AUDIT-11 — Controls + Test-Ergebnisse

Stand: 2026-05-18 (Initial-Lauf) **Methodik:** Code-Inspektion + simulierte Angreifer-Profile A1/A2/A3 gegen V1–V10. Bewertung pro Vektor: ✓ isoliert, ◐ teilweise isoliert, ✗ nicht isoliert.

Architektonische Kontrollen (Soll-Stand)

Kontrolle	Implementierung	Beleg
Verzeichnis-Trennung pro Koordinator	`workspaces/berater_<id>/mandanten/<slug>/`	`core/skills/cdp/mandanten_db.py` Z. 35–47
HMAC-Session-Token mit Rollen-Claim	`auth_service.py` `_create_token`	`ce-doc-portal/backend/auth_service.py` Z. 79–87
Rollen-Hierarchie operator > koordinator > hersteller	`_ROLLE_RANG`-Dict + `_rolle_mindestens`	`api.py` Z. 98 + 134
CSRF-Schutz auf POST	`_csrf_pruefen`	`api.py` Z. 105
Mandanten-DB mit Koordinator-Anker	`lade_mandant(berater_id, mandant_id)` baut Pfad konditional	`mandanten_db.py`

Test-Ergebnisse pro Vektor

V1 — Vorgangs-Endpoint URL-Manipulation ✗ NICHT isoliert

Test: Koordinator A authentifiziert, ruft `GET /api/vorgang/<case_id_von_B>` auf. **Erwartet:** HTTP 403 oder 404. **Beobachtet:** HTTP 200 mit vollem `akt_state` von B.

Code-Beleg (`api.py` Zeile 423–446):

```
@app.get("/api/vorgang/{case_id}")
async def vorgang_detail(case_id: str, request: Request) -> dict:
    state = _lade_state(case_id)              # ← KEIN Tenant-Check
    if state is None:
        raise HTTPException(404, ...)
    # ...
    return {"case_id": case_id, "state": state, ...}
```

`_lade_state(case_id)` baut den Pfad ohne Koordinator-Bezug auf (`WORKSPACES / case_id / "analyst_outputs" / "akt_state.json"`). Es gibt keinen Check, ob `state["meta"]["berater_id"]` mit dem authentifizierten Benutzer übereinstimmt.

Schwere: HIGH — direkter Vertraulichkeitsverlust gegenüber fremdem Koordinator.

Befund: AUDIT-11-F-01 (siehe 04_findings).

V2 — Mandanten-Endpoint URL-Manipulation ✓ isoliert

Test: Koordinator A ruft `GET /api/mandant/<mandant_id_von_B>` auf. **Beobachtet:** HTTP 404 — `lade_mandant(berater_id_A, mandant_id_von_B)` sucht unter `workspaces/berater_A/mandanten/<mandant_id_von_B>/` , das nicht existiert (weil B's Mandant unter `workspaces/berater_B/...` liegt).

Beleg (`api.py` Z. 1068–1086):

```
async def mandant_detail(mandant_id: str, request: Request) -> dict:
    user = _benutzer(request)
    berater_id = user.get("name") if user.get("rolle") in (...) else None
    if not berater_id: raise HTTPException(403)
    m = lade_mandant(berater_id, mandant_id)   # ← Koordinator-Anker
    if not m.get("stammdaten"):
        raise HTTPException(404, ...)
```

Schwere: keine (✓).

V3 — Path-Traversal in case_id ◐ teilweise

Test: A1 ruft `GET /api/vorgang/../../../etc/passwd` oder Varianten. **Beobachtet:** FastAPI URL-Routing trennt Pfad-Segmente, `case_id` wird als Single-Segment-String aufgefangen — keine direkte Path-Traversal-Wirkung. Bei Sonderzeichen wie `..%2F` würde der String roh nach `Path("…")` gehen.

Risiko: mittel — kein direkter Lese-Pfad in /etc/, aber innerhalb von `workspaces/` könnte ein gerateter case_id wie `_compliance` oder `mca_sichter` unbeabsichtigt geladen werden (sind nicht im `_KANBAN_AUSBLENDEN`).

Befund: AUDIT-11-F-02 (MED).

V4 — Akte-Render-Endpoint ✗ NICHT isoliert

Gleiches Muster wie V1: `vorgang_akte_html(case_id)` lädt direkt aus `WORKSPACES / case_id / "output"` ohne Tenant-Check.

Befund: AUDIT-11-F-01 (gleicher Defekt-Klasse wie V1).

V5 — OPL-Antwort-POST ✗ NICHT isoliert

Z. 932+: `_rolle_mindestens(request, "koordinator")` prüft Rolle, aber nicht Tenant. A1 kann theoretisch eine OPL-Antwort auf B's Vorgang setzen.

Befund: AUDIT-11-F-01 (gleiche Klasse).

V6 — Mail-Inbox-Bucket ✓ isoliert (dateibasiert)

Mail-Buckets liegen unter `workspaces/projects/<case_id>/inbox/`. Lesen erfordert API-Zugang (siehe V1), aber direkter Filesystem-Zugriff durch Mail-Daemon ist auf den Daemon-User beschränkt. **Indirekt** über V1 ausnutzbar (HIGH-Befund deckt das ab).

V7 — LLM-Prompt-Kontext-Leak ✓ isoliert

LLM-Aufrufe bauen Prompts pro Case unabhängig auf. Keine Daten anderer Cases landen im Kontext. Verifiziert durch Code-Stichprobe in `core/skills/cdp/` und `core/skills/mca_*`.

V8 — Backup-Restore ✓ isoliert

Backup-Restore erfordert R-OPS-Zugriff (Bastion + GF-Freigabe laut PROC-K3). Kein Koordinator-Self-Service-Restore.

V9 — Anonymisierte Logs ◐ teilweise

Aktuelle Logs enthalten case_id, das koordinator-spezifisch ist. PII innerhalb der State-Dumps in Logs ist möglich (AUD-1, offen, siehe AUDIT-08-F-01 + AUDIT-09-F-04). Koordinator-Cross-Read auf Logs ist organisatorisch nicht gegeben (R-COMP-Only-Zugriff), aber technisch nicht erzwungen.

Befund: AUDIT-11-F-03 (MED).

V10 — Session-Token Koordinator-Claim ◐ teilweise

Token-Format: `username:rolle:ts:hmac`. Es gibt **keinen separaten berater_id-Claim** — der Tenant wird aus `username` abgeleitet (`api.py` Z. 1055: `berater_id = user.get("name")`). Das funktioniert, solange `username` == `berater_id` invariant ist. Wird aber bei Multi-User- pro-Koordinator-Konstellationen (z. B. Koordinator-Sekretariat) nicht skalieren.

Befund: AUDIT-11-F-04 (LOW) — Konzeptionelle Lücke für Wachstum.

Zusammenfassung der Ergebnisse

Vektor	Stand	Klassifikation
V1 Vorgangs-Endpoint	✗	**HIGH**
V2 Mandanten-Endpoint	✓	OK

Vektor	Stand	Klassifikation
V3 Path-Traversal	◐	MED
V4 Akte-Render	✗	HIGH (gleiche Klasse wie V1)
V5 OPL-POST	✗	HIGH (gleiche Klasse wie V1)
V6 Mail-Inbox	✓ (indirekt V1-betroffen)	OK
V7 LLM-Prompt	✓	OK
V8 Backup-Restore	✓	OK
V9 Logs	◐	MED
V10 Token-Claim	◐	LOW

Reife-Übersicht: 5 voll isoliert, 3 teilweise, 3 nicht isoliert (V1/V4/V5 sind dieselbe Defekt-Klasse). **1 HIGH-Befund (AUDIT-11-F-01)** ist konsolidiert.

Mitigations-Skizze für AUDIT-11-F-01

Eine einzige Helper-Funktion `_assert_tenant(user, state)` in `api.py`, die in `_lade_state` oder am Endpoint-Eingang aufgerufen wird:

```
def _assert_tenant(user: dict, state: dict, case_id: str) -> None:
    """Prüft, dass der eingeloggte User auf diesen Vorgang Zugriff hat.

    Operator: ja, jeder Vorgang.
    Koordinator: nur Vorgänge, deren state.meta.berater_id == user.name.
    Hersteller: nur Vorgänge, in denen sie als kontakt_email eingetragen sind.
    """
    rolle = user.get("rolle")
    if rolle == "operator":
        return
    if rolle == "koordinator":
        owner = state.get("meta", {}).get("berater_id")
        if owner != user.get("name"):
            raise HTTPException(404, f"Vorgang '{case_id}' nicht gefunden")
    # ... weitere Rollen
```

Aufruf nach `_lade_state` in allen sechs `/api/vorgang/{case_id}/*`-Endpoints. Aufwand: ~2 Stunden + Test.