Contents

[5.2] Errori di affaticamento decisionale

1. Definizione operativa: Uno stato di esaurimento mentale e giudizio compromesso risultante da un alto volume di decisioni di sicurezza sequenziali, portando a un degradazione della qualità delle scelte nel tempo, come optare per l'opzione predefinita o più facile indipendentemente dal rischio.

2. Metrica principale e algoritmo:

- **Metrica:** Tasso di decadimento della qualità decisionale (DQDR). Formula: `Pendenza della linea di regressione lineare ( ) adattata ai punteggi di qualità decisionale (ad es. % di azioni corrette) dei ticket chiusi più recenti di un analista, ordinati per tempo. Una pendenza negativa indica decadimento.`.
- **Pseudocodice:**

```
def calculate_dqdr(tickets, analyst_id, window_size=20):
    # Recuperare gli ultimi 'window_size' ticket risolti dell'analista, ordinati per tempo
    analyst_tickets = get_tickets(assigned_to=analyst_id, status='closed', sort='-closed_a

    if len(analyst_tickets) < 5:  # Dati insufficienti
        return None

    # Per ogni ticket, calcolare un punteggio di qualità decisionale (ad es. 1.0 = rimedio
    quality_scores = []
    for ticket in analyst_tickets:
        score = assess_decision_quality(ticket.resolution_action, ticket.severity, ticket.
        quality_scores.append(score)

    # Invertire in modo che l'elenco sia in ordine cronologico (meno recente prima)
    quality_scores.reverse()

    # Eseguire regressione lineare: y =  x +  , dove x è il numero di sequenza (1,2,3,...)
    x = np.arange(1, len(quality_scores) + 1)
    slope, intercept = np.polyfit(x, quality_scores, 1)

    return slope  # Questo è il coefficiente  (DQDR)

# Funzione helper - la logica deve essere definita in base alla politica organizzativa
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
    # ... altre regole
    else:
```

```
        return 0.0
```

- **Soglia di avviso:** `DQDR < -0.02` (Un trend negativo consistente che indica la qualità si degrada con ogni decisione successiva).

3. Fonti di dati digitali (Input dell'algoritmo):

- **SOAR/SIEM:** Splunk ES o Elastic SIEM per la cronologia degli avvisi e le transizioni di stato. Campi: `assigned_analyst`, `timestamp`, `alert_severity`, `resolution_action`.
- **Sistema Ticketing (Jira/ServiceNow):** API REST per interrogare i ticket. Campi: `assignee`, `created`, `resolved`, `resolution`, `custom_field_asset_criticality`.

4. Protocollo di audit uomo-uomo: Condurre un breve sondaggio anonimo con gli analisti alla fine di un turno: "Su una scala da 1 a 5, quanto sei mentalmente esausto dalle decisioni che hai preso oggi?" e "Senti che la qualità delle tue decisioni di sicurezza è cambiata dall'inizio alla fine del tuo turno?" Correlare le risposte con le metriche DQDR.

5. Azioni di mitigazione consigliate:

- **Mitigazione tecnica/digitale:** Implementare un playbook SOAR che instrada automaticamente gli avvisi complessi ad alta severità all'analista più fresco disponibile (ad es. uno che inizia il turno o che ritorna da una pausa).
- **Mitigazione umana/organizzativa:** Rendere obbligatorie le pause strutturate (5 minuti ogni ora) per gli analisti durante i periodi ad alto volume.
- **Mitigazione dei processi:** Sviluppare e applicare alberi decisionali standardizzati o runbook per i tipi di avviso comuni per ridurre il carico cognitivo di decidere *come* rispondere.