Zawoo

Technical Analysis — Cti Report

REVERSE-ENGINEERED REPORT

RansomLook · ransomlook.io
File last modified: 2026-08-29

Sample SHA-256: `33d3afddaa5710cdcc4e93a7bae8be010c19747fb53d85fcd54ef32056a1eb0b`

ZAWOOO — CTI Report

Sample: `33d3afddaa5710cdcc4e93a7bae8be010c19747fb53d85fcd54ef32056a1eb0b` **Classification:** Ransomware, Linux/ELF64, NAS-specific (Synology DSM) **Confidence:** High — conclusions drawn from static reverse engineering of the binary only, cross-verified in a second disassembler (Ghidra) for the footer layout, the XTS key schedule and every function boundary in the malware range.

---

1. Executive Summary

ZAWOOO is a **dedicated Synology NAS encryptor**, not a general-purpose Linux locker. Every design choice in it points at one target class: the directory exclusion list contains only Synology internal shares, the default scan root is the `/volume*` glob, and the fastest execution mode consumes a pre-built file index laid out under `/etc/f-index`.

The strategic point of interest for defenders is not the cryptography — which is correct and unbreakable without the operator's key — but the **two-stage encryption model**. A first "silent" run destroys file contents in place while leaving filenames intact and dropping no ransom note. The estate looks completely normal. Backups keep running and keep capturing already-encrypted data. A second run, whenever the operator chooses, recognises its own work from an 8-byte footer magic and does nothing but rename files and drop notes — turning an invisible compromise into a visible ransom event across a whole NAS in seconds.

This inverts the usual detection assumption. By the time ZAWOOO is visible, encryption is already old. The detection opportunity is entirely in the silent window, and it is a behavioural one: sustained strided read/write across large files, two new directories under `/etc`, and a lock file in `/var/run`.

Countering that, the payload itself is technically unremarkable: no packing, no obfuscation, no anti-debug, no anti-VM, no C2, no persistence, and a full debug logger with source filenames left compiled in. This is a well-thought-out playbook running on a mediocre codebase.

---

2. Actor Assessment

What the code says about the operator

Observation	Assessment
A 16-hex victim ID compiled into the note template (redacted here)	**One build per victim.** The operator runs a builder and produces a bespoke binary per intrusion. Expect every sample to have a different RSA key, suffix and victim ID.
Per-build encryption suffix (`fHiQJervS`), with `SpXv2` / `MXvF0` visible as leftovers	Confirms at least **three distinct builds** of this family exist. Suffixes are random-looking 5–9 char tokens, so suffix-based detection will not generalise.
LockBit's Windows extension exclusion list embedded verbatim, typos included	The builder's configuration was **copied from a LockBit-derived toolkit** and never adapted. `.deskthemepack` , `.diagcab` , `.msstyes` are meaningless on DSM. Suggests an affiliate or a developer working from leaked/borrowed builder material rather than writing from scratch.
Two mutually inconsistent copies of the exclusion list left in the binary	Weak build hygiene; the codebase is edited by hand between builds and not regression-tested.
Compiled-in typos in filesystem paths (`/etc/slient-time`)	The path is part of the protocol between stages, so the typo is stable across builds and is a durable IOC.
`FTI_LOG` debug logger with `src/enc/thread.c` / `src/enc/work.c` and line numbers	The shipped payload is a **debug build**. Either the operator does not care, or they actively use the log to verify coverage during an engagement.
`-x` referenced in an error message but not implemented	Shared codebase across variants; feature flags come and go between builds.
GCC 11.2.1 20211120, static-pie, no `DT_NEEDED`	Deliberately built for **appliance portability** — it will run on any x86-64 DSM regardless of the libc version shipped by Synology. This is a considered choice, not an accident.

Note rhetoric

The ransom note is a **reputation-and-regulator** play rather than a technical one. It leads with "a ransomware that prioritizes reputation", explicitly threatens notification of customers and of "the GDPR (Data Protection Authorities)", frames the ransom as cheaper than a breach fine, offers an intrusion debrief as a "security test", and coaches the victim at length on how to buy Bitcoin without alerting a broker. It also explicitly instructs the victim not to contact police or the FBI.

This is European-facing extortion targeting organisations that hold customer data and are subject to GDPR — SMEs, MSPs, professional services, healthcare. The negotiation channel choice (Session + OnionMail, "send the message in two ways at the same time") indicates an operator expecting channel loss and running deliberately low-infrastructure.

Capability ceiling

The binary contains **no networking code at all** — no `connect` , no `sendto` , no C2. The note's exfiltration threat is therefore **not backed by this sample**. If data was stolen, it was stolen by separate tooling during the intrusion. Treat exfiltration claims as unverified-by-payload; verify independently from egress telemetry rather than assuming.

Likewise there is no lateral movement, no credential access, no privilege escalation and no persistence in this binary. ZAWOOO is a **terminal-stage payload** dropped by hand onto an already-compromised appliance. Root (or at least write access to `/etc` and `/var/run`) is a prerequisite.

3. Attack Lifecycle

```
[ Initial access — not in this sample ]
        Internet-exposed DSM / QuickConnect / weak admin creds / n-day in DSM
                 |
                 v
[ Staging — partially inferred ]
        Operator builds /etc/f-index/<YYYY>/<MM>/index.dat inventory
        (records: u64 mtime | u64 size | u16 namelen | path)
        Writes /etc/f-index/baseline.stamp
                 |
                 v
[ Stage 1 — SILENT ]   ./enc -p /volume1 -s
        Creates /etc/slient-time/baseline.stamp (guarded clock, T)
        Encrypts file contents in place
        NO rename. NO ransom note. NO extension.
        -> NAS appears completely normal; backups capture ciphertext
                 |
                 v   ( operator-chosen delay )
[ Stage 2 — DETONATION ]   ./enc -p /volume1 -a      (or no flag)
        flock /var/run/enc-nas.lock
        Reads T from /etc/slient-time/baseline.stamp
        For each file with mtime >= T: footer magic present -> RENAME ONLY
        For everything else: full encrypt + rename
        Drops "How To Restore Your Files.txt" per directory
        -> whole estate visibly ransomed in the time of a rename() sweep
```

Targeting priority

`-a` mode does not scan the filesystem — it replays the index in an explicit newest-first order, logged by the binary itself:

```
phase 0 = _other (year>2026, newest)
phase 1 = month buckets 2026/12 .. 2000/01
phase 2 = _other (year<2000, oldest)
```

In parallel, a second producer `glob()`s `/volume*` and queues anything modified since the index was built. The consequence for defenders: **the data the business touched most recently is destroyed first**. A response that starts an hour into the encryption run has already lost the current working set. Speed of containment matters more here than for a locker that walks directories alphabetically.

4. Impact Assessment

Dimension	Finding
Reversibility	None without the operator's RSA-2048 private key. Fresh 32-byte key per file from `/dev/urandom`, no key reuse, no PRNG weakness, no plaintext key retained on disk, key present only inside the RSA blob in each file's footer.
Filename recovery	Also impossible without the private key — the basename is XTS-encrypted with the same per-file key and the original is destroyed by `rename()`. Directory structure survives.
Data integrity	Partial by design. Only 6–25 % of each file is ciphertext, but it is always the head and it is spread at a fixed stride, so container/archive/database formats are unusable. Do not assume "only 6 % encrypted" means "94 % recoverable".
Speed	Very high. 512 KB per 8 MB on files ≥ 64 MB, three worker threads, index-driven ordering. A multi-terabyte NAS is destroyed in a fraction of the time a full-encryption locker would need.
Availability of DSM	Preserved deliberately. Synology system directories are excluded so the appliance stays bootable and the web UI reachable — the operator wants the victim able to read the note and negotiate.
Backups	**Primary risk.** Any backup taken during the silent stage contains ciphertext with valid-looking filenames. Backup jobs will report success. Snapshot retention windows may be entirely poisoned before anyone knows an incident occurred.
Recovery paths that survive	Synology/Btrfs snapshots and off-box replicas are not touched — the binary contains no snapshot, backup or log tampering whatsoever. Immutable/off-appliance copies predating the silent stage are the realistic recovery route.

5. Detection Engineering

File-based — encrypted artefacts

Every encrypted file ends with a 12-byte tail: a `u32` footer length followed by the 8-byte magic. This is exact, cheap, and independent of the per-build suffix.

```
rule ZAWOOO_Encrypted_File
{
    meta:
        description = "File encrypted by ZAWOOO - footer magic"
    condition:
        filesize > 268 and
        uint32be(filesize - 8) == 0xDFB307AC and
        uint32be(filesize - 4) == 0x09272154
}
```

Because silent mode does not rename, this rule is the **only** file-level way to detect stage-1 compromise. Run it across NAS shares and, critically, **across backup sets**.

It does not cover the earlier file format. Dead-but-present code in this sample (`0x83E0`, `0xB740`) shows a prior build that appended a bare 256-byte RSA blob with **no magic and no length field**, and that **kept the original filename** with the suffix appended (`report.xlsx.<suffix>`). To hunt those, look for double extensions where the second component matches a short random-looking token, and for files whose last 256 bytes are high-entropy while the rest of the file is not. Since the builder still ships this code path, a future build could re-enable it — do not treat the magic as a permanent family invariant.

File-based — the payload

```
rule ZAWOOO_Linux_NAS_Encryptor
{
    meta:
        description = "ZAWOOO Synology NAS ransomware (ELF64 static-pie)"
    strings:
        $magic   = { DF B3 07 AC 09 27 21 54 B1 E1 3B 58 7F 52 13 DE }
        $lock    = "/var/run/enc-nas.lock"              ascii
        $slient  = "/etc/slient-time/baseline.stamp"    ascii
        $findex  = "/etc/f-index/baseline.stamp"        ascii
        $note    = "How To Restore Your Files.txt"      ascii
        $log1    = "src/enc/thread.c"                   ascii
        $log2    = "src/enc/work.c"                     ascii
        $usage   = "usage: %s -p path"                  ascii
        $typo1   = "alredy run(-x ), exit"              ascii
        $typo2   = "xts encrypt fail: -0x%04X"          ascii
        $syno    = "@SynoFinder-etc-volume"             ascii
    condition:
        uint32(0) == 0x464C457F and filesize < 4MB and 3 of them
}
```

The `src/enc/*.c` strings and the `slient` typo are the most build-durable anchors: they are compiled-in paths and log format strings, not configuration, so they should survive across builds that change the RSA key, suffix and victim ID.

Host telemetry

Signal	Rationale
`/etc/slient-time/` created (mode 0755)	Not a Synology path. Its existence alone means stage 1 ran or is running. **Highest-value single indicator.**
`/etc/f-index/` created, containing `<YYYY>/<MM>/index.dat`	Pre-encryption staging. Detecting this catches the intrusion before any file is touched.
`/var/run/enc-nas.lock` present, or an exclusive `flock` held on it	Stage 2 in progress.
`How To Restore Your Files.txt` written to many directories	Stage 2 complete. Late signal.
Mass `rename()` matching `^[0-9A-F]{16}\.[A-Za-z0-9_]{3,12}$`	Stage 2 in progress.
One process, sustained `open(O_RDWR)` then `lseek` / `read` / `write` pairs at a **fixed stride** (4 MB or 8 MB) over many large files	Stage 1 in progress — the only high-fidelity behavioural signal during the silent window.
`sh -c "date +%s"` spawned by a non-shell long-running binary	Clock source of last resort; unusual on an appliance.
Repeated `open("/dev/urandom")` at file-processing cadence	One open per encrypted file; a distinctive rate signature.

Network telemetry

None from the payload. Do **not** build detections expecting ZAWOOO C2 — there is none. Focus network hunting on the intrusion that delivered it (DSM admin authentication from unusual sources, QuickConnect/relay usage, SSH enablement) and on egress that would indicate the separate exfiltration tooling the note implies.

---

6. Hunting Guidance

Run these on Synology appliances and any Linux host with NAS shares.

```
# 1. Stage-1 marker directories — the two highest-value checks
ls -la /etc/slient-time/ /etc/f-index/ 2>/dev/null
cat /etc/slient-time/baseline.stamp 2>/dev/null   # epoch: when silent mode started

# 2. Stage-2 lock
ls -la /var/run/enc-nas.lock 2>/dev/null
fuser -v /var/run/enc-nas.lock 2>/dev/null

# 3. Silently encrypted files — footer magic, name unchanged
find /volume1 -type f -size +268c -print0 2>/dev/null \
  | xargs -0 -P4 -I{} sh -c \
    'tail -c 8 "$1" | od -An -tx1 | tr -d " \n" | grep -q "^dfb307ac09272154$" && echo
"ENCRYPTED: $1"' _ {}

# 4. Renamed files (stage 2)
find /volume* -type f -regextype posix-extended \
     -regex '.*/[0-9A-F]{16}\.[A-Za-z0-9_]{3,12}$' 2>/dev/null | head

# 5. Ransom notes
find /volume* -name 'How To Restore Your Files.txt' 2>/dev/null | head

# 6. Any process holding many large files open read-write
ls -l /proc/*/fd 2>/dev/null | grep -E '/volume[0-9]' | head -50
```

If step 3 returns hits while steps 4 and 5 return nothing, **you are inside the silent window**. That is the single most valuable outcome of this hunt: contain immediately, and treat every backup taken since `/etc/slient-time/baseline.stamp` as suspect.

---

7. Response Guidance

1. **Isolate the appliance at the network layer before anything else.** Do not shut it down cleanly if stage 2 is running — a graceful stop lets in-flight writes complete. Pull the network first, then decide.
2. **Read `/etc/slient-time/baseline.stamp`.** It is a plain epoch. It tells you when silent encryption began, which directly scopes the backup poisoning window.
3. **Do not restore from any backup taken after that timestamp** without first validating it with the `ZAWOOO_Encrypted_File` rule. Backups from the silent window contain ciphertext under correct filenames and will restore cleanly and uselessly.
4. **Check `/etc/f-index/`.** Its `index.dat` files are a complete inventory of what the operator enumerated — effectively a free scoping document for the incident, including the size and mtime of every file they intended to encrypt.

5. **Preserve, do not delete, encrypted files.** Each footer holds the RSA-wrapped per-file key; if the private key is ever recovered by law enforcement, those files are decryptable. Deleting them forecloses that option permanently.
6. **Check Btrfs/Synology snapshots and off-box replicas.** The binary does not touch them. If snapshots predate the silent stage, they are the realistic recovery route.
7. **Collect the binary and any stderr capture.** If the operator ran it with `FTI_LOG` set, the log is a per-file record of exactly what was encrypted.
8. **Do not assume no exfiltration, and do not assume exfiltration.** The payload has no network capability; any data theft used separate tooling. Determine this from egress telemetry, not from the note.

---

8. Assessment Notes & Gaps

Item	Status
Who builds `/etc/f-index`	**Unknown.** Not created by this binary. Either a separate staging tool or repurposed Synology indexer output. Recovering that tool would materially improve early detection.
The `_F` and `c2w` / `ctw` suffix variants	**Unresolved.** The note promises three suffixes; this build emits one. Self-exclusion entries from sibling builds follow a `<base>` / `<base>_F` / `c2w` | `ctw` pattern, implying companion binaries not present in this sample set.
Earlier file format (v1)	**Recovered from dead code.** `zw_write_footer_v1_dead` @ `0x83E0` and `zw_rename_append_suffix_v1_dead` @ `0xB740` are compiled in but unreferenced. They describe an earlier build that appended a bare 256-byte RSA blob (no magic, no length) and **appended** the suffix to the original filename instead of replacing it. Files from that build are invisible to the footer-magic rule.
Exfiltration capability	**Absent from this binary.** Claims in the note are unverified by code.
Initial access vector	**Out of scope** for a payload-only analysis.
Group attribution	**Not attempted.** Conclusions here derive from the binary alone; the LockBit exclusion list indicates borrowed builder configuration, **not** a LockBit affiliation.
Build date	Inferred only. The hardcoded clock fallback `1784990000` = 2026-07-25T14:33:20Z sits inside the accepted execution window and is the closest available proxy.

Tracking recommendations

- Pivot on the RSA SPKI hash `66e37055247446e1f23efe62cf011b8cf77aa8aa0a937469319a5d8ce9053004` to link samples that share a key. A shared key across victims would indicate a single operator rather than an affiliate programme.
- Pivot on the 16-byte tweak constant `DF B3 07 AC 09 27 21 54 B1 E1 3B 58 7F 52 13 DE`. If it is stable across builds it is the best cross-build anchor available; if it varies, it becomes a per-build fingerprint.
- Watch the DLS onion for victim postings to correlate with the hardcoded victim ID model.

---

9. Indicator Appendix

Payload

```
sha256  33d3afddaa5710cdcc4e93a7bae8be010c19747fb53d85fcd54ef32056a1eb0b
sha1    9d371d77700dca0950249b8e8a1da128dbbdc719
md5     d8957e860cb421e9c06e0fd05010cd4b
```

Host artefacts

```
/etc/slient-time/baseline.stamp
/etc/f-index/baseline.stamp
/etc/f-index/<YYYY>/<MM>/index.dat
/etc/f-index/_other/index.dat
/var/run/enc-nas.lock
How To Restore Your Files.txt
<16 uppercase hex>.fHiQJervS
```

Cryptographic

```
footer magic (LE on disk)   DF B3 07 AC 09 27 21 54
footer magic (u64)          0x54212709AC07B3DF
XTS filename tweak          DF B3 07 AC 09 27 21 54 B1 E1 3B 58 7F 52 13 DE
CTR-DRBG personalisation    IEvDBAMINBgkIAqhQ
RSA SPKI SHA-256            66e37055247446e1f23efe62cf011b8cf77aa8aa0a937469319a5d8ce9053004
```

Contact / infrastructure

```
onion      fyenuhkq3pfhnbpidj5jm2fl2lryxip4byhg6eozynrnlomu4szf2nyd.onion
email      zawooorecover@onionmail.org
session    05b0794c896a54271d3d39c3b395d86aee2d5c05f196c59f3781bac1b1ac1a1a12
victim ID  <VICTIM_ID>   (16 uppercase hex, unique per build - redacted)
```

Execution window guardrail

```
TS_MIN    1780272000   2026-06-01T00:00:00Z
TS_MAX    1798761600   2027-01-01T00:00:00Z
fallback  1784990000   2026-07-25T14:33:20Z
```