CTF WRITEUP · FORENSICS / DFIR

Quiet Backup

Target: quite_backup.zip · **Category:** Forensics / Windows Event Log / Timeline Analysis · **Status:** Solved

`Lun4R{WKSDEV04_20250908_9f2c8a41_9SVC-BACKUP}`

Quiet Backup — Forensic Writeup

Artifact: `quite_backup.zip` (12,113 bytes, 16 evidence files) **Category:** Forensics (DFIR)
Flag format: `Lun4R{A_B_C_D}`

1. Challenge Overview

Meridian Analytics LLC flagged a backup archive created outside its documented maintenance window, alongside an unusual overnight spike in outbound network traffic. The investigator is handed a forensic export package — mail gateway logs, workstation triage artifacts (Amcache, Prefetch, MFT/USN journal, PowerShell script-block logging, EDR telemetry), a server-side Security event log, a system time-service health log, a facilities badge log, a VPN log, an SCCM deployment log, backup job configuration + rotation-ingestion records, filesystem metadata for two suspiciously similar archives, and a hash inventory — and asked to reconstruct what actually happened, identify the initial-compromise host, the true (UTC) foothold timestamp, the real malicious loader, and resolve a Windows identity-spoofing trick used to fabricate a "legitimate-looking" backup for exfiltration.

2. Evidence Inventory

```
$ unzip -l quite_backup.zip
quite_backup/evidence/mail/EV-01_mail_gateway_log.txt
quite_backup/evidence/workstation/WKS-DEV04/EV-02_amcache_prefetch.txt
quite_backup/evidence/workstation/WKS-DEV04/EV-03_mft_usnjrnl.txt
quite_backup/evidence/network/EV-04_dns_log.txt
quite_backup/evidence/network/EV-05_proxy_log.txt
quite_backup/evidence/workstation/WKS-DEV04/EV-06_security_evtx.txt
quite_backup/evidence/server/SRV-FS01/EV-07_security_evtx.txt
quite_backup/evidence/server/SRV-FS01/EV-08_system_health_log.txt
quite_backup/evidence/facilities/EV-09_badge_log.txt
quite_backup/evidence/network/EV-10_vpn_log.txt
quite_backup/evidence/sccm/EV-11_deployment_log.txt
quite_backup/evidence/backup/EV-12_backup_job_config.txt
quite_backup/evidence/backup/EV-13_archive_metadata.txt
quite_backup/evidence/workstation/WKS-DEV04/EV-14_powershell_operational.txt
quite_backup/evidence/workstation/WKS-DEV04/EV-15_edr_telemetry.txt
quite_backup/evidence/hashes/EV-16_hash_inventory.txt
```

Every one of the 16 files is used — this package has no filler decoys, but several entries function as *competing/misleading* evidence that has to be explicitly reasoned past (a timestomped file timestamp, a validly-signed-but-irrelevant binary, a spoofed logon identity). Grouped by role:

FILE	CONTRIBUTES
EV-01 mail gateway	Initial phishing delivery, the allowlist rule that let it through
EV-02 Amcache/Prefetch	First-seen time (UTC) for the loader, local-time run history
EV-03 MFT/$UsnJrnl	Timestomped `$STANDARD_INFORMATION` vs. authoritative `$FILE_NAME` /journal timestamps
EV-04 DNS	Baseline/benign query traffic (mostly noise, confirms nothing malicious at DNS layer)
EV-05 proxy	The C2 domain, and the two outbound transfers that separate the real backup from the fake one
EV-06 Security (WKS-DEV04)	Scheduled-task persistence (4698), audit-log tampering (4657)
EV-07 Security (SRV-FS01)	The Logon Type 9 (NewCredentials) event and its apparent identity

FILE	CONTRIBUTES
EV-08 system health	Proves SRV-FS01's local clock is unsynchronized — governs how EV-07's timestamp must be read
EV-09 badge log	Physical-presence alibi for the apparent identity in EV-07
EV-10 VPN	Rules out legitimate remote access by that same identity
EV-11 SCCM	Establishes the "signed tool" red herring is legitimate, routine software
EV-12 backup job config	The legitimate nightly backup's schedule and expected output naming
EV-13 archive metadata	Two archives, same folder, same night — one legitimate, one fabricated and misattributed
EV-14 PowerShell operational	A logging gap that lines up exactly with the malicious activity window
EV-15 EDR telemetry	Independent process/network telemetry: hidden PowerShell, LSASS access
EV-16 hash inventory	SHA-256 for both candidate "loader" binaries and the fake archive

3. Timeline Reconstruction

Two clocks are explicitly untrustworthy in this evidence set, and the challenge is built around not defaulting to "the timestamp says so."

Clock issue #1 — timestomping (`$STANDARD_INFORMATION`). EV-03 shows the loader's NTFS attributes disagree with each other:

```
$MFT record 88214 — OneDriveUpdater.exe
  $STANDARD_INFORMATION:
    Created:  2024-11-03 09:14:00 UTC
  $FILE_NAME:
    Created:  2025-09-08 14:12:03 UTC

$UsnJrnl:
  2025-09-08 14:12:03 UTC  USN 88214  FileCreate|DataExtend  ...OneDriveUpdater.exe
  2025-09-08 14:12:04 UTC  USN 88215  Close                 ...OneDriveUpdater.exe
```

`$STANDARD_INFORMATION` is the attribute every "timestomp" tool rewrites (it's what Explorer/ `dir` shows), and it claims the file has existed since November 2024 — ten months

before the incident, designed to make it look like an old, trusted file. `$FILE_NAME` timestamps and the `$UsnJrnl` transaction log are far harder to forge (attackers routinely skip them, and here they agree with each other to the second) and both say **2025-09-08 14:12:03 UTC**. EV-02's Amcache entry for the same file independently corroborates: `First seen: 2025-09-08 14:12:03 UTC` — three separate artifacts, three different subsystems (MFT, USN journal, Amcache), one timestamp. `$STANDARD_INFORMATION` is rejected as attacker-forged.

Clock issue #2 — local time vs. UTC (Prefetch). EV-02's Prefetch entries are logged in local time and say so explicitly:

```
[Prefetch] ONEDRIVEUPDATER.EXE-7C81D203.pf
  Last run (local): 2025-09-08 08:12:20    (America/Denver, UTC-6)
```

`08:12:20` local + 6h = `14:12:20` UTC — within seconds of the $UsnJrnl/Amcache creation time above. This isn't the timestamp used for B, but it's a useful independent cross-check that the UTC conversion convention (Denver, UTC-6, no DST ambiguity issue since both dates are in September) is right.

Clock issue #3 — an unsynchronized server clock. EV-08 (SRV-FS01 system health log):

```
2025-08-30 03:11:00 UTC  Event ID 129 (W32Time — "the time service has not
  been able to synchronize since the last reboot")
2025-09-09 06:00:00 UTC  Scheduled NTP sync attempt: FAILED (still
  unsynchronized since 2025-08-30)
```

SRV-FS01's local clock has been drifting, unsynced, since a reboot on Aug 30 — nine days before the incident. Every "(local)" timestamp recorded by SRV-FS01 after that point (including the Logon Type 9 event in §7) is **only trustworthy as a relative ordering marker, not an absolute UTC time**, which is exactly why D is resolved by the *logon type field* rather than by timeline position.

Consolidated chronology (UTC unless marked otherwise; SRV-FS01-local times are flagged unreliable per EV-08):

TIME (UTC)	EVENT	SOURCE
09:15:44	All-staff email: "IT pushing a GPO update tonight via remote PowerShell"	EV-01
14:02:11	Phishing email delivered to t.nguyen, `Q3_Budget_Review.xlsm`, bypassed quarantine via a stale 2023 allowlist rule (SPF fail, but rule not reviewed since creation)	EV-01
~14:09:52	Excel opened locally (08:09:52 Denver)	EV-02
14:12:03	`OneDriveUpdater.exe` **written to disk on WKS-DEV04** — the foothold (see §5)	EV-02, EV-03
14:20:11	Scheduled task `OneDrive Standalone Update Task-...` created, logon-triggered persistence, runs the loader	EV-06 (4698)
14:24:55–14:27:10	Benign-looking recon script blocks logged	EV-14
14:29:41	`ScriptBlockLogging` registry key flipped `1 → 0` by the loader (PID 6104), under cover of the "GPO update tonight" pretext from EV-01	EV-06 (4657)
14:29:41–15:12:00	**Gap in PowerShell 4104 logging** — logging was off for the most sensitive activity	EV-14
14:47:19	Hidden PowerShell spawned by `OneDriveUpdater.exe` (`-nop -w hidden -enc ...`)	EV-15
14:47:22	Outbound connection to `20.42.118.7:443`	EV-15
14:48:01	**LSASS access flagged suspicious**, requesting PID 6104 (`OneDriveUpdater.exe`)	EV-15
15:46:12	SCCM legitimately deploys signed `PSExec.exe` to WKS-DEV04 (routine, unrelated collection push)	EV-11
02:31 (9/9)	Legitimate nightly `NightlyFinanceBackup` job completes, 411.2 MB, ingested normally	EV-12
02:14:52 (SRV-FS01-local, **unsynced clock**)	Logon Type 9 (NewCredentials) on SRV-FS01, apparent account `d.reyes`	EV-07
07:51 (approx., per filename)	Second, fabricated backup archive created in the same staging folder, owned by `svc-backup`, no `BackupAgent.exe` telemetry	EV-13
07:58:02	18.4 MB egress from SRV-FS01 to `20.42.118.7:443` — the C2 domain, **not** the legitimate backup vault	EV-05

4. A — Initial Compromise Host

The phishing email in EV-01 is addressed to `t.nguyen@meridiananalytics.com`. Every subsequent artifact of the intrusion is scoped to one machine and one user profile: the Prefetch/Amcache triage (EV-02), the MFT/$UsnJrnl excerpt (EV-03), the scheduled-task persistence and audit-tampering events (EV-06), the PowerShell script-block gap (EV-14), and the EDR process/LSASS telemetry (EV-15) are **all collected from, and all reference paths under,** `WKS-DEV04` – specifically `C:\Users\t.nguyen\AppData\Roaming\Microsoft\OneDrive\OneDriveUpdater.exe`.

SRV-FS01 only enters the story later (§7, §5-adjacent) as the *second-stage target* the attacker pivots to using credentials obtained from WKS-DEV04 – it's where the fake backup is staged, not where the intrusion began. WKS-DEV04 is the initial foothold because it's the only host with (a) the phishing delivery's actual recipient and (b) the first malicious file write, first persistence mechanism, and first defense-evasion action, all timestamped hours before anything happens on SRV-FS01.

Per the clue ("clean its scars away, then let it speak in capitals"): the hostname as recorded everywhere in the evidence is `WKS-DEV04`. Stripping the hyphen ("scars") gives `WKSDEV04`; it's already fully uppercase in every artifact, satisfying "capitals" without further transformation.

```
A = WKSDEV04
```

5. B – Foothold Date

The artifact: EV-03's `$UsnJrnl` entry, corroborated by EV-02's Amcache `First seen` field:

```
$UsnJrnl:  2025-09-08 14:12:03 UTC  USN 88214  FileCreate|DataExtend
           C:\Users\t.nguyen\AppData\Roaming\Microsoft\OneDrive\OneDriveUpdater.exe

Amcache:   First seen: 2025-09-08 14:12:03 UTC
```

As established in §3, this is the moment the malicious loader – the actual foothold – was written to disk, and it is already recorded in UTC by both the journal and Amcache independently (matching to the second). The competing value, `$STANDARD_INFORMATION`'s `2024-11-03`

`09:14:00 UTC`, is the "hand that was moved" — a forged, timestomped date meant to backdate the file's apparent age — and is rejected for the reasons in §3. Prefetch's local-time entries are "a different sky" (a true, correctly-labeled clock, just not UTC); converting `08:12:20 America/Denver` gives `14:12:20 UTC`, within 17 seconds of the journal record and confirming the timezone-offset arithmetic rather than contradicting the chosen value.

```
B = 20250908
```

6. C — Real Loader

Two executables are in play on WKS-DEV04, and only one is what it looks like.

`PSEXEC.EXE` (`C:\Windows\CCM\Cache\PSEXEC.EXE`) — this is the binary that "wants to be found": it's deployed by SCCM through a documented, routine collection push (EV-11, `Sysinternals-Toolkit-v2.91`, targeted at `All-Eng-Workstations`, status Success), it carries a **valid Microsoft/Sysinternals digital signature**, and its hash in EV-16 is annotated `matches published Sysinternals PsExec v2.4 release hash`. Its Prefetch entry (run count: 1, EV-02) is consistent with a normal admin/SCCM tool invocation. It's legitimate — a real IT tool present on a real IT-managed machine — and its very legitimacy is the trap: it's the binary an investigator finds first when grepping for "suspicious-sounding" filenames.

`OneDriveUpdater.exe` (`C:\Users\t.nguyen\AppData\Roaming\Microsoft\OneDrive\OneDriveUpdater.exe`) is "the one that carried the burden": it's **unsigned** (EV-02: `Digital signature: Unsigned`), it lives in a *user* AppData path rather than the real OneDrive install location, its NTFS timestamps were tampered with (§3/§5), it's the parent process of a hidden, encoded PowerShell launch and the process that touched LSASS (EV-15), and it's the process that disabled PowerShell script-block logging on the host it runs on (EV-06, PID 6104). This is the actual malicious loader.

```
$ grep -A4 "OneDriveUpdater.exe" EV-16_hash_inventory.txt
File: OneDriveUpdater.exe
Path: C:\Users\t.nguyen\AppData\Roaming\Microsoft\OneDrive\OneDriveUpdater.exe
```

```
Host: WKS-DEV04
SHA-256: 9f2c8a41b7e3d9046c1a5f8b2e7d4c39a0f61b8d3e5c7a92146fbd0837c5e91a
```

"Take only the beginning of its shadow" — first 8 hex characters of the SHA-256:

```
C = 9f2c8a41
```

7. D — Credential Injection / Identity Confusion

This is the section the whole case hinges on, and it's built entirely around one field in one event record.

Identity #1 — what the record appears to say. EV-07 (SRV-FS01 `Security.evtx`):

```
Event ID 4624 (An account was successfully logged on)
  Time: 2025-09-09 02:14:52 (local, America/Denver — as recorded by the
        local system clock; see EV-08 for time-service status)
  Logon Type: 9 (NewCredentials)
  Account Name: d.reyes
  Account Domain: MERIDIAN
  Source Network Address: 10.10.4.41 (WKS-DEV04)
  Logon Process: seclogo
```

At face value, this says `d.reyes` logged on to SRV-FS01 from WKS-DEV04 overnight. Two independent artifacts rule that out as a real, physical/interactive action by that person:

- **EV-09 (badge log):** `d.reyes` badged *out* of the building at `17:42 local` on 9/8 and has **no further badge events until 07:58 local on 9/9** — they were not on-premises when the 02:14 logon was recorded.
- **EV-10 (VPN log):** *"no VPN sessions recorded for account d.reyes between 2025-09-08 17:42 local and 2025-09-09 07:55 local."* No remote-access path either.

`d.reyes` did not perform this logon. The name in the event record is not a fabricated field, though — it's an artifact of exactly *how* Windows Logon Type 9 works, which is the actual clue ("the distinction is hidden in the way Windows describes the door that was used").

Logon Type 9 (`NewCredentials`) is what Windows records for `runas /netonly`-style operations and equivalent credential-injection APIs (e.g., `LogonUser` with `LOGON32_LOGON_NEW_CREDENTIALS`, or pass-the-hash tooling that uses the same primitive): the *calling* session's identity is preserved for the 4624 record, while a **separate, injected set of credentials** is what actually gets used against the network resource. It is Windows' own built-in mechanism for using one identity locally while presenting a *different* identity to the network — precisely the "two identities on the same trail" the clue describes, and precisely why this logon type is a standard credential-injection/pass-the-hash indicator rather than a normal interactive or even RDP logon (Types 2/10).

Identity #2 — who actually walked it. EV-13 (filesystem metadata comparison, `SRV-FS01, C:\BackupStaging\Q3\`):

```
Backup_Archive_20250909_0751.zip
  Size: 18.4 MB
  Internal paths: 62 files, all under \Finance\Contracts\ and
    \Finance\WireInstructions\ only
  Created by process: (no BackupAgent.exe telemetry correlates to this file)
  Owner (NTFS): MERIDIAN\svc-backup
```

Compare this against the legitimate nightly job:

```
Backup_Archive_20250909_0230.zip     ← real, EV-12 ingestion log confirms
  Size: 411.2 MB
  Internal paths: 14,209 files, full \Finance\ tree
  Created by process: BackupAgent.exe (svc-backup, scheduled task context)
```

Both files are owned by `svc-backup` — but only one of them was actually produced by the real `BackupAgent.exe` scheduled-task process. The `07:51` archive is a hand-picked, high-value subset (contracts + wire instructions — exactly what an attacker wants, not what a routine full-tree backup produces), has no corresponding `BackupAgent.exe` telemetry, and doesn't appear anywhere in the backup system's own rotation-ingestion log (EV-12: *"no ingestion record exists for any archive timestamped 07:xx on 2025-09-09"*). It was created by *something* holding `svc-backup`'s credentials, outside the legitimate backup pipeline — which is exactly what the injected credentials from a Logon Type 9 / NewCredentials operation would provide: local execution

under one context, with `svc-backup`'s identity presented for the actual filesystem operation and the resulting file ownership.

EV-05 (proxy log) closes the loop and supplies the "anomalous egress" the challenge brief opens with:

```
2025-09-09 07:58:02 UTC  src=SRV-FS01  dst_ip=20.42.118.7:443
  SNI/Host: cdn-assets-sync-api.net
  bytes_out=18.4MB
```

`18.4MB` out, to `20.42.118.7` — the **same C2 IP** WKS-DEV04 beaconed to at 14:47:22 UTC the day before (EV-15) — and matching the fake archive's size to the tenth of a megabyte. This is the exfiltration of the fabricated backup, not the 412 MB legitimate transfer to `backupvault.meridianbackups.net` recorded three hours earlier in the same log.

Putting D together: the numeric logon-type code that proves credential injection rather than a human at a keyboard is `9`; the account whose credentials/memory were actually leveraged to produce the fake backup — evidenced by NTFS ownership of the fabricated archive and the absence of any legitimate `BackupAgent.exe` provenance — is `SVC-BACKUP`. Concatenated exactly as the clue frames it (mechanism, then identity):

```
D = 9SVC-BACKUP
```

8. Flag Construction

```
A = WKSDEV04
B = 20250908
C = 9f2c8a41
D = 9SVC-BACKUP
```

```
Lun4R{WKSDEV04_20250908_9f2c8a41_9SVC-BACKUP}
```

Three underscores separate exactly four components, matching the stated flag format `Lun4R{A_B_C_D}`.

9. Attack Narrative

A phishing email (EV-01) reaches `t.nguyen` on `WKS-DEV04` by exploiting a three-year-old, never-reviewed mail-flow allowlist rule that overrides an SPF failure. The attachment drops `OneDriveUpdater.exe` into the user's own AppData folder at **2025-09-08 14:12:03 UTC** (EV-03/EV-02) — the true foothold, its file-system timestamps deliberately backdated to November 2024 to blend in. Within eight minutes it establishes logon-triggered persistence via a scheduled task (EV-06, 4698) and, riding the cover of a company-wide "IT is pushing a GPO update tonight" email sent that same morning (EV-01), disables PowerShell script-block logging on the host (EV-06, 4657) before launching a hidden, base64-encoded PowerShell payload (EV-15) that reaches out to a look-alike CDN domain (`cdn-assets-sync-api.net`, EV-04/EV-05) and accesses LSASS (EV-15) — harvesting credentials cached on the box, ultimately including material tied to the `svc-backup` service account. A gap in PowerShell operational logging (EV-14) covers the most sensitive minute of this window, exactly as intended by disabling script-block logging.

Roughly twelve hours later, those harvested `svc-backup` credentials are used via a Logon Type 9 (NewCredentials) operation against the file server `SRV-FS01` (EV-07) — injected credentials presented to the network while the local session identity recorded in the event log shows an unrelated, off-premises employee (`d.reyes`, alibied by badge and VPN logs, EV-09/EV-10). Using those injected `svc-backup` credentials, the attacker creates a second, hand-curated archive containing only `\Finance\Contracts\` and `\Finance\WireInstructions\` — dropped into the same staging folder the legitimate nightly backup uses, and owned by the same service account, but never touched by the real `BackupAgent.exe` process and never recorded in the backup system's own ingestion log (EV-13/EV-12). Minutes later, that 18.4 MB archive leaves the network for the *same* external host WKS-DEV04 beaconed to the day before — not the legitimate backup vault (EV-05) — completing the exfiltration that triggered the original alert.

Four independent evidence streams — mail delivery, workstation host telemetry, server authentication logs, and network egress — corroborate one continuous story from initial phish to

data theft.

10. Lessons / Detection Opportunities

- **Legacy mail-flow allowlist rules rot.** A rule added in 2023 and never reviewed is exactly the kind of exception attackers go looking for; allowlist entries should expire or require periodic re-justification.
- **$STANDARD_INFORMATION alone is not a trustworthy creation time.** Any triage process that reports file age from Explorer/`dir`-style metadata without cross-checking `$FILE_NAME` / `$UsnJrnl` (or Amcache "first seen") is trivially fooled by timestomping.
- **A valid signature is not a clearance to ignore a binary — nor is an unusual path a reason to ignore a legitimately-deployed one.** Correlate execution behavior (parent/child process chains, LSASS access, network beaconing) rather than trusting signature status or deployment provenance in isolation.
- **Unexpected executable creation in user-writable profile paths** (`AppData\Roaming\...`) impersonating known-vendor filenames is a durable, high-value detection: alert on any `*Updater.exe` / `*Sync.exe`-style binary appearing outside its real vendor install path.
- **Logon Type 9 (NewCredentials) is inherently suspicious outside of known `runas /netonly` administrative tooling** — it is the textbook signature of credential injection and pass-the-hash-style tradecraft. Baseline and alert on it; the apparent account name in the 4624 record is not sufficient attribution on its own.
- **LSASS access from a non-security-tooling process** should page someone in real time, not wait for post-incident triage.
- **Backup archives created outside the documented job schedule, or that don't correlate to the legitimate backup agent's own telemetry, are a strong exfiltration-staging indicator** — even when file ownership matches the expected service account.
- **Abnormal outbound transfer size/destination pairs** (a backup-shaped file going somewhere that isn't the backup vault) are cheap to detect with basic egress allowlisting by destination domain.

```
Lun4R{WKSDEV04_20250908_9f2c8a41_9SVC-BACKUP}
```

Independent writeup · rootriet LUN4R CTF 2026 · reconstructed from live session command history