Contents

[10.2] Cascade Failure Triggers

1. Operational Definition: A state where a failure in one system or process initiates a sequence of failures in interdependent systems, leading to a large-scale security breakdown. This is measured by the speed and scale of incident propagation.

2. Main Metric & Algorithm:

- **Metric:** Cascade Propagation Rate (CPR). Formula: `CPR = (Number_of_Affected_Systems / Total_Interdependent_Systems) / Time_to_Containment (hours)`.
- **Pseudocode:**

  python

  ```
  def calculate_cpr(incident_id, systems_list):
      # Get the initial system from the first alert in the incident
      initial_alert = get_initial_alert(incident_id)
      initial_system = initial_alert.affected_system

      # Get all unique systems affected in the incident timeline
      timeline_alerts = get_incident_alerts(incident_id)
      affected_systems = set(alert.affected_system for alert in timeline_alerts)

      # Get time between first alert and incident closure
      time_to_contain = timeline_alerts[-1].time - initial_alert.time

      # Calculate the ratio of affected systems to total interconnected systems
      total_interconnected = get_interconnected_systems(initial_system, systems_list)
      propagation_ratio = len(affected_systems) / len(total_interconnected)

      # Avoid division by zero
      if time_to_contain.total_hours() == 0:
          time_to_contain_hours = 0.1
      else:
          time_to_contain_hours = time_to_contain.total_hours()

      cpr = propagation_ratio / time_to_contain_hours
      return cpr
  ```

- **Alert Threshold:** `CPR > 0.05` (i.e., more than 5% of interconnected systems are affected per hour of containment time).

3. Digital Data Sources (Algorithm Input):

- **CMDB API:** (e.g., ServiceNow) to map system dependencies (`depends_on`, `connected_to` relationships).

- **SIEM/SOAR:** (e.g., Splunk ES, Splunk Phantom) to get the incident timeline (`incident_id`, `alert_time`, `affected_system` fields from notable events).

4. Human-to-Human Audit Protocol: Conduct a tabletop exercise or a retrospective on a past major incident. Ask the team: "Walk us through the sequence of failures. Did one system's compromise directly lead to another's? How quickly did the propagation occur, and what finally stopped it?" The goal is to map the failure chain and identify single points of failure.

5. Recommended Mitigation Actions:

- **Technical/Digital Mitigation:** Implement automated containment playbooks in the SOAR platform to isolate compromised systems based on pre-defined dependency graphs.
- **Human/Organizational Mitigation:** Train incident responders on cascade recognition and containment prioritization, focusing on critical "choke points" in the system architecture.
- **Process Mitigation:** Mandate architectural reviews for new systems to ensure they are designed with failure isolation in mind (e.g., bulkheads, circuit breakers).