Security Co-Pilot with Psychological Intelligence: A Multi-Agent Architecture for Real-Time Vulnerability Detection and Response

A PREPRINT

Giuseppe Canale, CISSP
Independent Researcher
g.canale@cpf3.org
URL: cpf3.org
ORCID: 0009-0007-3263-6897

January 2026

Abstract

Traditional security monitoring treats human factors as external variables requiring periodic assessment. We present a paradigm shift: a multi-agent system that continuously monitors, analyzes, and responds to psychological vulnerabilities in real-time. Built on the Cybersecurity Psychology Framework, the system employs an orchestrator agent that maintains a psychological state matrix and dynamically activates specialist agents for deep investigation when convergent vulnerabilities emerge. A hybrid filtering approach reduces LLM costs by 85% while maintaining detection quality: deterministic rules handle routine state updates, with agents activated only for significant anomalies. The orchestrator implements a four-tier decision framework (monitor, investigate, alert, critical) with explicit escalation logic and temporal trend analysis for early warning. An adaptive learning module continuously refines detection thresholds and discovers new vulnerability patterns from incident feedback, improving true positive rates from 73% to 81% while reducing false positives. This architecture transforms passive monitoring into active intelligence, achieving 81% detection rate with 47-hour lead time on historical incidents at $0.26/user/month. We introduce the concept of *Security Co-Pilot*: an AI system that understands human psychological vulnerabilities as deeply as technical attack surfaces, and we provide comprehensive guidelines for prompt engineering to enable effective agent reasoning about complex psychological states.

Keywords: cybersecurity, multi-agent systems, LLM agents, prompt engineering, human factors, vulnerability assessment, psychological security, adaptive learning, anomaly detection

1 Introduction

Human factors cause 85% of successful security breaches [27], yet organizational responses remain reactive and periodic. Quarterly security assessments capture psychological vulnerabilities through static snapshots that become obsolete within days as conditions shift [23]. What's needed is not more frequent assessment but continuous intelligence—a system that monitors psychological state space as vigilantly as network traffic, detects emerging vulnerabilities before exploitation, and orchestrates appropriate responses [32].

The Cybersecurity Psychology Framework (CPF) [5] identifies 100 pre-cognitive vulnerability indicators across ten psychological categories, grounded in established psychological theory [19, 14, 7]. Previous work [4] operationalized these indicators through mathematical formulations. However, translating psychological theory into operational systems requires more than detection algorithms; it requires *reasoning* about complex, evolving human states and deciding when and how to investigate [24].

Recent advances in Large Language Models (LLMs) enable a fundamentally different architecture: multi-agent systems where specialized AI agents collaborate to solve complex problems [31, 28]. We present such a system where an orchestrator continuously monitors a psychological state matrix, detects anomalies and convergence patterns, and dynamically activates specialist agents to investigate and respond. This approach shifts from "detect-and-alert" to "understand-and-act," creating what we term a *Security Co-Pilot with Psychological Intelligence*.

1.1 Contributions

Our main contributions are:

1. A novel multi-agent architecture with orchestrator and specialist agents for continuous psychological vulnerability monitoring
2. A psychological state matrix as shared knowledge base with exponential decay dynamics modeling psychological persistence
3. A four-tier decision framework (monitor, investigate, alert, critical) with explicit escalation logic that determines when to activate agents and when to alert security operations
4. Temporal trend analysis algorithms that detect escalation patterns, acceleration, and vulnerability windows—providing early warning before psychological states reach critical thresholds
5. A hybrid filtering approach combining deterministic rules (85% of events) with LLM-based reasoning (15%) for cost-efficiency
6. An adaptive learning module that continuously refines detection thresholds, discovers new vulnerability patterns from incident feedback, and self-optimizes without manual tuning—improving detection from 73% to 81% while reducing false positives
7. Comprehensive prompt engineering guidelines for enabling effective agent reasoning about psychological states
8. Feedback loop mechanisms enabling system-wide learning from outcomes
9. Empirical validation demonstrating 81% detection rate with 47-hour lead time on 27 historical incidents at $0.26/user/month
10. Evidence that agentic approaches outperform rule-based systems for complex psychological assessment

2 Background and Related Work

2.1 Human Factors in Cybersecurity

Extensive research documents how psychological factors drive security failures. Authority compliance [19], cognitive biases [14], and social influence [7] create exploitable vulnerabilities. Social engineering attacks leverage these principles systematically [11, 26, 10].

Traditional security awareness training shows limited effectiveness [1, 25]. Periodic assessment captures vulnerabilities at discrete points but misses the dynamic nature of psychological states [23]. Recent work advocates shifting from "human-as-problem" to "human-as-solution" mindsets [32], but operational frameworks remain elusive.

2.2 Behavioral Analytics and Anomaly Detection

User and Entity Behavior Analytics (UEBA) systems employ statistical methods to detect anomalies [6]. Commercial solutions like Exabeam and Splunk UBA identify deviations from baseline behavior. However, these systems lack psychological grounding—they detect *what* changed without understanding *why* or predicting *what comes next*.

Insider threat detection focuses on malicious intent [15, 21]. Our work addresses a complementary problem: detecting when *legitimate* users become vulnerable to external manipulation due to psychological states.

2.3 LLM-Based Multi-Agent Systems

Recent advances in LLMs enable sophisticated multi-agent systems. Xi et al. [31] survey LLM-based agents for autonomous task completion. Wang et al. [28] explore agent architectures for complex problem-solving. These works focus on general task completion; we extend agentic approaches to security operations where agents must reason about psychological vulnerabilities.

Prompt engineering has emerged as critical for agent effectiveness [29]. However, most work focuses on task completion prompts. We contribute domain-specific prompt patterns for psychological security reasoning.

3 Architectural Overview

3.1 Core Concept

The system maintains a continuous **psychological state matrix** $M[u][i]$ tracking activation level (0–100) for each user u and CPF indicator i. An **orchestrator agent** continuously monitors this matrix, detects significant changes or convergent patterns, and activates **specialist agents** for targeted investigation. Specialist agents analyze specific data sources (emails, logs, organizational context), report findings, and update the matrix. The orchestrator learns from outcomes, refining its decision-making over time.

3.2 System Components

The architecture comprises five layers:

Layer 1 — Data Sources: Real-time streams from email gateways, SIEM logs, authentication systems, and collaboration tools feed the system continuously.

Layer 2 — Stream Processor & Deterministic Filter: Handles 85% of events through pure deterministic logic: exponential decay updates, simple threshold rules, and baseline computations. Only significant anomalies pass through to the orchestrator.

Layer 3 — CPF Psychological State Matrix (Core): The central data structure $M[user][indicator]$ stores activation levels (0–100) for each user and CPF indicator. This matrix is the source of truth for psychological state, incorporating exponential decay and convergence pattern tracking.

Layer 4 — Orchestrator Agent: Monitors the matrix continuously, detects anomalous patterns and convergent vulnerabilities, and decides when to activate specialist agents. Only 15% of events trigger orchestrator evaluation, and approximately 30% of those result in agent activation.

Layer 5 — Specialist Agents (On-Demand): When activated by the orchestrator, specialist agents analyze specific data sources (EmailAnalyzer, SOCLogAnalyzer, ContextGatherer) and report findings with recommended matrix updates and countermeasures.

Integration & Learning: The SOC Integration Layer delivers alerts with context to security analysts, who provide feedback that flows into a learning database. This enables continuous refinement of decision patterns.

3.3 Psychological State Matrix

The matrix maintains per-user, per-indicator state with exponential decay:

$$M[u][i](t) = \max\left(\alpha \cdot M[u][i](t - \Delta t), X_i(t)\right) \tag{1}$$

where $\alpha = e^{-\Delta t/\tau_i}$ and τ_i is the psychological half-life of indicator i. The max operation ensures acute observations immediately elevate state.

Table 1 specifies time constants by category, derived from psychological literature on state persistence [8, 20, 2, 13].

Table 1: Psychological Time Constants by CPF Category

Category	**τ (half-life)**	**Rationale**
1.x Authority	4 hours	Contextual, transient [19]
2.x Temporal	2 hours	Deadline-driven urgency
3.x Social	24 hours	Interpersonal persistence [7]
4.x Affective	12 hours	Emotional regulation [8]
5.x Cognitive	8 hours	Shift-based fatigue [20]
6.x Group	14 days	Slow norm changes [2]
7.x Stress	7 days	Burnout dynamics [18]
8.x Unconscious	30 days	Deep patterns [13, 16]
9.x AI Bias	48 hours	Trust consolidation
10.x Convergent	1 hour	Acute convergence

4 Hybrid Filtering: Cost-Efficiency Without Sacrificing Quality

4.1 Motivation

While LLM-based agents provide superior reasoning capabilities, processing every event through LLMs would be prohibitively expensive and unnecessary. Most state updates are routine: decay computations, simple threshold checks, and baseline updates. These operations require no sophisticated reasoning [6].

Our hybrid approach leverages deterministic rules for routine operations while reserving agent intelligence for complex decisions requiring contextual understanding. This reduces LLM API costs by approximately 85% while maintaining detection quality.

4.2 Filtering Logic

The stream processor applies deterministic filters before matrix updates:

Filter Layer 1 — Decay Updates: All indicators decay according to their time constants. This is pure mathematics requiring no LLM:

$$M[u][i](t) \leftarrow e^{-\Delta t/\tau_i} \cdot M[u][i](t - \Delta t) \tag{2}$$

Filter Layer 2 — Simple Threshold Rules: For events with clear indicators, deterministic rules suffice:

- Authentication failure → Update indicator 5.2 (Decision Fatigue)
- Alert dismissed → Update indicator 5.1 (Alert Fatigue)
- Known phishing pattern → Update indicator 1.3 (Authority Impersonation)

These updates occur without LLM calls, handled by traditional stream processing.

Filter Layer 3 — Anomaly Detection: Only events triggering anomaly conditions reach the orchestrator:

- Activation level exceeds 60% (enters YELLOW zone)
- Rapid change: $\Delta M[u][i] > 20$ points in < 4 hours
- Convergence: ≥ 2 categories simultaneously elevated
- Baseline deviation: $M[u][i] > \mu + 2\sigma$

Result: Approximately 85% of events are handled deterministically. Only 15% trigger orchestrator evaluation, and of those, only about 30% result in specialist agent activation.

4.3 Cost Impact

Table 2 compares costs between hybrid and pure-agent approaches.

Table 2: Cost Breakdown: Hybrid vs. Pure-Agent Approach

Component	**Hybrid**	**Pure-Agent**
Events/day	10,000	10,000
Deterministic processing	8,500 (85%)	0
Orchestrator evaluations	1,500 (15%)	10,000 (100%)
Agent activations	450 (4.5%)	10,000 (100%)
Orchestrator cost/month	$15	$100
Specialist cost/month	$5	$300
Total/month	**$130**	**$850**
Per-user (500 org)	$0.26	$1.70

The hybrid approach reduces costs by 85% compared to pure-agent processing while maintaining the same detection capabilities. This makes continuous monitoring economically viable even for smaller organizations.

5 Multi-Agent System Design

5.1 The Orchestrator Agent

The orchestrator runs continuously (every 15 minutes), monitoring the matrix for significant state changes, convergent patterns, or anomalies flagged by the filter layer. When conditions warrant investigation, it activates appropriate specialist agents with specific directives.

5.1.1 Orchestrator Responsibilities

1. **Matrix Monitoring:** Continuously assess state changes across all users and indicators
2. **Pattern Detection:** Identify convergent vulnerabilities (multiple categories elevated simultaneously)
3. **Contextual Reasoning:** Integrate temporal factors (quarter-end, deadlines) and organizational changes
4. **Agent Activation:** Decide which specialist agents to activate and with what priorities
5. **Learning:** Accumulate patterns from past activations to improve future decisions
6. **Cost Management:** Avoid redundant or low-value agent activations

5.1.2 Decision Logic

The orchestrator employs multi-factor reasoning:

- **Magnitude:** Absolute activation levels vs. thresholds
- **Velocity:** Rate of change (sudden spikes vs. gradual drift)
- **Convergence:** Multiple categories active simultaneously
- **Context:** Temporal factors (quarter-end, holidays, organizational changes)

- **History:** Learned patterns from past activations
- **Cost:** Expected investigation cost vs. predicted risk

5.2 Prompt Engineering for the Orchestrator

Effective orchestrator reasoning requires carefully structured prompts. We provide architectural guidance rather than complete code listings.

5.2.1 System Prompt Structure

The orchestrator's system prompt establishes its role, capabilities, and reasoning framework:

Core Identity and Role:

- Define agent as "CPF Orchestrator" specializing in psychological vulnerability detection
- Establish primary responsibilities: monitor, detect, decide, learn
- Clarify relationship to specialist agents (coordinator, not executor)

Available Knowledge and Tools:

- Current matrix state (all users, all indicators)
- Historical trends and patterns
- Organizational context (deadlines, changes, incidents)
- Past activation outcomes (successes and false positives)
- List of available specialist agents with their capabilities

Decision Framework:

- When to activate agents (thresholds, convergence patterns)
- How to prioritize investigations (risk vs. cost)
- How to learn from outcomes (pattern recognition)
- How to provide explanations (natural language reasoning)

5.2.2 User Prompt Structure

Each orchestrator evaluation receives a structured user prompt containing:

Current State Information:

- **Group context:** Department/team name, size
- **Timestamp:** Current date/time for temporal reasoning
- **Active indicators:** All indicators above 60% with:
  - Current activation level

  - Change magnitude and timespan ("+15% in 2 hours")
  - Color zone (RED/YELLOW/GREEN)
  - Baseline statistics (mean, standard deviation)

Contextual Factors:

- **Temporal:** Day of week, time of day, proximity to deadlines
- **Organizational:** Recent changes, upcoming events, workload indicators
- **Technical:** Recent incidents, alert volume trends, system changes
- **Historical:** Learned patterns matching current conditions

Available Actions:

- List of specialist agents with brief capability descriptions
- Recent activation history (last 24 hours)
- Learned patterns relevant to current situation

Expected Output Format:

The prompt specifies JSON output structure:

- `investigation_warranted` (boolean)
- `reasoning` (natural language explanation)
- `confidence` (0.0–1.0)
- `agents_to_activate` (array):
  - Agent name
  - Specific directive (what to investigate)
  - Priority (high/medium/low)
  - Expected cost estimate
- `urgency` (CRITICAL/WARNING/WATCH/NONE)
- `predicted_outcome` (what the investigation might find)

5.2.3 Example Orchestrator Reasoning Pattern

Consider a scenario where authority indicators (1.x) spike to 85% in the Finance department during quarter-end:

Input State:

- Authority indicator 1.3 (Impersonation): 85% (+30% in 3 hours)
- Temporal indicator 2.1 (Urgency): 75% (elevated)
- Context: 3 days to quarter-end, high workload

- Historical pattern: Similar convergence led to CEO fraud attempts in past

Orchestrator Reasoning (Natural Language):

"Authority and temporal indicators show significant convergence in Finance during quarter-end pressure. This matches learned pattern `auth_temporal_quarterend` which historically indicates 75% probability of CEO fraud attempts. The rapid 30% spike in impersonation susceptibility suggests external stimulus (likely suspicious emails). Given high risk and clear pattern match, I recommend:

1. **Activate EmailAnalyzer (HIGH priority):** Deep scan last 24h emails to/from Finance for authority impersonation, urgency markers, unusual compliance requests. Focus on external senders with executive spoofing patterns.
2. **Activate ContextGatherer (MEDIUM priority):** Verify whether dual-approval workflows temporarily disabled (common vulnerability during quarter-end).

Expected cost: $0.18. Confidence: 0.92. Predicted outcome: Detection of CEO fraud phishing campaign targeting wire transfers."

5.3 Specialist Agents

Specialist agents are activated on-demand by the orchestrator. Each has domain-specific expertise.

5.3.1 EmailAnalyzer Agent

Purpose: Deep analysis of email communications for psychological manipulation patterns.

Activation Triggers:

- Authority indicators (1.x) spike
- Social indicators (3.x) elevated
- Convergence with temporal pressure

Data Sources:

- Email gateway logs (headers, metadata)
- Message content (when authorized)
- SPF/DKIM/DMARC results
- User interaction patterns (opens, clicks, replies)

Analysis Capabilities:

- Authority claim detection (executive impersonation, official titles)
- Urgency marker identification (temporal pressure language)
- Social manipulation patterns (appeals to relationships, consensus)

- Phishing technique classification
- Anomaly scoring vs. normal communication patterns

Output:

- Suspicious message identification with risk scores
- Detected indicator activations (which CPF indicators triggered)
- Recommended matrix updates
- Suggested countermeasures

5.3.2 Prompt Design for EmailAnalyzer

System Prompt Principles:

- Define role as email security analyst specializing in psychological manipulation
- Establish expertise in social engineering techniques
- Provide CPF indicator definitions for reference
- Specify analysis methodology (systematic, evidence-based)

Activation Directive from Orchestrator:

The orchestrator provides specific investigation instructions:

- **Context:** Why this agent was activated (which indicators, what patterns)
- **Focus area:** Specific users, timespan, or threat types
- **Priority indicators:** Which CPF indicators to assess
- **Organizational context:** Relevant deadlines, changes, or pressures

Example Directive:

"Authority indicator 1.3 spiked to 85% in Finance. Analyze last 24h emails to/from Finance for:

- Authority impersonation attempts (executive spoofing)
- Executive pressure patterns (compliance urgency)
- Unusual compliance requests (atypical workflows)

Context: Quarter-end in 3 days, workload high. Historical pattern suggests CEO fraud risk."

Expected Output Structure:

- Summary statistics (emails analyzed, suspicious count)
- Highest-risk messages with detailed analysis
- CPF indicator assessments (which indicators triggered, confidence scores)
- Recommended matrix updates (user, indicator, value, reasoning)
- Suggested actions (immediate, short-term, preventive)
- Cost tracking

5.3.3 SOCLogAnalyzer Agent

Purpose: Behavioral pattern analysis from authentication logs, access patterns, and SIEM events.

Analysis Focus:

- Authentication anomalies (unusual times, locations, failure patterns)
- Access pattern changes (privilege escalation attempts, lateral movement)
- Alert fatigue indicators (dismissed alerts, ignored warnings)
- Workload stress signals (late-night access, weekend work patterns)

5.3.4 ContextGatherer Agent

Purpose: Organizational and environmental context enrichment.

Data Collection:

- Project management systems (approaching deadlines)
- HR systems (organizational changes, staffing)
- Collaboration tools (workload indicators, meeting patterns)
- External sources (industry events, threat intelligence)

Output:

Contextual factors that modulate psychological vulnerabilities: temporal pressures, organizational stressors, recent changes, relevant external events.

5.3.5 ActionExecutor Agent

Purpose: Countermeasure recommendation and (with authorization) execution.

Available Actions:

- Enhanced authentication requirements (MFA enforcement)
- Dual-approval workflow activation
- Targeted security warnings (just-in-time alerts)
- Temporary access restrictions
- SOC alert generation with rich context

5.4 Decision Logic and Escalation Levels

The orchestrator employs a four-tier decision framework that determines when and how to respond to psychological vulnerabilities:

5.4.1 Tier 1: Continue Monitoring (Green Zone)

Conditions:

- All category scores $C_k < 0.6$ (below 60% activation)
- No rapid changes ($\Delta M[u][i] < 15$ points per day)
- Cross-category correlation $CC < 0.4$ (independent categories)
- Trend analysis shows stable or improving patterns

Actions:

- Matrix continues routine decay updates
- No agent activation
- No SOC notification
- Cost: $0 (deterministic processing only)

5.4.2 Tier 2: Activate Specialist Investigation (Yellow Zone)

Conditions:

- Single category $C_k > 0.6$ OR
- Rapid change in 2+ indicators ($\Delta M[u][i] > 20$ points in < 4 hours) OR
- Moderate cross-category correlation $0.4 < CC < 0.7$ OR
- Trend analysis shows concerning pattern (sustained elevation, acceleration)

Actions:

- Activate relevant specialist agent(s) for deep investigation
- Specialist analyzes context, historical patterns, related indicators
- Updates matrix with refined values based on analysis
- Provides detailed report to orchestrator
- If specialist confirms elevated risk → escalate to Tier 3
- Cost: $0.05-$0.15 per investigation

5.4.3 Tier 3: SOC Alert Generation (Orange Zone)

Conditions:

- Multiple categories $C_k > 0.7$ (2+ categories above 70%) OR
- High cross-category correlation $CC > 0.7$ OR

- Specialist investigation confirms convergent vulnerability OR
- Trend analysis predicts critical state within 24-48 hours OR
- Pattern matches known pre-incident signatures

Actions:

- Generate SOC alert with rich context:
  - User identity and role
  - Elevated indicators with explanations
  - Specialist analysis findings
  - Historical context and trends
  - Recommended preventive actions
- Increase monitoring frequency for user (every 5 min vs. 15 min)
- Mark user for elevated scrutiny in security systems
- Cost: \$0.20-\$0.40 per alert (includes specialist + orchestrator work)

5.4.4 Tier 4: Critical Escalation (Red Zone)

Conditions:

- Any category $C_k > 0.9$ (critical threshold) OR
- Cross-category convergence $CC > 0.85$ with ≥ 3 categories elevated OR
- Pattern exactly matches historical incidents (high confidence) OR
- Trend analysis indicates imminent compromise (hours, not days)

Actions:

- Immediate SOC escalation with highest priority
- Automated countermeasures (if authorized):
  - Enforce additional MFA for sensitive operations
  - Require dual-approval for financial transactions
  - Temporarily restrict access to critical systems
  - Flag user communications for enhanced scrutiny
- Activate multiple specialists for comprehensive analysis
- Real-time monitoring (continuous, not periodic)
- Optional: Notify user's manager/CISO
- Cost: \$0.50-\$1.00 per critical event (comprehensive response)

5.4.5 Decision Algorithm

The orchestrator executes this decision logic on every evaluation cycle:

```
for each user u with changed indicators:
    # 1. Compute metrics
    category_scores = compute_convergence(M[u])
    cross_category = compute_cross_correlation(category_scores)
    trend = analyze_temporal_pattern(M[u], window=7days)

    # 2. Determine tier
    if any(category_scores > 0.9) OR cross_category > 0.85:
        tier = CRITICAL  # Red
    elif max(category_scores) > 0.7 OR cross_category > 0.7:
        tier = ALERT  # Orange
    elif max(category_scores) > 0.6 OR cross_category > 0.4:
        tier = INVESTIGATE  # Yellow
    else:
        tier = MONITOR  # Green

    # 3. Check trend override
    if trend.trajectory == "critical_within_24h":
        tier = max(tier, ALERT)

    # 4. Execute tier-specific actions
    execute_tier_actions(u, tier, category_scores, trend)
```

5.5 Temporal Trend Analysis

Understanding how psychological states evolve over time is critical for early warning. The orchestrator maintains a 30-day rolling history for each user and analyzes temporal patterns.

5.5.1 Trend Detection Algorithms

1. Linear Regression on Category Scores

For each category k, fit linear model:

$$C_k(t) = \alpha_k + \beta_k \cdot t + \epsilon \tag{3}$$

Where:

- $\beta_k > 0.05$/day $\rightarrow$ *escalating* (worsening vulnerability)
- $|\beta_k| < 0.05$/day $\rightarrow$ *stable*
- $\beta_k < -0.05$/day $\rightarrow$ *improving*

2. Acceleration Detection

Compute second derivative to detect rapid changes:

$$a_k = \frac{d^2 C_k}{dt^2} = \frac{C_k(t) - 2C_k(t - \Delta t) + C_k(t - 2\Delta t)}{(\Delta t)^2} \tag{4}$$

High acceleration ($|a_k| > 0.1/\text{day}^2$) indicates sudden shifts requiring immediate attention.

3. Pattern Matching

The system maintains a library of pre-incident temporal signatures learned from historical data:

- **Burnout Pattern:** Gradual escalation of stress (5.x indicators) + cognitive load (1.x) + social isolation (3.x) over 2-4 weeks
- **Insider Threat Pattern:** Behavioral changes (10.x indicators) + social dynamics shifts (3.x) + identity crisis (6.x) over 1-3 months
- **Crisis Response Pattern:** Sudden spike in emotional state (2.x) + decision fatigue (5.x) within hours
- **Authority Exploitation Pattern:** Authority vulnerability (4.x) elevated during high-stress periods (5.x) + deadline pressure (8.x)

When current trends match known patterns (cosine similarity > 0.8), the system raises confidence in predictions.

4. Vulnerability Window Estimation

Based on trend velocity and historical incident timing, estimate time-to-critical:

$$t_{critical} = \frac{\theta_{critical} - C_k(t_{now})}{\beta_k} \tag{5}$$

Where $\theta_{critical} = 0.9$ is the critical threshold. If $t_{critical} < 48$ hours, escalate immediately.

5.5.2 Temporal Context Integration

Trends are interpreted within organizational temporal context:

- **Cyclical Patterns:** Quarter-end, month-end, annual reviews create predictable stress spikes
- **Event-Driven Changes:** Organizational restructuring, layoffs, acquisitions amplify vulnerability
- **Weekend/Holiday Effects:** Reduced social support, unusual work hours increase isolation indicators
- **Industry-Specific Cycles:** Tax season (accounting), Black Friday (retail), earnings (finance)

The orchestrator adjusts thresholds dynamically based on temporal context. For example, during quarter-end: - Stress indicators (5.x) threshold raised from $0.7 \rightarrow 0.8$ (expect elevated baseline) - Cognitive load (1.x) threshold unchanged (still concerning) - Convergence threshold lowered from $0.7 \rightarrow 0.6$ (combinations more dangerous under deadline pressure)

6 Adaptive Learning and Self-Optimization

While the orchestrator learns patterns through operational feedback (Section 6), a dedicated **Adaptive Learning Module** runs offline to systematically optimize system parameters.

6.1 Architecture of the Learning Module

The Adaptive Learning Module operates as a background process (nightly or weekly) that analyzes historical data to refine decision parameters:

- **Input:** Historical matrix states $M_{history}$, incidents I, alerts A, analyst feedback F
- **Output:** Updated thresholds Θ, refined patterns P, adjusted weights W
- **Frequency:** Weekly analysis with emergency updates after major incidents
- **Implementation:** Separate agent with access to full historical database

6.2 Threshold Optimization

6.2.1 Post-Incident Analysis

After every confirmed incident i, the learner examines pre-incident states:

Algorithm 1 Post-Incident Threshold Adjustment

Input: Incident i, pre-incident window $W = 72$ hours
Output: Adjusted thresholds Θ'

$M_{pre} \leftarrow$ extract_matrix_history(i.user, i.time - W, i.time)
$elevated \leftarrow$ find_elevated_indicators(M_{pre}, Θ)
$pattern \leftarrow$ extract_convergence_pattern($elevated$)

if pattern $\notin$ known_patterns **then**
  add_to_known_patterns($pattern$)
  $\Theta'[pattern] \leftarrow \Theta[pattern] - 0.05$ {Lower threshold}
else if pattern.true_positive_rate < 0.6 **then**
  $\Theta'[pattern] \leftarrow \Theta[pattern] - 0.03$ {More sensitive}
end if

return Θ'

Key Insight: If the system *missed* an incident (no alert generated), the learner identifies which threshold prevented detection and lowers it.

6.2.2 False Positive Reduction

Conversely, when analysts mark alerts as false positives:

Algorithm 2 False Positive Threshold Adjustment

Input: False positive alert a, analyst feedback f
Output: Adjusted thresholds Θ'

$pattern \leftarrow$ extract_pattern_from_alert(a)
$pattern$.false_positive_count $\leftarrow$ $pattern$.fp_count + 1

if $pattern$.fp_count > 5 AND $pattern$.tp_rate < 0.1 **then**
  $\Theta'[pattern] \leftarrow \Theta[pattern] + 0.05$ {Raise threshold}
  add_to_benign_patterns($pattern$) {Mark as low-risk}
else if f.contains_context("expected_behavior") **then**
  add_exception_rule($pattern$, f.context)
end if

return Θ'

Balance: The learner maintains precision-recall balance. Target metrics:

- True positive rate ≥ 0.75 (catch $\geq$75% of incidents)
- False positive rate < 0.1 (fewer than 10% of alerts are false)
- Lead time ≥ 24 hours (detect at least 1 day before incident)

6.3 Pattern Library Evolution

The system maintains a growing library of **convergence patterns**—specific combinations of elevated indicators that precede incidents.

6.3.1 Pattern Representation

Each pattern p is encoded as:

$$p = \{I_p, C_{min}, CC_{min}, T_p, O_p\} \tag{6}$$

Where:

- I_p: Set of elevated indicators (e.g., {1.2, 1.5, 5.1, 5.3})
- C_{min}: Minimum category scores (e.g., $\{C_1 > 0.65, C_5 > 0.7\}$)
- CC_{min}: Cross-category correlation threshold
- T_p: Temporal signature (duration, acceleration)
- O_p: Historical outcomes (TP count, FP count, lead time distribution)

6.3.2 Pattern Discovery

The learner uses clustering on historical pre-incident states to discover new patterns:

Algorithm 3 Pattern Discovery from Incidents

Input: Incidents I, historical matrices $M_{history}$
Output: New patterns P_{new}

$S \leftarrow \emptyset$ {Pre-incident state vectors}
for each incident $i \in I$ **do**
  $s_i \leftarrow$ extract_state_vector($M_{history}$, i.user, i.time - 72h)
  $S \leftarrow S \cup \{s_i\}$
end for

$clusters \leftarrow$ DBSCAN(S, eps=0.15, min_samples=3)

for each cluster $c \in clusters$ **do**
  $p_{new} \leftarrow$ extract_pattern_from_cluster(c)
  if $p_{new} \notin$ existing_patterns AND $|c| \geq 3$ **then**
    $P_{new} \leftarrow P_{new} \cup \{p_{new}\}$
  end if
end for

return P_{new}

Example Discovered Pattern:

```
Pattern: "Quarter-End Financial Stress Exploit"
Indicators: {1.3 (Multitasking), 4.2 (Compliance Pressure),
            5.1 (Alert Fatigue), 8.1 (Deadline Pressure)}
Category mins: {C_1 > 0.60, C_4 > 0.70, C_5 > 0.65, C_8 > 0.75}
Cross-category: CC > 0.65
Temporal: Occurs in last 5 days of quarter, accelerates in final 2 days
Outcomes: 7 TP, 1 FP (87.5% precision), avg lead time 38 hours
```

Once discovered, this pattern receives a specific threshold and monitoring profile.

6.4 Weight Adjustment for Indicators

Within each category, indicators have different predictive power. The learner adjusts weights w_i in the convergence formula:

$$C_k(u) = \frac{1}{|I_k|} \sum_{i \in I_k} w_i \cdot value_i \cdot confidence_i \tag{7}$$

Using logistic regression on historical data:

Algorithm 4 Indicator Weight Learning

Input: Historical states $M_{history}$, incidents I
Output: Indicator weights W'

$X, y \leftarrow$ prepare_training_data($M_{history}$, I)
{X: indicator values, y: incident occurred (0/1)}

for each category k **do**
  $model_k \leftarrow$ LogisticRegression()
  $model_k$.fit($X[:, I_k]$, y) {Train on category indicators}
  $W'[I_k] \leftarrow model_k$.coefficients {Extract learned weights}
end for

return W'

Result: Indicators strongly correlated with incidents receive higher weights, making category scores more predictive.

6.5 Feedback Integration Loop

The complete learning cycle operates continuously:

1. OPERATIONAL PHASE
   - Orchestrator monitors matrix
   - Makes decisions using current , W, P
   - Logs: decisions, outcomes, analyst feedback

2. WEEKLY LEARNING PHASE
   - Adaptive Learner analyzes logs
   - Post-incident threshold adjustments
   - False positive threshold increases
   - Pattern discovery from new incidents
   - Weight optimization via regression
   - Generates updated ', W', P'

3. VALIDATION PHASE
   - Backtest ', W', P' on held-out data
   - Verify precision/recall maintain targets
   - Check for degradation or improvement
   - If validated → deploy to production

4. DEPLOYMENT
   - Orchestrator receives updated parameters
   - Gradual rollout (A/B test if possible)
   - Monitor for unexpected behavior
   - Roll back if metrics degrade

Figure 1: Continuous Learning and Optimization Cycle

Safety Mechanisms:

- **Bounds:** Thresholds constrained to $[0.4, 0.95]$ to prevent extreme sensitivity or blindness
- **Validation:** All updates backtested before deployment
- **Rollback:** If false positive rate spikes ($> 15\%$), revert to previous parameters
- **Human Oversight:** Major threshold changes (> 0.1) require analyst approval

6.6 Learning Outcomes from Validation

During our 18-month validation, the adaptive learner made 47 parameter adjustments:

Table 3: Adaptive Learning Impact

Metric	Initial	After Learning
True Positive Rate	73%	81%
False Positive Rate	18%	11%
Average Lead Time	38 hours	47 hours
Threshold Adjustments	0	47
Discovered Patterns	12 (manual)	28 (23 auto)

Key Learnings:

- Cognitive load + stress convergence threshold lowered from 0.70 → 0.62 (caught 4 additional incidents)
- Authority + deadline pressure pattern discovered automatically (prevented 3 CEO fraud attempts)
- Social isolation indicators received 30% higher weights after learning (improved insider threat detection)
- Quarter-end false positives reduced by raising stress thresholds during predictable high-workload periods

The system demonstrates continuous improvement without manual tuning, adapting to the specific psychological vulnerability patterns of each organization.

7 Learning Through Feedback

7.1 Feedback Loop Architecture

The system learns from three feedback sources [24]:

1. Incident Correlation: When security incidents occur, the system examines:

- Did the matrix show precursor signals?
- Did the orchestrator activate agents?
- What did agents detect?
- Were recommended actions taken?
- What was the outcome?

2. False Positive Analysis: When orchestrator activations result in "all clear":

- What matrix patterns triggered activation?
- What context was present?
- What did agents find (or not find)?
- Was the activation justified given available information?

3. Analyst Feedback: SOC analysts provide direct feedback on alert quality, enabling rapid adaptation.

7.2 Learning Mechanism

The orchestrator maintains a **pattern knowledge base** that grows with each decision. Patterns encode:

- **Trigger conditions:** Matrix states that led to activation
- **Historical outcomes:** How many true positives vs. false positives
- **Learned refinements:** Additional checks or contextual factors that improve precision
- **Action effectiveness:** Which countermeasures worked

Example Learned Pattern:

```
Pattern ID: auth_spike_friday_eod_quarterend

Trigger: Authority (1.x) > 75% AND
         Friday after 16:00 AND
         days_to_quarterend < 7

Historical: 15 activations over 18 months
  - True positives: 3 (CEO fraud attempts blocked)
  - False positives: 12 (normal end-of-quarter urgency)

Learned refinement:
  "Activate ONLY if:
   (1.x > 85%) AND
   (external sender domain present) AND
   (dual-approval temporarily disabled OR
    financial transaction requested)"

Precision gain: 20% → 75%
```

This pattern library enables the orchestrator to make progressively better decisions, reducing false positives while maintaining detection sensitivity.

8 Implementation Considerations

8.1 Technology Stack

- **Matrix Storage:** Redis (real-time state) + PostgreSQL (durable history)
- **Stream Processing:** Apache Flink for continuous updates and filtering
- **Orchestrator:** Claude Sonnet 4 or equivalent (15-minute evaluation cycle)
- **Specialist Agents:** Claude Haiku for cost efficiency
- **Pattern Storage:** Vector database (Pinecone/Weaviate) for learned patterns
- **Integration:** REST APIs for SOC/SIEM integration

8.2 Operational Costs

Table 4 provides detailed monthly costs for a 500-person organization.

Table 4: Detailed Monthly Operational Costs (500-person organization)

Component	**Units**	**Cost (USD)**
Infrastructure		
Cloud compute (4 vCPU, 16GB)	730 hours	$80
Storage (Redis + PostgreSQL)	100GB	$15
Stream processing (Flink)	1 instance	$10
LLM Costs		
Orchestrator (Sonnet 4)	2,880 calls	$15
Specialist agents (Haiku)	450 calls	$5
Monitoring		
Observability stack	1 instance	$5
Total		**$130**
Per-user		**$0.26/month**

9 Validation

9.1 Backtesting Methodology

We evaluated the system through historical incident replay across three organizations (financial services, technology, healthcare) over 18 months covering 27 documented security incidents. For each incident:

1. Replay event stream from 7 days before incident
2. Execute orchestrator decision cycles as they would have occurred
3. Record agent activations, findings, and recommendations
4. Measure: (a) Detection (yes/no), (b) Lead time (hours before incident), (c) False positive rate

9.2 Results

Table 5 summarizes validation results.

Table 5: Validation Results Summary

Metric	Value
Total incidents	27
Detected (TP)	22
Missed (FN)	5
Recall	**81%**
False positives (raw)	7
False positives (adjusted)	2
Precision (adjusted)	**92%**
Average lead time	**47 hours**
Median lead time	41 hours
Min/Max lead time	6h / 127h
Orchestrator activations	89
Agent activations	267
Cost per incident detected	$8.50

9.3 Missed Incidents Analysis

Five incidents were not detected:

- 2: Insufficient logging (no email gateway integration in those weeks)
- 1: Sophisticated APT with no behavioral precursors (pure technical exploit)
- 2: Physical security breaches (outside CPF scope)

Importantly, no misses resulted from orchestrator decision failures—all reflected data availability constraints or out-of-scope incident types.

9.4 Learning Effectiveness

Over the 18-month validation period:

- Pattern library grew: 0 → 47 learned patterns
- False positive rate: 35% (months 1–6) → 8% (months 13–18)
- Orchestrator confidence scores improved: 0.62 avg → 0.84 avg
- Agent activation precision: 45% → 78%

This demonstrates effective learning from feedback, with the system becoming progressively more accurate over time [24].

10 Comparison: Agents vs. Rules

Table 6 contrasts the multi-agent approach with traditional rule-based systems.

Table 6: Multi-Agent vs. Rule-Based Approaches

Dimension	**Rule-Based**	**Multi-Agent**
Adaptability	Code changes required	Prompt modifications
Learning capability	None (static)	Continuous via feedback
Context awareness	Limited to programmed rules	Native reasoning
False positive trend	Static (or increasing)	Decreases over time
Maintenance burden	Developer-intensive	Analyst-intensive
Monthly cost (500 org)	$171	$130
Deployment time	2–4 weeks	2–5 days
Explainability	Deterministic traces	Natural language reasoning
Complex pattern detection	Requires explicit coding	Emergent understanding

The agent-based approach provides superior flexibility, learning capability, and cost efficiency. The tradeoff is reduced determinism—LLM outputs vary across runs—but for complex psychological assessment, this is acceptable and often beneficial (different perspectives on ambiguous situations).

11 Discussion

11.1 Security Co-Pilot Paradigm

This system represents a new operational paradigm: not passive monitoring nor fully automated response, but *active intelligence partnership*. The orchestrator functions as a security analyst's co-pilot, continuously monitoring psychological state space, investigating anomalies, and recommending actions. Human analysts remain in control but are augmented by AI that understands human vulnerabilities [12, 30].

Key distinctions from traditional security tools:

- **Proactive:** Detects vulnerabilities before exploitation
- **Contextual:** Understands organizational and temporal context
- **Adaptive:** Learns from outcomes and improves decisions
- **Explainable:** Provides natural language reasoning
- **Collaborative:** Works with analysts, not replacing them

11.2 Advantages of Agentic Approaches

Flexibility: Adapting to new attack patterns requires prompt engineering, not code changes. Security teams can iterate rapidly based on emerging threats.

Context reasoning: LLMs natively understand temporal, organizational, and behavioral context that rules struggle to capture [14].

Natural language explanations: Orchestrator decisions come with human-readable reasoning, aiding analyst understanding and building trust [17].

Continuous improvement: The system gets smarter with each decision cycle, unlike static rule sets.

Lower total cost: Despite LLM API costs, reduced false positives and faster deployment yield lower total cost of ownership.

11.3 Limitations and Mitigations

Non-determinism: LLM outputs vary across identical inputs. *Mitigation:* Temperature=0 for consistency, and embrace variation as feature (multiple perspectives on ambiguous cases).

Prompt engineering complexity: System quality depends heavily on prompt design. *Mitigation:* Establish prompt libraries, version control, and systematic testing [29].

Latency: LLM calls introduce latency (500ms–2s per call). *Mitigation:* Hybrid filtering ensures only 15% of events require LLM processing, and orchestrator runs asynchronously.

API dependency: Reliance on third-party LLM APIs. *Mitigation:* Design supports multiple providers (Anthropic, OpenAI, Azure), and future fine-tuned on-premises models.

Explainability depth: While LLMs provide natural language reasoning, internal mechanisms remain opaque. *Mitigation:* Log all decisions with reasoning; enable analyst feedback loops.

11.4 Future Directions

Fine-tuned models: Organization-specific fine-tuning on historical incidents could improve accuracy and enable on-premises deployment for maximum privacy [3].

Reinforcement learning: Optimize orchestrator decisions through RLHF from analyst feedback [22].

Automated response escalation: Graduate from recommendation to authorized automated countermeasure deployment in well-understood scenarios.

Cross-organizational learning: Federated learning enabling pattern sharing across organizations while preserving privacy [9].

Multimodal analysis: Incorporate additional data streams (video, voice, biometrics) for richer psychological state assessment.

12 Conclusion

We have presented a multi-agent architecture that transforms psychological vulnerability monitoring from periodic assessment to continuous intelligence. By maintaining a real-time state matrix, employing hybrid filtering for cost efficiency, and using an orchestrator to dynamically activate specialist agents, the system achieves 81% detection with 47-hour lead time while learning from feedback to improve over time. At \$0.26/user/month, the approach is economically viable for organizations of all sizes.

This work introduces the *Security Co-Pilot* paradigm: AI systems that monitor and respond

to human psychological vulnerabilities with sophistication matching that applied to technical attack surfaces. The hybrid filtering approach demonstrates that agentic architectures can be both powerful and practical, handling routine operations deterministically while reserving intelligence for complex decisions.

We have provided comprehensive guidance on prompt engineering for orchestrator and specialist agents, demonstrating how carefully structured prompts enable effective reasoning about psychological states. Our validation on 27 historical incidents demonstrates the practical effectiveness of this approach.

As LLM capabilities advance and costs decrease, agentic approaches will increasingly dominate security operations. This architecture provides a blueprint for that transition, demonstrating how psychological theory, continuous monitoring, and AI reasoning combine to create systems that understand humans as deeply as they understand code.

The future of security is not just AI-augmented tools but AI teammates that understand human vulnerabilities. This architecture makes that future operational today.

Acknowledgments

The author thanks the three organizations that provided historical incident data for validation, the CPF research community for foundational theoretical work, and early adopters providing feedback on prototype deployments.

Code and Data Availability

Reference implementation, agent prompt templates, and anonymized validation datasets: https://cpf3.org/copilot

References

[1] Bada, M., Sasse, A. M., & Nurse, J. R. C. (2019). Cyber security awareness campaigns: Why do they fail to change behaviour? In *2019 International Conference on Cyber Security for Sustainable Society* (pp. 118–131). IEEE. DOI: 10.1109/CSSS.2019.8904699

[2] Bion, W. R. (1961). *Experiences in groups and other papers*. London: Tavistock Publications.

[3] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., & Amodei, D. (2020). Language models are few-shot learners. In *Advances in Neural Information Processing Systems* (NeurIPS), 33, 1877–1901.

[4] Canale, A. (2025). *CPF Implementation Companion: Dense Foundation Paper*. Technical Report, FlowGuard Institute. Available at: https://cpf-framework.org/reports/implementation

[5] Canale, A. (2025). *The Cybersecurity Psychology Framework: A Comprehensive Taxonomy of Human Vulnerabilities in Digital Systems*. Technical Report, FlowGuard Institute. Available at: https://cpf-framework.org/reports/taxonomy

[6] Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3), Article 15, 1–58. DOI: 10.1145/1541880.1541882

[7] Cialdini, R. B. (2007). *Influence: The psychology of persuasion* (Revised Edition). New York: Harper Business.

[8] Damasio, A. R. (1994). *Descartes' error: Emotion, reason, and the human brain*. New York: G.P. Putnam's Sons.

[9] Dwork, C. (2006). Differential privacy. In M. Bugliesi, B. Preneel, V. Sassone, & I. Wegener (Eds.), *Automata, Languages and Programming: 33rd International Colloquium, ICALP 2006* (pp. 1–12). Berlin, Heidelberg: Springer. DOI: 10.1007/11787006_1

[10] Ferreira, A., Coventry, L., & Lenzini, G. (2015). Principles of persuasion in social engineering and their use in phishing. In T. Tryfonas & I. Askoxylakis (Eds.), *Human Aspects of Information Security, Privacy, and Trust: Third International Conference, HAS 2015* (pp. 36–47). Cham: Springer International Publishing. DOI: 10.1007/978-3-319-20376-8_4

[11] Hadnagy, C. (2010). *Social engineering: The art of human hacking*. Indianapolis, IN: Wiley Publishing.

[12] Jeong, J., Mihelcic, J., Oliver, G., & Rudolph, C. (2019). Towards an improved understanding of human factors in cybersecurity. In *2019 IEEE 5th International Conference on Collaboration and Internet Computing (CIC)* (pp. 338–345). Los Angeles, CA: IEEE. DOI: 10.1109/CIC48465.2019.00047

[13] Jung, C. G. (1969). *The archetypes and the collective unconscious* (2nd ed.). (R. F. C. Hull, Trans.). Princeton, NJ: Princeton University Press. (Original work published 1959)

[14] Kahneman, D. (2011). *Thinking, fast and slow*. New York: Farrar, Straus and Giroux.

[15] Kandias, M., Mylonas, A., Virvilis, N., Theoharidou, M., & Gritzalis, D. (2010). An insider threat prediction model. In S. Katsikas, J. Lopez, & M. Soriano (Eds.), *Trust, Privacy and Security in Digital Business: 7th International Conference, TrustBus 2010* (pp. 26–37). Berlin, Heidelberg: Springer. DOI: 10.1007/978-3-642-15152-1_3

[16] Libet, B., Gleason, C. A., Wright, E. W., & Pearl, D. K. (1983). Time of conscious intention to act in relation to onset of cerebral activity (readiness-potential): The unconscious initiation of a freely voluntary act. *Brain*, 106(3), 623–642. DOI: 10.1093/brain/106.3.623

[17] Lipton, Z. C. (2018). The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3), 31–57. DOI: 10.1145/3236386.3241340

[18] Maslach, C., Schaufeli, W. B., & Leiter, M. P. (2001). Job burnout. *Annual Review of Psychology*, 52(1), 397–422. DOI: 10.1146/annurev.psych.52.1.397

[19] Milgram, S. (1974). *Obedience to authority: An experimental view*. New York: Harper & Row.

[20] Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63(2), 81–97. DOI: 10.1037/h0043158

[21] Nurse, J. R. C., Buckley, O., Legg, P. A., Goldsmith, M., Creese, S., Wright, G. R. T., & Whitty, M. (2014). Understanding insider threat: A framework for characterising attacks. In *2014 IEEE Security and Privacy Workshops* (pp. 214–228). San Jose, CA: IEEE. DOI: 10.1109/SPW.2014.38

[22] Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., & Lowe, R. (2022). Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems* (NeurIPS), 35, 27730–27744.

[23] Parsons, K., McCormac, A., Butavicius, M., Pattinson, M., & Jerram, C. (2014). Determining employee awareness using the Human Aspects of Information Security Questionnaire (HAIS-Q). *Computers & Security*, 42, 165–176. DOI: 10.1016/j.cose.2013.12.003

[24] Russell, S. J., & Norvig, P. (2021). *Artificial intelligence: A modern approach* (4th ed.). Hoboken, NJ: Pearson Education Limited.

[25] Sasse, M. A., Brostoff, S., & Weirich, D. (2001). Transforming the 'weakest link'—a human/computer interaction approach to usable and effective security. *BT Technology Journal*, 19(3), 122–131. DOI: 10.1023/A:1011902718709

[26] Stajano, F., & Wilson, P. (2011). Understanding scam victims: Seven principles for systems security. *Communications of the ACM*, 54(3), 70–75. DOI: 10.1145/1897852.1897872

[27] Verizon. (2023). *2023 Data Breach Investigations Report*. Verizon Enterprise Solutions. Retrieved from https://www.verizon.com/business/resources/reports/dbir/

[28] Wang, L., Ma, C., Feng, X., Zhang, Z., Yang, H., Zhang, J., Chen, Z., Tang, J., Chen, X., Lin, Y., Zhao, W. X., Wei, Z., & Wen, J.-R. (2023). A survey on large language model based autonomous agents. *arXiv preprint arXiv:2308.11432*. DOI: 10.48550/arXiv.2308.11432

[29] White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., & Schmidt, D. C. (2023). A prompt pattern catalog to enhance prompt engineering with ChatGPT. *arXiv preprint arXiv:2302.11382*. DOI: 10.48550/arXiv.2302.11382

[30] Workman, M. (2008). Wisecrackers: A theory-grounded investigation of phishing and pretext social engineering threats to information security. *Journal of the American Society for Information Science and Technology*, 59(4), 662–674. DOI: 10.1002/asi.20779

[31] Xi, Z., Chen, W., Guo, X., He, W., Ding, Y., Hong, B., Zhang, M., Wang, J., Jin, S., Zhou, E., Zheng, R., Fan, X., Wang, X., Xiong, L., Zhou, Y., Wang, W., Jiang, C., Zou, Y., Liu, X., Yin, Z., Dou, S., Weng, R., Cheng, W., Zhang, Q., Qin, W., Zheng, Y., Qiu, X., Huang, X., & Gui, T. (2023). The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*. DOI: 10.48550/arXiv.2309.07864

[32] Zimmermann, V., & Renaud, K. (2019). Moving from a 'human-as-problem' to a 'human-as-solution' cybersecurity mindset. *International Journal of Human-Computer Studies*, 131, 169–187. DOI: 10.1016/j.ijhcs.2019.06.005