The Chain Dies at Step Three: Measured Three-Arm Closure, Mechanism Location Without Decompilers, and a Live-Fired Detection Package for the PaperCut NG Authentication-Bypass to Unsafe-Driver-Loading Chain (CVE-2026-81578 / CVE-2026-82078)

Xpectra Security Research
Autonomous Red/Blue Operations

***Abstract*—The PaperCut NG/MF chain CVE-2026-81578 → CVE-2026-82078 was disclosed by vendor advisory on 2026-08-27, assigned CVEs on 2026-08-28, added to CISA KEV on 2026-08-31 (due 2026-09-14), and a merged Metasploit module existed by 2026-09-03. This paper makes no discovery claim. It contributes what the public record left unmeasured, from a fictional, zero-egress, three-arm Docker lab built on stock vendor installer bytes: (i) an end-to-end reproduction of the full chain on unmodified 25.0.11.75758 using *zero credentials* and a hand-assembled 977-byte proof-of-execution class, with restore-to-factory configuration proven by cold database reads on repeat runs; (ii) a same-day three-arm closure matrix showing the chain fires on 25.0.11 and dies at the same step (3/11, a 302 bounce of the forged request) on both the emergency-patch arm 25.0.12-PO-4560.76533 and the maintenance release 25.0.13.76605, plus mechanism location obtained without any decompiler — the shipped `papercut.application` service override plus a `ValidatingDirectService` class, the vendored Tapestry engine jars being hash-identical across all arms — and unit corroboration that the payload artifact still executes on the fixed build, so the fix closes the *path*, not the artifact; (iii) a detection package in which every rule was live-fired or cut, including an attempt-level wire rule proven to fire on patched arms and a process-lineage rule that cannot; and (iv) the non-vacuous-negative methodology behind these results (cold reads, same-day positive controls, a five-pillar gate returned `VALID`), with the 33-entry attempt log published as the honest-limitations record.**

***Index Terms*—PaperCut, CVE-2026-82078, CVE-2026-81578, authentication bypass, unsafe reflection, closure matrix, closed-source fix analysis, detection engineering, reproducible experiments, honest negative results.**

I. Introduction

Print-management servers are attractive targets: they run as a service, hold a directory of users, and expose a management web interface that must by design interact with the local system. PaperCut NG/MF is such a product. In late August 2026 the vendor disclosed a two-link chain: an authentication bypass in the web management interface (CVE-2026-81578, urgent advisory 2026-08-27, CVE published 2026-08-28) that an unauthenticated remote request can use to modify certain system configurations, and an unsafe dynamic class-loading flaw in the database connection utilities (CVE-2026-82078) which the CVE Record describes verbatim: the application "instantiates database driver classes based on configurable driver names without validating against an allowlist of approved drivers … enables the execution of arbitrary Java bytecode residing on the application classpath under the security context of the PaperCut server process" [3]. Chained, one unauthenticated HTTP session becomes arbitrary Java bytecode execution as the `papercut` service user.

The public record is rich. The vendor published mechanism, scores, fixed versions and per-installer SHA-256 sums [1]; Rapid7 named the display/execute divergence that glues the chain [6]; Huntress reproduced a pre-auth configuration takeover and the complete RCE on a stock 25.0.11.75758 [7]; a full-chain exploit module was merged into Metasploit on 2026-09-03 [9]; and both CVEs entered the CISA KEV catalog on 2026-08-31 with a federal due date of 2026-09-14 [5].[1] **No discovery claim is made anywhere in this paper**; the credit division is the timeline above.

What the public record does not contain — and what defenders actually need — is measurement: *which* fixed build kills the chain *at which step*; whether the kill is proven with a live, non-vacuous negative; where the fix physically lives when the vendor ships no source and no public commit; and what a defender can actually see on the wire or the host, with false-positive behavior measured rather than asserted. At intake the public detection surface for this chain was genuinely empty: no Sigma, Suricata, YARA, ExploitDB, or vulhub entries, and the official Nuclei template contribution was still an open pull request [12]. That empty surface is now a published, fired package.

Contributions

- **C1 — Measured end-to-end reproduction on stock bytes in a zero-egress lab.** The full 11-step / 24-request chain fires on an unmodified vendor installer

[1] A third-party write-up additionally claims a bypass of the *first* emergency patch [8]; the Metasploit module header and the Rapid7 post carry the same claim independently. We cite it, attribute it, and do not adopt it — our lab never measured EPR1, whose bytes were pulled from the vendor CDN before we could fetch them (Section VII).

(25.0.11.75758, our recorded SHA-256 as custody record) with *zero credentials* (Section IV). Unlike the merged Metasploit module — which likewise needs no credentials but ships a Groovy/HTTP-server payload JAR — our artifact is a 977-byte, hand-assembled proof-of-execution classfile whose success path throws nothing; it self-deletes its own dropped bytes, and restore-to-factory `tbl_config` is proven by cold (in-process-lock-free) database reads on three repeat runs and re-proven as a same-day positive control twice more.

- **C2 — A three-arm closure matrix measured in one day, plus mechanism location without decompilers.** The same unmodified exploit bytes fire on 25.0.11, die at step 3/11 on emergency-patch arm 25.0.12-PO-4560.76533 (EPR3), and die identically on maintenance release 25.0.13.76605; the 302 death is captured at raw-packet level (Section V). The fix is located by diffing *shipped* artifacts: the stock `WEB-INF/papercut.application` spec has no direct-service override, both fixed arms register `ValidatingDirectService` (byte-identical between arms), and every vendored Tapestry engine jar is SHA-256-identical across all three arms — the fix replaces the service class, not the engine. Unit corroboration inside the fixed arm shows the exact payload bytes still execute when loaded under the vendor's own JRE: the fix closes the *path*, not the *artifact* — a transferable method for closed-source fix analysis.
- **C3 — A detection package where every shipped rule is live-fired or cut.** Seven attack SIDs (attempt-, stage-, and execution-labeled) plus a sensor-health heartbeat SID, six Sigma rules with field aliases, six YARA rules with clean-stock negatives, and a self-test scanner; three candidate rules were *cut* with measured reasons, and one attempt rule is proven to fire on patched arms too (Section VI).
- **C4 — The non-vacuous-negative methodology itself**: cold database reads, same-day positive controls bracketing every negative within minutes, a five-pillar automated gate returned `VALID` (10 checks), harness supersessions logged rather than deleted, and a detector self-test with a measured non-mutation proof (Section III).

Sections II–V present the vulnerability, lab, and measurements; Section VI the detection package; Section VII patch and IR guidance; Section VIII ethics, honesty, and limits.

II. Vulnerability Characterization

A. Chain anatomy

The two CVEs are one kill-chain; neither alone reaches code execution from the internet. **Link 1 (CVE-2026-81578):** PaperCut's web tier is Apache Tapestry 3.0.4 (vendored). A Tapestry *direct service* request encodes a page and a component listener in the URL. In the affected builds the request "names one page to render and a different page owning the component to execute; the access check trusts the *rendered* page" [6], [12] — the shipped public Home/Error pages are unauthenticated, so a request that displays `Home` while executing a listener of the privileged `ConfigEditor` page runs the listener *before* access validation completes (the CVE Record's own phrasing: backend actions "prior to the completion of access validation checks" [2]). The reachable effect is the configuration editor: arbitrary `tbl_config` property writes.

Link 2 (CVE-2026-82078): among those writable keys are the external user-lookup database settings (`user-lookup.db-driver`, `db-url`, `id-to-username-sql`). The connection utility calls `Class.forName()` on the configured driver name with no allowlist; the loaded class's `<clinit>` runs under the server process's security context. Because the configuration stage already points the JDBC URL at the *bundled Derby* engine with an attacker-chosen stored-procedure call (`SYSCS_UTIL.SYSCS_EXPORT_QUERY_LOBS_TO_EXTFILE`), the chain also owns a file-write gadget: it writes a chosen hex BLOB to a CWD-relative path — `lib/<name>.class` is on the classpath. The "arbitrary Java bytecode residing on the application classpath" of the CVE wording is thus reachable without uploading any file through any upload feature.

Vocabulary note, measured rather than assumed: the phrase "complex direct" is Rapid7's and the vendor FAQ's, not the product's — a grep for its spellings across every entry of every extracted jar and the WAR `WEB-INF` of all three arms returns **0 hits in shipped bytes** (Section V); the shipped strings that evidence the mechanism are different (Section V-B).

B. Trust-hop analysis

Table I decomposes the chain by the trust assumption broken at each hop. Every hop was exercised in our lab (Section IV); the chain's length is its detection surface, which is what Section VI harvests.

C. Access-control analysis of the scores

The vendor scores the legs separately: 81578 at CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:L/VI:H/VA:L/SC:N/SI:N/SA:N = 8.8 HIGH and 82078 at `CVSS:4.0/AV:N/AC:L/AT:N/PR:H/UI:N/VC:H/VI:H/VA:H/SC:H/SI:H/SA:H` = 9.4 CRITICAL [2], [3], [1]. The `PR:H` on 82078 encodes "if an attacker can manipulate system configuration" — true only via 81578, whose PR:N is what the chain supplies. **Our measurement matches the chained reading: every leg ran with zero credentials** (Section IV); no login, no session beyond an anonymous JSESSIONID.

NVD analyzed both CVEs with its own CVSSv3.1 primary metrics: **81578 = 9.8** (CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H) and **82078 = 9.1** with `S:C` [4]. These diverge from the vendor v4.0 in effect/scope (81578 `VC:L→C:H`; 82078 `S:N→S:C`); we record the divergence, cite both, and blend neither. On our measurements the impact of the chain is "full CIA inside the service account": the executed proof-of-execution payload wrote a file as `uid=1000(papercut)` and ran arbitrary process children as that user — the vendor `PR:H` understates reachability *for the chain*, which is exactly what the companion CVE is for.

D. Both CWEs, verbatim, as a conflict — no winner

For CVE-2026-82078 every source agrees: CWE-470, "Use of Externally-Controlled Input to Select Classes or Code

TABLE I: Trust hops, each with the measured consequence. "Assumption broken" is the implicit trust the build places in something an unauthenticated request controls.

Hop	Assumption broken	Measured effect
H1 anonymous HTTP	management interface is reachable but gated	public `Error`/`Home` pages answer without a session; version oracle returns `NG 25.0.11 build 75758`
H2 direct-service dispatch	the rendered page authorizes the executed component	a forged direct-service URL naming `Home` executes the privileged listener; response 200 with `<!-- Page:Home -->`
H3 config editor listener	only operators mutate `tbl_config`	four `user-lookup.*` keys rewritten (driver, URL, SQL, enabled)
H4 driver `Class.forName`	configured drivers are trusted allow-list members	attacker-named class initializes in the server JVM; a non-`Driver` class is tolerated (connect error is caught and logged)
H5 Derby procedure	SQL run for lookup is benign	hex BLOB written to `lib/<B>.class` (classpath, CWD-relative)

('unsafe reflection')" [3], [5]. For CVE-2026-81578 the classification *contradicts itself inside the vendor's own artifacts*, and we present both verbatim rather than invent a winner:

- PaperCut's **CVE Record** (CNA container): "CWE-305 Authentication Bypass by Primary Weakness" [2]. NVD inherits 305 (Secondary) and adds no CWE of its own [4].
- PaperCut's **own advisory page**: "CWE-306 Missing Authentication for Critical Function" [1]. CISA KEV names the entry "PaperCut NG/MF *Missing Authentication for Critical Function* Vulnerability" with CWE-306 [5], and Rapid7 follows 306 [6].

No source reconciles them [12]. Our measurements cannot adjudicate the taxonomy — both classes describe what we observed (an unauthenticated request reaching a privileged function; authorization performed *after* dispatch) — so both stand cited, unlabeled by preference, in this paper.

III. Lab and Methodology

A. Target provenance

Table II records the three installers — the experiment's single variable. All three were fetched from the vendor CDN and hashed on intake.

B. Lab conditions (S-A set) and integrity controls

The lab was built once (Dockerfile + per-arm installer), and identical conditions across arms were the design constraint: *S-A0 stock integrity:* vendor installer `--non-interactive` as user `papercut`, no config pre-edits; the setup wizard is completed only by replaying its own Tapestry

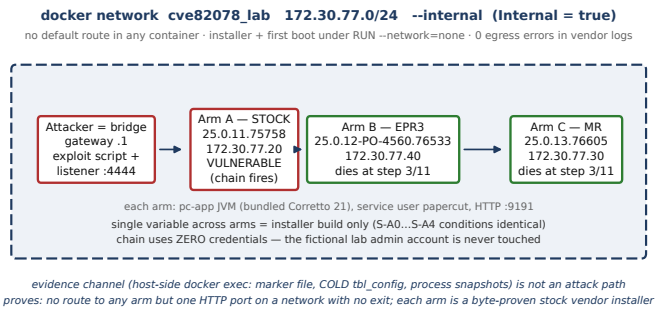


Fig. 1: Three-arm lab on one `--internal` Docker bridge (`cve82078_lab`, 172.30.77.0/24). The network is `Internal=true` (counted check), containers are single-homed with no default route, the vendor installer and first boot run under `RUN --network=none`, and vendor logs show zero `UnknownHost`/`ConnectException` lines. The only variable across arms is the installer build; the attacker sits on the bridge gateway. Evidence reads (`docker exec`, host-side) are a measurement channel, not an attack path. Generated figure (`evidence/figures/make_figs.py`).

TABLE II: Installer provenance (all fetched 2026-09-11; hashes recomputed by `claim_gate`). The stock build's integrity rests on our custody hash alone — the vendor publishes checksums only for the fixed builds. The first intake comparison of the patched arm reported MISMATCH because the wrong bulletin row (MF 76604) was read; re-deriving the NG row programmatically gave MATCH — a gate that failed correctly and is logged (attempt log #1).

Arm (build)	Size (B)	SHA-256 (12-hex head, full in `ioc/` §6)	Vendor hash verdict
stock 25.0.11.75758 (arm A, .20)	960057398	`644957717fcb...`	none published (custody)
EPR3 25.0.12-PO-4560.76533 (arm B, .40)	986902850	`68858979b4db...`	MATCH (Wayback row Linux/76533)
MR 25.0.13.76605 (arm C, .30)	986907259	`f21311fa3833...`	MATCH (bulletin NG/Linux/76605)

POST (browser-equivalent, not a bypass shortcut). *S-A1 database state:* stock `database.type=Internal` with empty `database.driver`; the chain pre-configures nothing — the exploit itself writes the `user-lookup.*` rows via the web bypass. *S-A2 fictional admin:* lab-only admin credentials (free-tier trial licence, $\leq$5 users, no registration — vendor licence rules respected [11]) exist for the product to be healthy; the exploit never uses them (zero-credentials is a testable property, not a claim). *S-A3 isolation:* as Figure 1. *S-A4 resources:* three arms $\approx$6.2/7.9 GB; fixed arms are stopped between phases and re-greened before use.

Integrity controls used throughout: `verify.sh` health gates (25 ok on stock / 24 ok on non-stock arms, which lack the flag-file check) run *before and after every* attack run; every evidence file carries `### cmd:/### at:` headers; and configuration

reads are *COLD* — the app-server is stopped, the Derby table is read `ij bootReadOnly`, then the server restarts and re-greens. (The first two warm `ij` attempts died on the live JVM's embedded lock and are kept in the attempt log, #8, as invalid — a warm read of a locked in-process database is not evidence.)

C. The agent pipeline

The case ran as a gated pipeline (intake → research → lab → exploit design/implementation → evidence → fix verification → detection → paper), one child agent per phase under a parent that re-ran gates independently rather than trusting child self-reports (e.g., the parent re-executed the lab gate to a `25 ok/0 fail` green of its own; `ab_gate` and `claim_gate` are scripts, not prose). Two rules did real work: (i) every state change requires a fresh re-verify before any green is citable — after a rebuild sequence destroyed a green container mid-run (attempt log #5), a self-report of "green" would have shipped a phantom; (ii) invalid attempts are kept, never rewritten (attempt log #2 records a confabulated sink-location line written *before* extraction and later replaced by byte-level evidence — nothing downstream had consumed it). C4's value is exactly this machinery made citable; Section VIII publishes the digest.

IV. Exploitation Evidence (C1)

A. The attack, step by step

Fig. 2 reproduces the chain as it happened on screen, captured during the live single-take recording (`videos/takes/t5-show.mov`, SHA-256 `f9c8b0f8fc5e...`, sealed in the attempt log) and *never* from the edited film, so no post-production element appears in any still. The seven times were located by optical character recognition (OCR) on raw frames before cutting, and the generator (`evidence/figures/make_stills.py`) re-runs that OCR gate on every final crop, so a caption cannot drift away from its screenshot (placement log: `evidence/figures/STILLS-OCR.md`). Step numbers match the chain anatomy of Fig. 4.

1) *Oracle* (R1, unauthenticated): one `curl` to the anonymous `Error` page prints the exact build, `PaperCut NG 25.0.11 (Build 75758)`, which is in the KEV-patched range, before any credential exists.
2) *Weapon*: the exploit is a single stdlib script, 1,910 lines. The excerpt shown is the payload's socket pump, an outbound connect and a `ProcessBuilder` of `/usr/bin/script -qc /bin/sh -i` (a pty that line-buffers the shell), drained with per-direction `available()` polling, the mechanism behind the per-command shell responsiveness evidenced in §IV (kill-shot subsection).
3) *Listener armed* on the gateway (`172.30.77.1:4444`) before the chain launches, so the callback cannot race the catcher.
4) *Callback* (T4): the victim calls back from its own address, `CONNECT from 172.30.77.20:50512`, the first outbound proof of code execution.
5) *Owned shell*: `id` answers `uid=1000(papercut)`; `whoami` answers `papercut`, and `cat /root/flag.txt` is refused. The shell sits exactly at the service account, no privilege bonus.
6) *Defacement re-rendered* (T6/T7): after writing the payload CSS and one service restart, the login page is reloaded from a blank tab and comes back serving the banner *from the server*, persistent content change on the anonymous surface.
7) *Full-control proof*: the flag `PCUT[25.0.11-stock-82078]` is read (lab evidence read over `docker exec`, our integrity channel, not an in-shell path), and the session transcript is SHA-256 sealed before the take ends.

B. Wire script

The exploit is one stdlib-Python script (`exploit/papercut_82078.py`, SHA-256 `eb8782e900b7...` pinned in this paper), no third-party dependencies, no attacker-side HTTP service. Its `--wire-dump` mode prints the exact requests deterministically (fixed PRNG seed, golden-hashed in `exploit/tests/golden/`, 24 request blocks) with no socket opened; Listing 1 shows the two shapes that matter: the anonymous version/session requests and one forged direct-service write pair. The whole dry-run chain is 11 logical steps / 24 HTTP requests on one keep-alive connection. A hard subnet guard (target must be a literal IPv4 in the lab subnet; no flag bypasses it, hostnames refused) is shipped and refusal-tested against out-of-lab addresses.

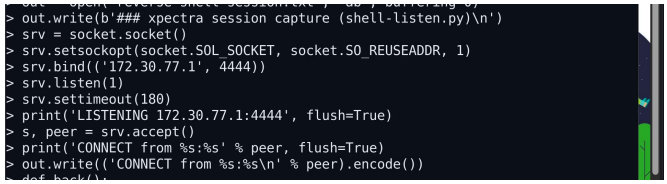
Listing 1: Wire script, exact golden-dump bytes (`exploit/tests/golden/wire-dump-dry-run.bin`; R1, R2, and the first forged write pair; `<slot>` is a fresh random per request). The `quickFindForm` request selects the config key; the `$Form` request writes the Derby driver name. Percent-decoded `%24` = the `$` of Tapestry's form grammar; the wire rule of Listing 4 keys on the encoding-agnostic substring.

```
----- WIRE DUMP [R1] (162 bytes, NOT SENT) -----
GET /app?service=page/Error HTTP/1.1
Host: 172.30.77.20:9191
Origin: http://172.30.77.20:9191
Referer: http://172.30.77.20:9191/app
Connection: keep-alive
----- WIRE DUMP [R2.h0] (143 bytes, NOT SENT) -----
GET /app HTTP/1.1                    [ ...headers as above
    ... ]
----- WIRE DUMP [R3.driver.qf] (365 bytes, NOT SENT) -----
POST /app?service=direct/<slot>/Home/ConfigEditor/
    quickFindForm HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 96

sp=S0&Form0=%24TextField%2CdoQuickFind%2Cclear&%24
    TextField=user-lookup.db-driver&doQuickFind=Go
----- WIRE DUMP [R3.driver.set] (389 bytes, NOT SENT)
    -----
POST /app?service=direct/<slot>/Home/ConfigEditor/$Form
    HTTP/1.1
Content-Length: 127

sp=S1&Form1=%24TextField%240%2C%24Submit%2C%24Submit
    %240&%24TextField%240=org.apache.derby.jdbc.
    EmbeddedDriver&%24Submit=Update
```

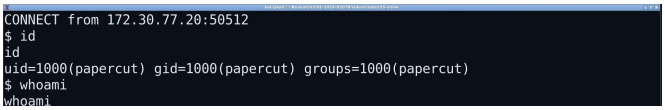
C. Proof-of-execution payload anatomy

The host carries no `javac` (JRE-only JDK 25), so the payload is hand-assembled: a pure-`struct` ClassFile writer emits

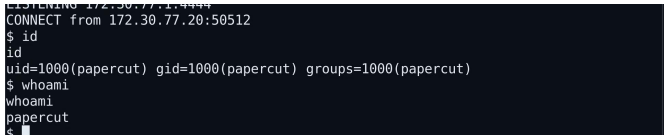
(a) Step 1: version oracle



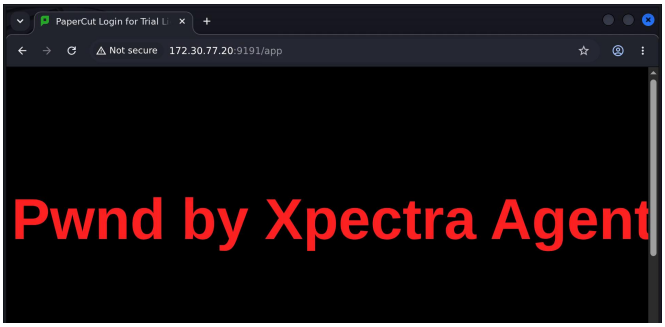
(b) Step 3: listener armed



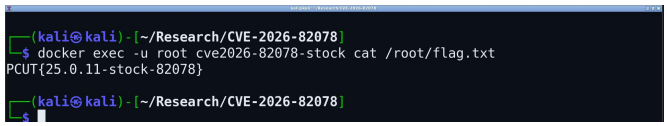
(c) Step 4: callback from the victim



(d) Step 5: shell as `papercut`



(e) Step 6: defacement served by the victim



(f) Step 7: full-control flag

Fig. 2: The attack on screen: six of the seven stills (step 2's code excerpt is Fig. 3), cut from the raw recorded take `t5-show` with no editing between take and figure. Rows: steps 1/3, 4/5, 6/7. Each frame is OCR-gated against the string quoted in its caption. Screenshots from our own isolated lab: attacker `172.30.77.1`, victim `172.30.77.20`.

TABLE III: Closure matrix — same unmodified exploit script (SHA-256 head `eb8782e900b7`), same day (2026-09-11, UTC), same payload (vocabulary note: the proof-of-execution PoC artifact appears in raw captures under its internal capture name); `--dry-run` against all three arms. The two fixed arms die at the *identical* step; the sink leg (82078) is never reached on either. Artifacts: `evidence/phase05/run-0[6-9]-*` + `evidence/pcap/`.

Leg	arm A: stock 25.0.11.75758	arm B: EPR3 25.0.12-PO-4560.76533	arm C: MR 25.0.13.76605
R1 version oracle (public page)	200 — NG 25.0.11/75758	200 — NG 25.0.12/76533	200 — NG 25.0.13/76605
R2 anonymous Home + JSESSIONID	200	200	200
R3 first forged `service=direct` POST	**200** — listener runs un-authed (81578 holds)	**302** `Location: /app?service=page/Home`	**302**, identical
R4–T10 (config writes, SYSCS drop, classload)	run	*never reached*	
marker `/tmp/cve2026-82078-canary`	PRESENT, `papercut`:644	ABSENT (host-side check)	ABSENT (host-side check)
COLD `tbl_config` pre vs post	restored-to-stock, byte-identical	byte-identical (untouched)	byte-identical (untouched)
run.json outcome	`success=true`, rc 0	`error: ...(status 302)`, rc 1	same error, rc 1

a valid Java-8 (major 52) class with explicit constant pool, one `<clinit>` and one `<init>`, and the StackMapTable frames the verifier demands at branch and handler targets. The dry-run payload is **977 bytes**; its `<clinit>` writes the case marker string to the marker path under `/tmp/` via `Files.write`, deleting its own `lib/<B>.class` and `tmp/<c>.csv` in a `finally`, and throws `RuntimeException` *only* if the write failed — the success path throws nothing, and the class need not implement `java.sql.Driver` (the connect failure after `Class.forName` is caught and logged by the application; the response still renders 200, which is itself the wire signal SID 202682047 keys on). Byte identity under a fixed class name makes the artifact testable: the golden build is diffable and the `--selftest-classfile` mode builds and parses back every shape.

D. Dry-run matrix and restore-to-stock (three runs)

Three fresh runs against arm A (`run-01/02/03`, ≈70 s apart) each: success `true`, `teardown_errors = []`, the proof-of-execution marker file present at `/tmp/cve2026-82078-canary` as `papercut`:644 with the marker content, and — after each run — **COLD** `tbl_config` reads byte-identical to the factory baseline on all seven `user-lookup.*` keys. Every run uses fresh random names (class, CSV, memory-DB), so idempotency was proven against a changing artifact, not one. Post-run `verify.sh` shows *exactly one* expected failure (the marker-absence checks are a post-exploit state the pre-attack gate cannot pass); this honesty

Fig. 3: Step 2: the exploit's reverse-shell pump as it sat in the take, the payload the agent wrote (SHA-256 `eb8782e900b7...`) opened at the relay loop: connect, pty via `script`, two-way `available` pump, single clean exit.

TABLE IV: Latency, in milliseconds, as recorded by the tool itself in `run.json` (never hand-copied): `T1−T0` = SYSCS export-ack (second lookup that runs the export SQL), `T3−T2` = driver-swap class-load ack — i.e., configuration write to `<clinit>` complete. Across all nine measured runs the class-load ack spans 15.2–41.6 ms; the export ack (which includes Derby memory-DB boot) spans 89.1–501.3 ms. The marker file is observed in-band with the same ack (no out-of-band race assumed).

Run	mode	class B	T1−T0 export ack	T3−T2 load ack
run-01 (16:56Z)	dry-run	977 B	269.0	17.0
run-02 (16:57Z)	dry-run	977 B	501.3	26.4
run-03 (16:58Z)	dry-run	977 B	396.5	18.2
run-04c-a (17:12Z)	dry-run	977 B	135.6	15.2
run-04c-b (17:12Z)	reverse-shell	1705 B	121.1	27.1
run-07 (17:52Z, positive control)	dry-run	977 B	358.1	22.1
run-09 (17:57Z, positive control)	dry-run	977 B	498.9	24.5
run-06-p06-a/b (18:35Z)	dry + shell	977/1705 B	496.0 / 89.1	27.3 / 41.6

detail is kept in the run record rather than hidden by reordering the gate.

E. Kill-shot: reverse-shell process tree

The same chain with the connect-back variant (refused by the tool unless a dry run preceded it in-session) landed a shell inside arm A at 17:12:58Z — 1.47 s after the harness listener bound. The payload's own `ps -ef` (captured over the shell, and cross-checked by host-side polls of the same window) is the strongest single artifact of execution:

Listing 2: Measured process tree (abridged; `evidence/processes/run-04c-*-tree.txt` + `run-04c-*/reverse-shell-session.txt`, transcript 13017 B). The application JVM (`comm pc-app`, Tanuki wrapper + `AppServer` in its command line) spawns the interactive shell; the shell's grandchild is our evidence command. `uid=1000(papercut)`, CWD `server`. A fresh phase-06 re-fire reproduced the identical lineage with drifted PIDs (2096/1127) — shape constant, PIDs are not.

```
$ id; pwd
uid=1000(papercut) gid=1000(papercut) ...
/usr/local/papercut/server
  PID  PPID USER     COMMAND
 8694     1 papercut ./app-monitor [conf]
 8696  8694 papercut .../jre/bin/pc-app [flags
    ...]
   ...wrapper.WrapperSimpleApp
        biz.papercut.pcng.server.AppServer
11666  8696 papercut /bin/sh -i  <- payload
11676 11666 papercut ps -ef      <- evidence
```

V. Fix Verification (C2): Three-Arm Closure and Mechanism Location

A. The closure matrix

Table III states the measured result: the chain *fires* on 25.0.11 and *dies at step 3/11* on both fixed arms — the forged direct-service POST from an unauthenticated session is bounced (`302 → Location: /app?service=page/Home`) before any config write exists, so the 82078 sink leg is never reached on either arm. The 302 was verified in the raw-packet capture (server-side tcpdump ascii), not only in client-side logs, and the POST body carrying the sink payload string `user-lookup.db-driver` is present in *both* fixed arms' pcaps — the bytes arrive; the authorization layer refuses

TABLE V: Detection master table — every shipped rule, its label class, its measured fire(s) against real evidence (recorded pcaps/log windows plus fresh phase-06 live runs), and the named negative control that returned zero. Attempt-level rules fire on PATCHED arms too (bytes arrive); execution-level rules require the payload to have run. Transcripts: `evidence/rule-fires/`; full ledger `detection/sigma-fires.md`.

Rule	Class	Fires on (count)	Negative control → 0
SID 202682041 forged direct POST naming `user-lookup.key`	attempt	dry-run pcap ×10, rshell ×20, fresh live ×10/×20; **patched negctl ×1**	benign GET pcap → 0 (health SID alive)
SID 202682042 forged `$Form` setter (driver/JDBC values)	stage	×20 + live	patched pcap has no `$Form` setter bytes
SID 202682043 SYSCS export SQL config write	stage	×2 + live	patched arm: zero SYSCS bytes (grep-verified)
SID 202682044 `VALUES CAST(X'CAFEBABE...'` card value	stage	×2 + live	patched arm: no such bytes; benign 0
SID 202682047 forged direct POST answered **200** (bypass ACCEPTED)	execution (wire)	×1 per attack run	both fixed arms → 0
SID 202682045 SYN server → listener ports	execution (wire)	×1 per reverse-shell run	dry-run & fixed arms → 0
SID 202682046 forged POST answered **302** page-bounce (BLOCKED)	blue info	fires exactly on the two fixed-arm negctls ×1	n/a
Sigma `serverlog_cardid_hexblob` (server.log sink line)	execution (host)	24 hits incl. fresh live events	negctl stream (fixed arm) → 0
Sigma `derby_memory_boot`	execution (host)	9 hits incl. LIVE-caught boot	stock *file-dir* boot (no `memory:`) → 0
Sigma `pcapp_sh_child` (process lineage kill-shot)	execution (host)	4 hits (fresh run + run-04c lineage)	16 benign proc events → 0
Sigma catch-all (pc-app → shell/net tools)	execution (host)	6 hits	negctl stream → 0
Sigma `tblconfig_system_churn` (+ burst corr.)	slow signal	12 + 1 correlated alert	negctl stream → 0 (cannot fire)
YARA remnants (6 rules)	post-forensics	attack log/pcap windows incl. fresh runs	**clean idle** `server.log`/`derby.log` windows; benign pcap

them. This is consistent with the vendor FAQ's "EPR3 users are protected" and the Metasploit header's "v2 verified to remediate", and *refutes nothing public*: every published bypass claim concerned EPR1, not EPR3 or the MR.

B. Mechanism location without decompilers

No fix commit is public (closed source; the CVE Record's only reference is the advisory). Per operator ruling we used differential *binary* inspection only: extraction, hashing, and `strings` — no decompiler was run, and we claim nothing about internal code beyond shipped strings. Three artifacts locate the mechanism:

1) **Full jar manifest (345 `server/lib/*.jar` per arm):** stock→fixed moves exactly the vendor's own stems (`pcng-server*`, install-helper, log4j 2.25.3→2.25.5); the vendored `tapestry-3.0.4.jar` and `tapestry-contrib` are SHA-256-*identical across all three arms* — the fix is *not* in the engine.
2) **Per-class hash sets of the web/authz jar set:** both fixed arms add the same three classes; two are `ValidatingDirectService` and `ValidatingActionService`, byte-identical EPR3 = patched. Shipped strings show the mechanism: `extends org/apache/tapestry/engine/DirectService`, a `validate(IRequestCycle)` member, `ownerPage`/`OWNER_PAGE_INDEX` members (it re-validates the *owner* page of a direct-service invocation), and on the page base class a rejection string `'Rejected post-validation access to page {}' from {} with IP: {}'` plus `PageRedirectException` — the 302 we measured, from the bytes.
3) **The shipped application spec (decisive):** the WAR's `WEB-INF/papercut.application` is the service-to-class wiring the engine loads at boot. Stock (`aa6240c2...`) has *no* override of the `direct` service; EPR3 and the maintenance release ship the identical spec (`f4bdbf5d...`, byte-identical pair) adding two overrides (Listing 3); runtime copies inside all three containers byte-match their shipped specs.

The honest negative travels with the positive: the advisory phrase "complex direct" has **0 hits** across every entry of every extracted jar and the WAR specs of all arms — it is third-party vocabulary, and the shipped-string evidence is Listing 3 plus the `Validating*` classes. The fixed MR additionally drops three sink-side `user-lookup` keys in its external-lookup class (defense-in-depth on leg 2) — EPR3 closes the chain without it, so we classify that as hardening, not the kill.

Listing 3: The fix, located by shipped-spec diff alone (`evidence/phase05/static/` wp4c): stock adds nothing; EPR3 and 25.0.13 ship the *same* two service overrides; engine jars identical across arms. This is a mechanism *location* argument from bytes the vendor ships — not a source-code claim.

```
--- war-stock/WEB-INF/papercut.application
+++ war-epr3/WEB-INF/papercut.application    (=
    war-patched, byte-identical)
@@ -11,6 +11,8 @@
     <library id="contrib" specification-path="
    ...Contrib.library"/>
+    <service name="direct" class="biz.papercut.
    pcng.web.
+       services.ValidatingDirectService"/>
+    <service name="action" class="biz.papercut.
    pcng.web.
+       services.ValidatingActionService"/>
     <service name="custom-content" class="...
    CustomContentService"/>
```

C. Unit corroboration: the fix closes the path, not the artifact

A 302 proves the request was refused; it does not prove the sink *would* have failed. To separate path-closure from artifact-removal we ran the sink's own call shape *inside the fixed arm*: the exact `--generate-classfile` payload bytes (980 B, major 52, SHA-256 head `43d60b84...`) were placed in a scratch directory of the EPR3 container and loaded with a 20-line loader calling `Class.forName(NAME)` under the arm's *bundled* Corretto-21 JRE. The class initialized: `<clinit>` ran, the marker file appeared with content and SHA-256 (`aeba89c3...`) identical to the stock-arm fire. An ordered negative control ran first (empty slot + absent class name → `ClassNotFoundException`, slot stays empty), and cleanup was verified. Two failures precede this result and are kept in the attempt log: the host JDK-25 `javac` could not emit the loader (no `--release` data for ≤21 targets), so the loader was compiled *inside* the container with the vendor-shipped `ecj-3.33.0.jar` under the bundled JRE (#14, both the failed and successful commands archived); and a first-pass negative control checked the slot *after* the positive load, proving nothing, and was superseded by the ordered reproof (#15). Caveats: the loader ran as root in a scratch dir (not as the `papercut` JVM) — owner-`papercut` execution is banked from the stock-arm live runs — and the class "runs its proof-of-execution body when loaded" on the fixed build; what the fix removes is the *load path* the web chain needs. Generalizing: for closed-source targets, behavioral closure + shipped-artifact diff + in-situ unit corroboration together locate a fix and characterize what it does *not* change ("path, not artifact"), without any decompilation.

D. Why the negatives are not vacuous: five pillars

`ab_gate` re-derived the fix-verification evidence from artifacts and returned `verdict=VALID checks=10 pillars_proven=5/5` (`evidence/phase05/ab_gate-output-*.txt`): *monitor-alive* — both fixed arms answered benign GETs with 200 through each exploit window, `gc.log` grew during it, `verify.sh` green before and after; *witnessed-inert* — the sink payload string is present in the *server-received* pcap bytes of both fixed arms while the marker file is absent and the config is byte-identical (honesty note: on these arms the chain dies before the sink's own logger can speak, so inertness is wire-level + zero-effect; payload-level inertness is instead isolated by §V-C); *unit-corroboration* — §V-C; *positive-control* — the stock arm fired the same payload with the same script *the same day*, bracketing each negative within minutes (EPR3 negative 17:46:42Z → stock positive 17:52:34Z; patched negative 17:54:51Z → positive 17:57:1xZ): environment drift is excluded by time, not by assertion; *config-diff-only-fix* — COLD full `tbl_config` and `server.properties` across all three arms: every changed key classifies as install-noise or build-identity (10/959 and 17/959 rows; the 7 extra patched rows are `user-lookup.*` keys the MR never seeds), and `server.properties` differs by 1/23 keys per pair (the per-install random bcrypt admin hash) — no functional divergence other than the fix could explain the behavioral difference.

VI. Detection (C3)

Doctrine: a rule that cannot be fired against real evidence is deleted, not shipped; every row of Table V carries a fire transcript and a named negative control. The attack corpus is the phase-04/05 recorded pcaps and log windows plus three *fresh* live fires produced by the detection phase itself (reverse-shell run, dry-run, and a live-caught Derby memory-boot window). Tools: Suricata 8.0.6, YARA 4.5.8, pySigma 1.5.0 (parse/translate validation: 6 ok / 0 failed) with firing performed by our own stdlib Sigma-semantics evaluator over 96 events built from real evidence (Sysmon-style + ECS field aliases replicated), *not* a SIEM engine — a caveat stated on every transcript header.

A. Attempt versus execution: measured, not claimed

The fix is authorization-side: the forged bytes still *arrive* on patched arms (Section V). So request-content rules are labeled `attempt` and **proven** to fire on a patched arm (SID 202682041 fired ×1 against the phase-05 patched negctl pcap), while the response-pairing rule (forged direct POST answered `200` + `<-- Page: Home -->`) states the stronger `bypass-accepted` fact and returned `0` on both fixed arms, whose 302s instead fire the informational `BLOCKED` SID. On the host side the process-creation rule cannot fire on a patched arm at all (the payload is never loaded). The wire side honestly cannot claim host execution: the class drop is filesystem, not wire.

Listing 4: Full attempt-level wire rule (SID 202682041) exactly as shipped in `detection/rules/suricata/cve2026_82078_suricata.rules` (8.0.6 `-T` clean; `$PCUT_PORTS` etc. from the lab yaml — this Suricata build rejects in-rule `var` declarations, measured). The msg's parenthetical carries the honest scope: fires identically on patched arms.

```
alert http $EXTERNAL_NET any -> $HOME_NET
    $PCUT_PORTS ( \
```

```
    msg:"CVE-2026-82078 PaperCut attempt - forged
    service=direct ConfigEditor\
 quickFind POST naming user-lookup config key (
    attempt: fires on patched\
 arms too)"; \
    flow:to_server,established; \
    content:"POST"; http_method; \
    content:"service=direct/"; http_uri; \
    content:"Home/ConfigEditor"; http_uri; \
    content:"TextField=user-lookup.";
    http_client_body; \
    flowbits:set,pcut_82078_forged_direct; \
    classtype:attempted-admin; \
    sid:202682041; rev:1; \
    metadata:cve 2026_82078 2026_81578, stage
    attempt, severity high, \
 confidence high, created 09_11_2026, author
    Xpectra_Security_Research; )
```

Listing 5: The process-creation Sigma rule (abridged to detection logic; full file with aliases, false-positive notes and tags: `detection/rules/sigma/cve2026_82078_pcapp_sh_child.yml`; pySigma-validated). Fired 4× over real process snapshots incl. a fresh live reverse-shell run; 0 over 16 benign process events. Container debug (`docker exec`) parents under entrypoint/init, which axis 1 excludes — measured in clean-window snapshots, not assumed.

```
title: CVE-2026-82078 PaperCut JVM spawns interactive
    shell (kill-shot)
logsource: {category: process_creation, product: linux}
detection:
  pimg_pcapp_sysmon: {ParentImage|endswith: '/pc-app'}
  pimg_pcapp_ecs:    {parent_process.executable|endswith:
    '/pc-app'}
  pcmd_wrapper_sysmon:
    ParentCommandLine|contains|all:
      - 'org.tanukisoftware.wrapper'
      - 'papercut'
  pcmd_wrapper_ecs:
    parent_process.command_line|contains|all:
      - 'org.tanukisoftware.wrapper'
      - 'papercut'
  cimg_sh_sysmon:
    Image|endswith: ['/sh', '/bash', '/dash']
  cimg_sh_ecs:
    process.executable|endswith: ['/sh', '/bash', '/dash']
  cargv_i_sysmon: {CommandLine|contains|all: ['sh', '-i']}
  cargv_i_ecs:
    {process.command_line|contains|all: ['sh', '-i']}
  condition: >
    (1 of pimg_pcapp_* or 1 of pcmd_wrapper_*)
    and (1 of cimg_sh_*) and (1 of cargv_i_*)
level: critical
```

B. Rules we cut, and why (publishable negatives)

Three shipped-rule candidates were measured and deleted: (i) any rule keyed on `class load failed` text — the success path *emits none* (measured across attack windows; the expected sink log line is a caught `DatabaseUtils` cardID error, not a load failure); (ii) a wire rule claiming the class-file *drop* — at drop time there are no wire bytes (the CSV lands on disk; the self-deleting class lives seconds), so the filesystem signal is where that IOC honestly lives (IOC dossier §1); (iii) a SYSCS CSV content rule on the network — the CSV never traverses the wire (verified against the pcap corpus). Stated gaps we did not paper over: process rules were evaluated over the harness's own `ps` snapshots (weaker than kernel provenance; labeled in each rule), and no wire rule claims host-side execution.

C. Self-test detector with non-mutation proof

The package ships a stdlib audit tool whose `--audit` mode is passive (version threshold + an auth-free direct-shape probe, no form, no payload): against the stock arm it printed `VERDICT: VULNERABLE`, and its *non-mutation* was proven by COLD `tbl_config` reads before and after the audit — byte-identical including `modified_date`. Its `--active` mode refuses without a consent flag, refuses non-stock lab targets, and refuses out-of-lab addresses (all refusals transcripted); when authorized it re-ran the chain to a verified marker file, removed it (fresh-run rule), and left the lab at 25 ok / 0 fail.

VII. Remediation and Incident-Response Guidance

A. Patch levels — including the emergency-patch reality

- **Affected:** "from 0 before 24.1.10, 25.0.13, 26.0.5" — the CNA treats *all* versions as potentially impacted [3]. (NVD's CPE ranges disagree, treating the EPR lines and some 25.0.x/26.0.x as not vulnerable [4]; we record the divergence, cite both, act on neither in isolation.)
- **Emergency-patch history (from the vendor's own updates log, including Wayback captures of removed rows):** EPR1 2026-08-28 02:10 AEST (NG 25.0.12.76497 / MF 76496, plus 26.0.4 builds); EPR2 the same day for the v24/v25/v26 interim `-PO-4560` lines; EPR3 2026-09-01 18:22 AEST, explicitly "additional hardening and mitigation against potential attack chains"; maintenance releases **24.1.10 / 25.0.13 / 26.0.5** published **2026-09-10** and stated to "replace all emergency patches".
- **The EPR bytes are partly gone.** EPR1 and EPR2 installers now `404` on the CDN (our probes; the build table survives only in Wayback snapshots) — which is why our middle arm is EPR3, and why *we could not* test the third-party "EPR1 is bypassable" claim even though Metasploit's author tested against it [9], [8], [6]. What we measured is that the closure lands at EPR3: chain fires on 25.0.11, dies at 3/11 on 25.0.12-PO-4560.76533 and on 25.0.13.76605, identically.

CISO guidance: treat any build before an emergency patch as exposed; "patched with EPR1/EPR2" is *not* a defensible state (bytes withdrawn, bypass claim carried by three independent sources, KEV deadline 2026-09-14); **migrate to a maintenance release (24.1.10 / 25.0.13 / 26.0.5); if you cannot, be on EPR3 or later** — EPR3 is where we measured the path close. Verify your installer against the vendor-published SHA-256 (we demonstrated why an intake gate must read the *right* row: product NG vs MF differ).

B. Using the IOCs honestly

The IOC dossier tags every indicator `[LAB]` / `[GENERIC]` / `[VENDOR]` (`ioc/CVE-2026-82078.ioc.md`). `[LAB]` items — our marker path/sha, listener address, exact random class names — identify *our* runs, not attackers, and must never ship as threat-intel pivots. Hunt the `[GENERIC]` shapes: an eight-lowercase `.class` in `server/lib` appearing and vanishing within seconds (fs-watch/EDR, not scheduled scans); a

companion `tmp/*.csv`; `Booting Derby ...memory:` where the stock DB is file-directory; the `pc-app→/bin/sh -i` lineage; `tbl_config user-lookup%` rows churned by `modified_by=system` with no admin session. The vendor's own IOC list (server.log strings, Windows-side `data/content/*.cmd/.out` litter, the observed `whoami & ver` → SimpleHelp/AnyDesk escalation chain) and a GreyNoise-reported IP [10] complete the union — the vendor's caveat travels verbatim: "The absence of the above indicators is not confirmation that a system has not been affected."

C. IR hunt (measured signal strengths)

For triage: the server.log sink line is the highest-fidelity single artifact (our YARA/Sigma fired 24× on real windows, 0 on clean-stock windows); the wire pair tells you accepted-vs-blocked at a given instant; the process lineage is the critical alert (cannot false-fire on patched hosts by construction); the config-churn row is the slow tail for finding cleaned-up intrusions after the fast artifacts self-deleted. For compromise-vs-attempt on exposed `9191/tcp`: a 302 to `service=page/Home` in answer to a `service=direct` URL after 2026-09-01 is *the fix refusing you* — hunt the paired inbound attempt, not an incident.

VIII. Ethics, Reproducibility, and Limitations

A. Ethics

All exploitation ran in the fictional lab of Section III: a Docker `--internal` network with zero runtime egress, fictional domain and users, fictional admin credentials the chain never used, no third-party host touched, and the vendor's free ≤5-user licence rules respected. Dates, scores, and CVE/CWE wording come from the cited primary record (advisory 2026-08-27; CVEs 2026-08-28; KEV 2026-08-31, due 2026-09-14; MRs 2026-09-10); no discovery claim appears anywhere. The functional PoC ships per operator ruling with minimal proof-of-execution payloads, a hard subnet guard (no bypass flag), and an `--active` mode that refuses out-of-contract targets. Publication of the pack awaits explicit operator instruction while KEV exposure persists.

B. Honest limitations, from the attempt log

The case keeps an append-only attempt log (51 entries); the load-bearing ones:

- #1 the intake checksum gate's correct MISMATCH from a wrong bulletin row (Table II note); #2 an early sink-location line written before extraction was caught by self-audit and replaced by byte evidence;
- #5 a rebuild sequence destroyed a green container mid-session — after which the rule "no citable green without a fresh re-verify after every state change" was enforced, at the cost the paper now reports honestly;
- #8 warm `ij` reads of the locked embedded DB were invalid, forcing the COLD read procedure this paper depends on;
- #12 the first two reverse-shell evidence runs were harness-side capture failures (listener closed before the payload's output pump; listener crashed at bind) — the *chain* succeeded both times, the evidence was re-made on a third run, and the supersessions are logged, not deleted;
- #14/#15 the unit-corroboration loader build and negative-control ordering failed, were replaced by in-container ecj compilation and an ordered reproof;
- #16 a first Derby-boot race capture failed and is shipped marked `README-FAILED` (byte-cut windows with stale offsets) — never cited.

C. Closed-source and adaptation boundaries

We never decompiled; every fix statement is a hash, a diff of shipped bytes, or a shipped string (Section V-B) — we claim nothing about internal implementations beyond those artifacts. The reproduction is bound to its conditions: stock PaperCut **NG** 25.0.11.75758, Linux x64, free tier, one /24 lab network; the v26 H2/Groovy branch of the public exploit is *not* what we reproduced (ours is the v25/24 Derby path), EPR1/EPR2 could not be tested (bytes withdrawn), and the vendor's own claim that all versions are affected means arm A's "pre-patch" status understates the space of vulnerable installs we simply did not measure. Single-operator, single-host RAM/IO constraints (arm parking between phases) do not affect the chain/closure measurements, which were same-day and bracketed.

D. Reproducibility

The case pack is self-contained: lab definitions and gates (`lab/`), the exploit script with golden wire-hash tests (`exploit/`), every run's artifacts with command/time headers (`evidence/run-*`, `evidence/phase05/`), the rule set with fire transcripts (`detection/`, `evidence/rule-fires/`), IOCs, the attempt log, machine facts (`designs/facts.json`), and `claims.json` binding this paper's claims to those paths (checked by `claim_gate`). Figures are generated from the case data (`evidence/figures/make_figs.py` — box geometry is text-measured and asserted at build time); video stills are a phase-08 handoff tracked in `paper/figures-manifest.md`.

IX. Conclusion

For a KEV entry with real deployments and a merged Metasploit module, the defensively relevant open questions were measurement questions. We measured them: the chain works end-to-end on stock bytes with zero credentials (977-byte hand-assembled classfile; config restore cold-proven); both fixed arms kill it at the same authorization step, and the kill is located in the shipped service-override wiring without any decompiler; the payload artifact still executes on the fixed build, so remediation closes a path, not an artifact; and a detection package whose every shipped rule fired against real evidence — including one that honestly fires on patched hosts and three cut rules whose failure modes we publish. The methodology layer (cold reads, minute-scale positive bracketing, pillar-gated negatives, an append-only attempt log) is the reusable contribution: it is what makes each number above a measurement rather than an assertion.

REFERENCES

[1] PaperCut, "Security Bulletin (27 Aug 2026) — Urgent Security Advisory," page last updated 2026-09-10. Captured `evidence/raw-research/advisory.txt`.

[2] CVE Record CVE-2026-81578 (CNA: PaperCut), published 2026-08-28. Capture: `evidence/raw-research/cveawg_81578.json`.

[3] CVE Record CVE-2026-82078 (CNA: PaperCut), published 2026-08-28. Capture: `evidence/raw-research/cveawg_82078.json`.

[4] NVD entries for CVE-2026-81578 / CVE-2026-82078 (Analyzed; CVSSv3.1 primary 9.8 / 9.1). Captures: `evidence/raw-research/nvd_8*.json`.

[5] CISA KEV, catalogVersion 2026.09.10, feed SHA-256 recorded in `designs/facts.json`; both CVEs `dateAdded` 2026-08-31, `dueDate` 2026-09-14.

[6] Rapid7, "PaperCut NG/MF: Critical zero-day exploited in the wild," 2026-08-28. Capture: `evidence/raw-research/r7.html.txt`.

[7] Huntress, "PaperCut actively exploited," 2026-08-28. Capture: `evidence/raw-research/hunt.txt`.

[8] techanarchy.net, "From patch to exploit … (reverse engineering a zero-day in PaperCut NG)," 2026-08-30 — third-party EPR1-bypass claim, cited and attributed, not adopted.

[9] `metasploit-framework`, `papercut_ng_external_user_lookup_rce.rb`, merged 2026-09-03 (sfewer-r7), rank Excellent; captured sha256 `fc6a22a9...` as `evidence/raw-research/msf_module.rb`.

[10] GreyNoise reported scanner IP 45.142.193.132 (linked from the vendor advisory's IOC addendum of 2026-09-10, which itself states the vendor has not independently verified it).

[11] PaperCut, "NG Free Licence Questions" KB (free $\leq$5 users, no registration; last updated 2026-04-13).

[12] ARL Case 003 verified-facts machine record, `designs/facts.json` (provenance-keyed; merged 2026-09-11).

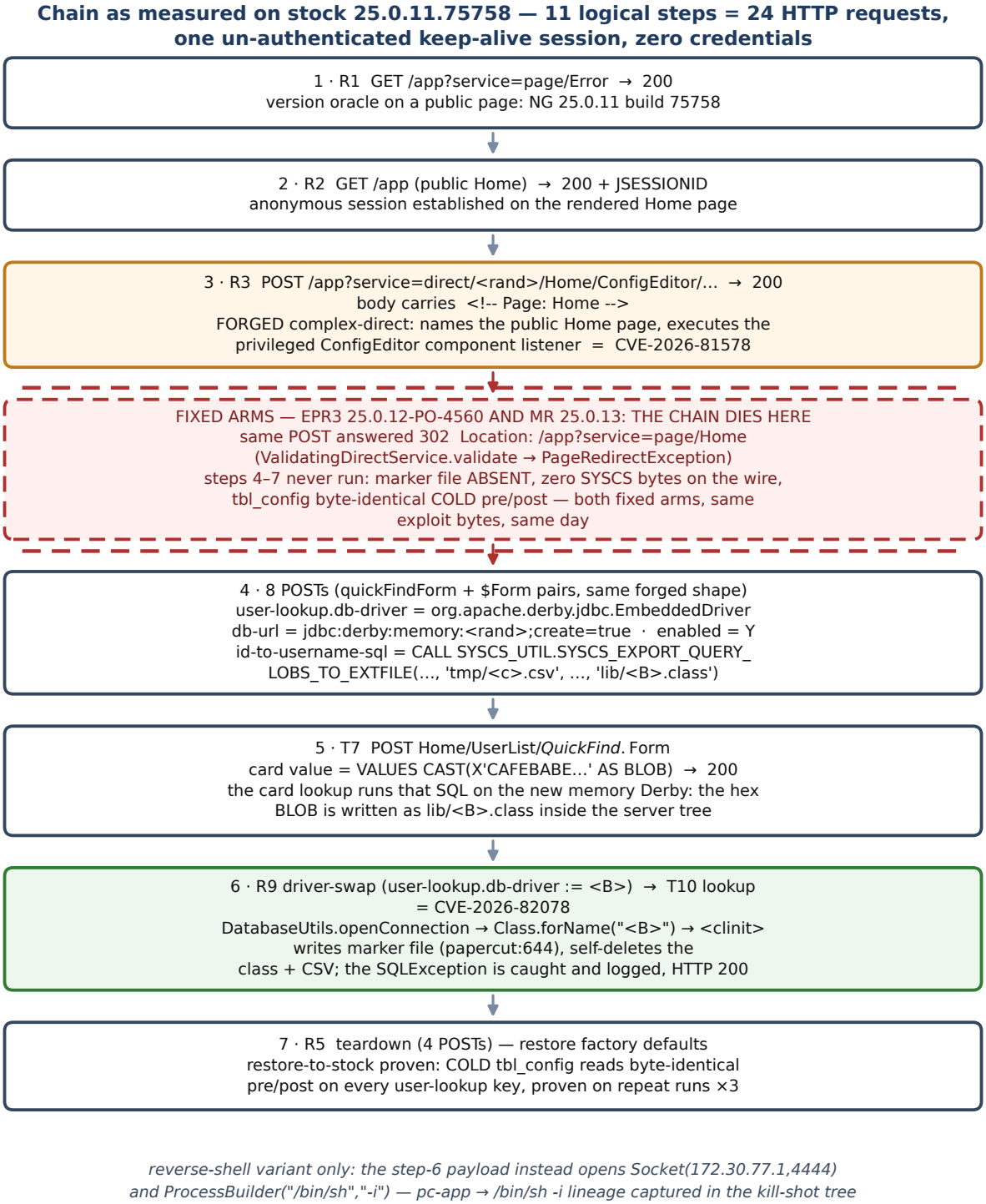


Fig. 4: The chain as measured, end to end, with the fixed-arm death path overlaid: steps and response status from the run logs and pcaps (`evidence/run-0*`, `evidence/phase05/run-0*`). The dashed red band is the 302 measured identically on both fixed arms — everything below it was never reached there. Generated figure; see also Table III.