android applications deployed on those devices as well as those available on public app stores. The work followed responsible disclosure practices and resulted in assigned and pending CVE identifiers via MITRE with the National Vulnerability Database (NVD) program managed by the National Institute of Standards and Technology (NIST) and sponsored by the Department of Homeland Security (DHS) via Cybersecurity and Infrastructure Agency (CISA).

Summary of Probable Cause

Analysis identified recurring deficiencies in cryptographic enforcement, access control implementation, and key management. Version 1.2-PR of the white paper documents fifty-one (51) findings across three device families, shared applications, and publicly available applications; twenty-two (22) carry CVE identifiers, with additional CVEs pending. The Total Findings Chart below substantiate category distribution across devices.



Material Facts

1) Hardware trust chain is disabled at scale. Raven Gunshot Detection System ships without Secure Boot and with flash unencrypted; rollback protection absent (CVE-2025-47819, CVE-2025-47820). LPR units have Secure Boot disabled, bootloader unlocked, EDL/QDL accepting unsigned loaders (CVE-2025-47822) and unencrypted EMMC (CVE-2025-47824). Pircard/Picard/Bravo Compute Box repeats the pattern: Secure Boot off (CVE-2025-59408), bootloader unlocked (CVE-2025-59404), unauthenticated EDL/QDL (CVE-2025-59402), unencrypted UFS.

2) Administrative plane is unauthenticated. The "Collins" service exposes device administration on all Interfaces without authentication, including live view toggling, reboots, log/crashpack retrieval, and enabling ADB over TCP (CVE-2025-59403).

3) Hidden debug enables wireless access with uniform weak credentials. A specific but simple button press sequence starts a device hotspot; default password "<redacted_password>" is uniform across units (CVE-2025-59403).

4) ADB authentication disabled; sideloading permitted. LPR and Picard/Bravo allow unauthenticated ADB and APK sideloading, providing trivial code execution footholds when chained with the items above.

5) End of Life operating system in the field. LPR devices run Android Things 8.1 (EOL 2022), violating basic lifecycle controls and increasing exploit exposure.
6) Hardcoded secrets and key material in production. Multiple applications embed API keys, a Java Keystore password used for mTLS, and a static OAuth client secret (e.g., CVE-2025-47823, CVE-2025-59407, CVE-2025-59406).
7) Data policy weaknesses. Retention defaults to capacity thresholding rather than time/event policy; recordings may be written to a public external path without app-level encryption. It remains unclear whether these configurations persist in deployed production units. Media recordings are accessible to anyone with access to the device. Additionally, there is a lack of encryption when the device is running and collecting recordings as well as cross application data exposure. (Multiple CVEs Pending)

Context of Exposure

Devices are commonly mounted on short, publicly accessible poles; physical interfaces remain externally accessible under typical deployment configurations. In the latest iteration, the physical trigger for enabling the hotspot remains exposed, compounding the risk from unauthenticated APIs and ADB. Observed weaknesses permit unauthorized command execution, data exfiltration, and device manipulation through trivially reproducible vectors.

Technical Detainment/Remediation

1) Enforce Secure Boot, lock bootloaders, require authenticated, signed loaders for all EDL/QDL interactions; enable at rest encryption for flash/EMMC/UFS across Raven, LPR, and Picard/Bravo.
2) Bind administrative APIs (Collins) to authenticated channels; disable ADB over TCP enablement via HTTP; restrict to loopback or a mutually authenticated control plane.
3) Remove hidden hotspot triggers; replace uniform passwords with per-device credentials; require explicit operator pairing for any debug or service shell.
4) Ship Android apps with 'android:debuggable=false;' remove unauthenticated ADB pathways and sideload capability on production builds.
5) Rotate and revoke embedded keys, keystore passwords, OAuth client secrets; issue per-device mTLS materials; eliminate client-side key exposure.
6) Replace capacity only retention with time/event policies; store media in file based encrypted app-private storage or encrypt before externalization.
7) Migrate LPR off End-of-Life OS; establish patch management and inventory capable of immediate decommissioning and SIM revocation where remediation is infeasible.

Declaration & Contact

This declaration is accurate to the best of the researcher's knowledge and derived from contemporaneous research records.

It is intended for distribution to journalists, privacy advocates, regulators, law enforcement leadership, counsel, and legislators evaluating current deployments and required corrective actions.

This formal statement may be redistributed verbatim for transparency, provided operational details remain redacted. Derivative publication requires preservation of attribution, version number (1.2 PR), and checksum of the signed PDF.

Full Whitepaper is available via https://github.com/gainsec/anti-crime-ecosystem-research
Mirror: https://zenodo.org/records/17529424
White Paper DOI: 10.5281/zenodo.17529424

For further inquiries about this statement or its formal whitepaper, please reach out to the email included at the beginning of this statement.

APPENDIX A: TERM GLOSSARY

ADB (Android Debug Bridge): A tool that allows communication with Android devices for maintenance or testing. When left unsecured, it can provide full access to the device.
API (Application Programming Interface): A bridge that allows one program or device to communicate with another, often over a network.
Authentication / Authorization: Authentication confirms identity, while authorization determines what actions that identity is allowed to take.
Bootloader: The low-level startup program that loads an operating system when a device powers on. If unlocked, it allows anyone to replace the system software.
CVE (Common Vulnerabilities and Exposures): The global identification system for publicly disclosed cybersecurity flaws.
CWE (Common Weakness Enumeration): A standardized catalog describing the type of coding or design flaw that causes vulnerability.
Debug Interface (UART / JTAG): Hardware ports used by engineers to test and repair devices. If left active, they can allow attackers to bypass protections.
EDL / QDL Mode (Emergency / Qualcomm Download): Manufacturer recovery modes used to re-flash or modify device memory. If not secured with authentication, they can be exploited to rewrite firmware.
Encryption: A process that protects data by converting it into a coded format that only authorized parties can read.
Firmware: The built-in software stored on a hardware device that controls its core functions.
Firehose Loader: A Qualcomm-specific tool used in EDL mode to read or write device memory during manufacturing or repair.
Hardcoded Credentials: Built-in usernames, passwords, or keys directly written into software. This is considered insecure because they cannot easily be changed or revoked.
Immutable Root of Trust / Secure Boot: A hardware-based security mechanism that ensures only verified and trusted software runs when a device starts.
JTAG (Joint Test Action Group): A hardware debugging interface that provides low-level access to chips and system components.
Magisk: An Android tool that modifies the boot image to grant "root" (administrator) access for testing or research purposes.
mTLS (Mutual Transport Layer Security): A secure communication method where both sides (the device and the server) verify their identities before exchanging information.
Root / Root Shell: The highest level of administrative access, granting unrestricted control over a device or operating system.
Rollback Protection (Anti-Rollback): A security feature that prevents reverting to older, potentially vulnerable firmware or software versions.
SELinux (Security-Enhanced Linux): A kernel-level security module that limits what processes can do, even if compromised, by enforcing strict access rules.
UART (Universal Asynchronous Receiver-Transmitter): A simple hardware interface used for serial communication and device debugging.
UFS / eMMC / Flash Storage: Types of internal memory chips used in embedded systems to store firmware and user data.
Responsible Disclosure: The ethical practice of reporting security vulnerability privately to a vendor before making it public.
Exploitability: A measure of how easily a vulnerability can be used by an attacker to cause harm.
Mitigation: A technical or procedural change implemented to reduce or eliminate a security risk.

APPENDIX B: METHODOLOGY

A. Preliminaries

Scope & Authorization: Define targets, date ranges, and limitations. List what is out-of-scope.
Acquisition & Chain-of-Custody: Acquire devices legally; Confirm they are legitimate and prepare lab environment
Lab Controls: Follow industry standard or beyond in terms of documentation and network topology.

B. Reconnaissance

Public Artifact Collection: Gather vendor docs, firmware images, published binaries, app store listings, and support pages.
Surface Enumeration: Map visible services/endpoints on the host, Map physical interfaces. Research what chipset(s) the device run. Note any APIs, web services or other commonly exposed services on the host at a high level.
Physical Inspection: Photograph device markings, access panels, and connector locations for later reference. Determine if any type of new equipment is required.

C. Host-Based (Embedded, RTOS, Linux, Android)

Imaging & Preservation: Create forensic-quality images of storage where permitted; preserve boot logs and configuration snapshots.
Configuration Review: Inventory running services, startup scripts, user accounts, and installed packages.
Log & Artifact Harvesting: Collect system logs, installed certificates, and configuration files for offline review.
Behavioral Observation: Observe boot sequences, update behaviors, and service registration in non-destructive runs.

D. Mobile Applications (Android/iOS)

App Acquisition: Obtain APKs/IPAs from official sources or device extractions; preserve original package and signatures.
Static App Review: Inspect manifest/entitlements, embedded certificates, resource files, and strings for hardcoded endpoints or secrets.
Binary Analysis: Decompile/reverse high-level logic to identify authentication flows, API usage, and cryptographic patterns (focus on design, not exploits).
Runtime Observation: Monitor app behavior in an instrumented testbed (network captures, logs) to confirm observed static findings without executing harmful payloads.

E. Firmware & Software Reverse Engineering

Firmware Extraction: Collect firmware images from published updates or device dumps, recording checksums and version metadata.
Partition Analysis: Identify boot, kernel, rootfs, and config partitions; extract filesystems when readable.
Static Reverse-Engineering: Catalog libraries, interpreters, and custom binaries; identify insecure crypto usage, default keys, or weak update verification.
Dependency & Component Mapping: Note outdated third-party components and CVE history for exposed libraries.

F. Hardware & Debug Interfaces

Non-Invasive Recon: Identify exposed headers, debug ports, switches, and fuses through visual inspection and vendor docs.
Interface Enumeration: Document presence of UART, JTAG, EDL/QDL, test pads, and external connectors; record labeled signals and access barriers.
Passive Observation: Capture boot serial logs and pinouts where available; preserve all raw output logs.
Tamper & Protection Assessment: Check for physical protections (fuses, epoxy, secured connectors) and anti-rollback or secure-boot indicators.

G. Memory, Storage & Forensics

Volatile Data Capture: When permitted, record memory images or transient logs in a forensically sound manner.
Storage Analysis: Extract and inspect filesystem artifacts, databases, and retained credentials (redacted in outputs).
Correlation: Correlate volatile and persistent artifacts with observed behavior to validate severity.
H. Validation, Triage & Risk Assessment
Reproducible Checks: Validate findings using repeatable, non-destructive checks and independent peer review.
Categorization: Map issues to CWE/CVE where applicable and assign impact and exploitability tiers using documented criteria.
Abstraction for Publication: Replace sensitive data with abstractions or redacted examples to prevent operational misuse.

I. Remediation Advice (High-Level)

Design Fixes: Recommend secure boot, authenticated update channels, anti-rollback, and removal of hardcoded secrets.
Configuration Hardening: Recommend least-privilege services, rotated credentials, and disabled debug interfaces in production images.
Operational Controls: Suggest monitoring for anomalous firmware changes and strict supply-chain controls.

J. Responsible Disclosure & Follow-up

Vendor Coordination: Share findings privately with vendors, providing sanitized evidence and reproduction notes under embargo until responsible full disclosure timeline ends.
Archive & Audit: Keep track of timelines, and disclosure communications.

K. Documentation & Reporting

Evidence Records: Maintain annotated screenshots, logs, hashes, and analysis notes tied to each finding.
Non-Actionable Reporting: Present root causes, impact statements, and mitigation recommendations in public reports without stepwise exploitation details.

APPENDIX C: DEFENDERS CHECKLIST

Top 15 Checks for Defenders to perform; Reach out to Author for full Defenders Checklist						
Rank	Phase	Task	Findings Ref	Devices Affected	Priority	Standard Map
1	Field	Enable Secure Boot (enforce anti-rollback)	1,12,21	Raven, LPR, Picard/Bravo	P0	800-53 SI-7(17), CSF PR.PT-1
2	Field	Lock Bootloader after firmware install	13,22	LPR, Picard/Bravo	P0	800-53 CM-5, SI-7
3	Field	Authenticate and disable EDL/QDL loaders (signed firehose only)	14,23	LPR, Picard/Bravo	P0	800-53 AC-3, SC-12
4	Field	Disable unauthenticated ADB; require pairing; disable sideload	15,16,24,25	LPR, Picard/Bravo	P0	OWASP MASVS, 800-53 AC-17
5	Field	Enable Flash/EMMC/UFS encryption for all devices	5,17,26	Raven, LPR, Picard/Bravo	P0	800-53 SC-28
6	Field	Bind Collins admin API to loopback or control network; require authN/mTLS	27	LPR, Picard/Bravo	P0	800-53 AC-3, SC-13
7	Field	Remove hidden hotspot trigger; set strong unique password per device	28	All	P0	800-53 AC-18, SC-7
8	Field	Remove debug build flags (android:debuggable=false) from production apps	32	All	P0	OWASP MASVS, 800-53 CM-6
9	Field	Rotate/revoke hardcoded secrets and mTLS keys	33,35,37,40	All	P0	800-53 IA-5, SC-12
10	Field	Encrypt media and logs; enforce time-based retention policy	38,39	All	P0	800-53 MP-6, SI-12
11	Field	Migrate LPR off Android Things 8.1 (EOL OS)	18	LPR	P0	800-53 SI-2
12	Field	Implement server certificate validation / pinning	10	Raven	P1	800-53 SC-23, SC-8
13	Field	Disable UART/JTAG/UART-Download debug ports	2,6,7	Raven	P1	800-53 SC-7, AC-3
14	Field	Remove hardcoded Wi-Fi SSIDs; disable auto-connect	4,19	Raven, LPR	P1	800-53 SC-12
15	Field	Enable SELinux enforcing on deployed Android builds	34,36	Picard/Bravo, LPR	P1	800-53 AC-6, SI-7