📋 TASK 3: CREATE PROPER AUDIT LOG

Priority: CRITICAL
Estimated Time: 1 day
Status: NOT STARTED
Dependencies: None (can do in parallel with other tasks)

🎯 WHAT THIS TASK IS ABOUT

The Requirement:

> "It will implement an audit log of all events that occur, indicating in a structured way: (a) Date and time of the event, (b) Who and from where the event occurs: IP of the machine, (c) What action is performed, (d) Parameters or description of the event."

In Simple Words:

Teacher wants to open a text file and see EVERYTHING that happened in the system:

- Who authenticated
- Who charged
- What failed
- Who did admin commands
- All with timestamps and IP addresses

Currently: You have Kafka events (good!) but no visible audit file (bad!)

❌ WHAT'S MISSING

Current state:

- Events go to Kafka topics
- Kafka stores them internally
- Teacher can't easily see them
- No single audit file to inspect

Problems:

- Teacher can't do: `cat data/audit_log.txt`
- No IP address tracking
- No structured format
- Will lose points in correction

✅ WHAT YOU NEED

Create: `data/audit_log.txt`

Format example:

```
2025-12-16T10:30:45.123 | 192.168.1.100 | CP-001 | AUTHENTICATION_SUCCESS | credentials_verified=true
2025-12-16T10:31:12.456 | 192.168.1.101 | DRIVER-001 | CHARGE_REQUESTED | cp=CP-001, kwh=10
2025-12-16T10:31:15.789 | 192.168.1.100 | CP-001 | CHARGE_AUTHORIZED | driver=DRIVER-001
2025-12-16T10:31:30.012 | 192.168.1.100 | CP-001 | SUPPLY_UPDATE | kwh=2.5, amount=0.75€
2025-12-16T10:32:00.345 | 192.168.1.100 | CP-001 | CHARGE_COMPLETED | total_kwh=10, total_amount=3.00€
2025-12-16T10:32:05.678 | 192.168.1.50 | ADMIN | COMMAND_EXECUTED | action=stop_cp, target=CP-002
2025-12-16T10:32:10.901 | 192.168.1.100 | CP-001 | FAULT_DETECTED | component=engine, status=health_ko
2025-12-16T10:32:15.234 | 192.168.1.100 | CP-001 | RECOVERY | fault_resolved=true
```

Each line has:

- Timestamp with milliseconds
- IP address (where the event came from)
- Entity (who did it: CP-001, DRIVER-001, ADMIN, etc.)
- Event type (what happened)
- Parameters (details)

🔧 WHAT YOU NEED TO DO

Step 1: Create Audit Logger Module

Create file: `shared/audit_logger.py`

What to put in it:

Function 1: Get IP Address

- Extract IP from socket connection
- Handle IPv4 and IPv6
- Return "UNKNOWN" if can't get IP

Function 2: Log Event

- Takes: entity, event_type, ip_address, parameters
- Formats as single line
- Appends to `data/audit_log.txt`
- Thread-safe (use lock)
- Handles file doesn't exist

Function 3: Read Recent Logs

- Reads last N lines
- Returns as list
- Used for debugging

Step 2: Modify Central to Log Everything

File: `central/ev_central.py`

Import audit logger at top:

```python
from shared.audit_logger import AuditLogger
```

Create instance:

```python
self.audit = AuditLogger("data/audit_log.txt")
```

Add logging to these methods:

In `_handle_authenticate()`:

python

```python
# After checking credentials
if valid:
    self.audit.log(
        entity=cp_id,
        event_type="AUTHENTICATION_SUCCESS",
        ip_address=client_ip,
        parameters=f"username={username}"
    )
else:
    self.audit.log(
        entity=cp_id,
        event_type="AUTHENTICATION_FAILED",
        ip_address=client_ip,
        parameters=f"username={username}, reason=invalid_credentials"
    )
```

In `_handle_charge_request()`:

python

```
self.audit.log(
    entity=driver_id,
    event_type="CHARGE_REQUESTED",
    ip_address=client_ip,
    parameters=f"cp={cp_id}, kwh={kwh_needed}"
)

# Later, if authorized:
self.audit.log(
    entity=cp_id,
    event_type="CHARGE_AUTHORIZED",
    ip_address=cp_ip,
    parameters=f"driver={driver_id}, kwh={kwh_needed}"
)

# If denied:
self.audit.log(
    entity=cp_id,
    event_type="CHARGE_DENIED",
    ip_address=cp_ip,
    parameters=f"driver={driver_id}, reason={reason}"
)
```

In `_handle_supply_end()`:

python

```
self.audit.log(
    entity=cp_id,
    event_type="CHARGE_COMPLETED",
    ip_address=cp_ip,
    parameters=f"driver={driver_id}, total_kwh={total_kwh}, total_amount={total_amount}€"
)
```

In `_handle_fault()`:

python

```
self.audit.log(
    entity=cp_id,
    event_type="FAULT_DETECTED",
    ip_address=cp_ip,
    parameters=f"component=engine, reported_by=monitor"
)
```

In `_handle_recovery()`:

python

```
self.audit.log(
    entity=cp_id,
    event_type="RECOVERY",
    ip_address=cp_ip,
    parameters=f"fault_resolved=true"
)
```

In admin commands (stop/resume):

python

```
self.audit.log(
    entity="ADMIN",
    event_type="COMMAND_EXECUTED",
    ip_address="localhost",
    parameters=f"action=stop_cp, target={cp_id}"
)
```

Step 3: Track IP Addresses Properly

Where to get IP from:

For socket connections:

python

```
def _handle_client(self, client_socket, client_id):
    # client_id is like "192.168.1.100:12345"
    client_ip = client_id.split(':')[0]
    # Now use client_ip in audit logs
```

Store IP with entity:

```python
# When registering
self.entity_to_ip[entity_id] = client_ip

# When logging
ip = self.entity_to_ip.get(entity_id, "UNKNOWN")
```

Step 4: Add Rotation (Optional but Nice)

If file gets too big:

- When it reaches 10,000 lines
- Rename to `audit_log_2025-12-16.txt`
- Start new `audit_log.txt`
- Keeps logs manageable

🧪 HOW TO TEST

Test 1: Log File Created

```
1. Delete data/audit_log.txt if exists
2. Start Central
3. Check: data/audit_log.txt should be created
4. Should be empty initially
```

Expected: File exists

Test 2: Authentication Logged

```
1. Create CP using CP Manager
2. Check audit_log.txt
3. Should see line like:
   "2025-12-16T... | 192.168.1.100 | CP-001 | AUTHENTICATION_SUCCESS | ..."
```

Expected: Authentication events logged

Test 3: Charging Logged

```
1. Use Driver to request charge
2. Complete the charge
3. Check audit_log.txt
4. Should see lines for:
   - CHARGE_REQUESTED
   - CHARGE_AUTHORIZED
   - Multiple SUPPLY_UPDATE (optional)
   - CHARGE_COMPLETED
```

Expected: All charging events logged

Test 4: Faults Logged

```
1. Kill CP_Engine (simulate fault)
2. Monitor detects it
3. Check audit_log.txt
4. Should see: FAULT_DETECTED
5. Restart CP_Engine
6. Should see: RECOVERY
```

Expected: Fault and recovery logged

Test 5: Admin Commands Logged

```
1. Attach to Central admin console
2. Execute "stop CP-001"
3. Check audit_log.txt
4. Should see: COMMAND_EXECUTED with action=stop_cp
```

Expected: Admin actions logged

Test 6: IP Addresses Correct

```
1. Look at audit_log.txt
2. Every line should have valid IP
3. Same entity should have same IP (usually)
4. No lines with "UNKNOWN" IP (unless actually unknown)
```

Expected: IP addresses make sense

📋 STEP-BY-STEP IMPLEMENTATION

Hour 1-2: Create Audit Logger

- Create `shared/audit_logger.py`
- Write `AuditLogger` class
- `__init__()` method
- `log()` method
- `get_recent()` method
- Test independently with simple script

Hour 3-4: Integrate with Central

- Import in `ev_central.py`
- Create instance
- Add logging to authentication
- Add logging to charging
- Test with one CP

Hour 5-6: Complete All Events

- Add logging for faults
- Add logging for admin commands
- Add logging for weather alerts (if time)
- Test all scenarios

Hour 7-8: Polish & Test

- Add IP address tracking
- Test log format is clean
- Test file doesn't break with special characters
- Create dummy data for demonstration
- Test teacher can open and read file easily

📋 CHECKLIST

Before marking complete:

- [ ] Created shared/audit_logger.py
- [ ] Tested logger independently
- [ ] File created in data/audit_log.txt
- [ ] Authentication events logged
- [ ] Charge requests logged
- [ ] Charge completions logged
- [ ] Faults logged
- [ ] Recoveries logged
- [ ] Admin commands logged
- [ ] IP addresses tracked
- [ ] Timestamps with milliseconds
- [ ] Format is clean and readable
- [ ] Thread-safe (won't corrupt with concurrent writes)
- [ ] File readable by teacher
- [ ] Tested with multiple CPs
- [ ] Tested with multiple drivers
- [ ] At least 50+ log entries exist for demo

💡 WHAT TO TELL TEACHER

Teacher asks: "Show me the audit log"

Your action:

1. Open terminal
2. Type: `cat data/audit_log.txt` or `tail -50 data/audit_log.txt`
3. Teacher sees nicely formatted log

Your explanation: "The audit log captures all security-relevant events in the system with structured data. Each entry includes a precise timestamp, the originating IP address, the entity performing the action, the event type, and relevant parameters. This follows SIEM principles and allows for security analysis, compliance verification, and incident investigation. The log is append-only and thread-safe, ensuring data integrity even under concurrent operations. It captures authentication attempts, charging sessions, fault conditions, and administrative actions - providing complete traceability of system operations."

⚠ THINGS TO REMEMBER

Don't log:

- Passwords (even hashed)
- Encryption keys
- Too much data (keep parameters short)

Do log:

- Every authentication attempt (success AND failure)
- Every charge start and end
- Every fault and recovery
- Every admin command
- Every denied/rejected action

Format tips:

- Use pipe `|` as separator (easy to parse)
- Use ISO timestamps (sortable)
- Keep lines under 200 characters
- Use consistent event type names

✅ SUCCESS CRITERIA

This task is complete when:

1. ✅ File `data/audit_log.txt` exists
2. ✅ Every important event is logged
3. ✅ IP addresses are tracked
4. ✅ Format is clean and readable
5. ✅ Teacher can open file and understand it
6. ✅ At least 50 entries exist after testing
7. ✅ No sensitive data logged (passwords, keys)

Status: Not started
Next Task: TASK 4 (HTTPS for Registry)