### **Deep Dive: Week 4 – Building the Digital Twin of a Workplace**

Hello everyone. This week, our goal was to lay the entire foundation for the Zero Trust simulator. Before we could test any security measures, we first needed to build a realistic, living, breathing digital world that mimics the day-to-day operations of a modern company. We called this our "baseline simulation."

#### **What We Did This Week: Simulating a Normal Workday**

Our main focus was to answer the question: "What does a normal day at a company look like from a data perspective?" We achieved this by creating a script that generates a continuous stream of event logs, simulating employees performing their everyday tasks.

The core of this work was a single, powerful script:

1. **src/sim/run\_sim.py (The Event Generator):** This is the engine of our simulation. We programmed it to understand a set of key concepts:  
   * **Users:** We defined a cast of simulated employees, like "alice" and "bob".
   * **Devices:** We gave them devices to work on, such as "laptop-1" and "phone-2".
   * **Actions:** We defined a set of common actions they would perform throughout the day: login, accessing a file, writing to a file, and executing a program.
2. This script runs and randomly combines these elements to create a realistic sequence of events. For example, it might generate a log showing "Alice logged in from her laptop," followed by "Alice accessed a file on the marketing server."

#### **How We Built It: The Key Code Files**

To make this work in a structured and professional way, we created a few key files:

* **src/sim/run\_sim.py:** As mentioned, this was the main workhorse. We designed it to be highly configurable, so we can easily tell it, "Generate 10,000 events for me," and it will do so.
* **configs/sim\_config.json:** We made a critical design choice early on: no hardcoding. Instead of putting values like the number of users or devices directly into the code, we created a simple JSON configuration file. This allows us to easily change the parameters of our simulation without ever touching the code itself, making our experiments much more flexible.
* **src/utils/config.py:** To support this, we wrote a small utility module to handle loading these configuration files. It also includes a crucial function for reproducibility. By setting a "random seed," we can ensure that we can re-run the exact same "random" simulation multiple times, which is essential for verifying our results scientifically.
* **tests/test\_run\_sim.py:** From day one, we committed to quality. We wrote a suite of tests for our event generator. These tests automatically verify that the script produces the correct number of events, that the events follow the correct data format, and that our reproducibility feature works as expected.

#### **The Outcome: Our "Baseline" Data**

The tangible result from this week is a log file, typically named data/events.jsonl. This file is a rich timeline of everything that happened in our simulated company. It represents our baseline—a world with no advanced security. This is our control group. Every security measure we add from now on will be measured against this baseline to see how much of a difference it makes.

### **What We're Implementing Next Week (Week 5)**

Now that we have successfully simulated a normal, unprotected environment, our next logical step is to start securing it.

Next week, we will implement the first and most important layer of Zero Trust: Identity and Authentication.

Our plan is to:

1. **Build the Authentication Module (src/controls/auth.py):** We will write the code that handles user authentication. This will include functions to verify passwords and, more importantly, to simulate Multi-Factor Authentication (MFA).
2. **Integrate It into the Simulation:** We will update our event generator (run\_sim.py) to use this new authentication module. This means that simulated login events will now go through a security check.
3. **Measure the Impact:** For the first time, we will see a "decision" being made in our logs. Logins with correct credentials and MFA will be marked as "allow," while failed attempts will be marked as "deny."

By the end of next week, our simulation will no longer be just a passive observer; it will be an active security system capable of making its first critical decisions.