Contents

[5.2] Decision Fatigue Errors

1. Operational Definition: A state of mental exhaustion and impaired judgment resulting from a high volume of sequential security decisions, leading to a degradation in the quality of choices over time, such as opting for the default or easiest option regardless of risk.

2. Main Metric & Algorithm:

- **Metric:** Decision Quality Decay Rate (DQDR). Formula: `Slope of the linear regression line ( ) fitted to the decision quality scores (e.g., % correct actions) of an analyst's last N closed tickets, ordered by time. A negative slope indicates decay..`
- **Pseudocode:**

  python

```
def calculate_dqdr(tickets, analyst_id, window_size=20):
    # Fetch the analyst's last 'window_size' resolved tickets, sorted by close time
    analyst_tickets = get_tickets(assigned_to=analyst_id, status='closed', sort='-closed_a

    if len(analyst_tickets) < 5:  # Insufficient data
        return None

    # For each ticket, calculate a decision quality score (e.g., 1.0 = perfect remediation
    quality_scores = []
    for ticket in analyst_tickets:
        score = assess_decision_quality(ticket.resolution_action, ticket.severity, ticket.
        quality_scores.append(score)

    # Reverse so list is in chronological order (oldest first)
    quality_scores.reverse()

    # Perform linear regression: y =  x +  , where x is the sequence number (1,2,3,...)
    x = np.arange(1, len(quality_scores) + 1)
    slope, intercept = np.polyfit(x, quality_scores, 1)

    return slope  # This is the  coefficient (DQDR)

# Helper function - logic must be defined based on organizational policy
def assess_decision_quality(action_taken, severity, criticality):
    if action_taken == "fully_remediated":
        return 1.0
    elif action_taken == "partially_remediated" and severity == "high":
        return 0.3
    # ... more rules
    else:
```

```
        return 0.0
```

- **Alert Threshold:** `DQDR < -0.02` (A consistent negative trend indicating quality degrades with each subsequent decision).

3. Digital Data Sources (Algorithm Input):

- **SOAR/SIEM:** Splunk ES or Elastic SIEM for alert history and status transitions. Fields: `assigned_analyst`, `timestamp`, `alert_severity`, `resolution_action`.
- **Ticketing System (Jira/ServiceNow):** REST API to query tickets. Fields: `assignee`, `created`, `resolved`, `resolution`, `custom_field_asset_criticality`.

4. Human-to-Human Audit Protocol: Conduct a brief, anonymous survey with analysts at the end of a shift: "On a scale of 1-5, how mentally drained do you feel by the decisions you made today?" and "Do you feel the quality of your security decisions changed from the beginning to the end of your shift?" Correlate responses with DQDR metrics.

5. Recommended Mitigation Actions:

- **Technical/Digital Mitigation:** Implement a SOAR playbook that automatically routes complex, high-severity alerts to the freshest available analyst (e.g., one just starting their shift or returning from a break).
- **Human/Organizational Mitigation:** Mandate structured micro-breaks (5 minutes every hour) for analysts during high-volume periods.
- **Process Mitigation:** Develop and enforce standardized decision trees or runbooks for common alert types to reduce the cognitive load of deciding *how* to respond.