The Cybersecurity Psychology Framework: From Theoretical Model to Operational Implementation

A Technical Guide for Pilot Deployment

Giuseppe Canale, CISSP
kaolay@gmail.com
Framework Website: https://cpf3.org

September 5, 2025

Abstract

The Cybersecurity Psychology Framework (CPF) provides a systematic approach to predicting security incidents through psychological state assessment. With 85% of breaches involving human factors, traditional technical controls are insufficient. This paper presents a practical implementation guide leveraging modern LLM capabilities for real-time psychological signal processing. We seek collaboration with organizations and research institutions to refine and validate the framework through controlled pilots.

1 Introduction: Why Now?

Three convergent factors make CPF implementation suddenly feasible:

1. **LLM Revolution**: Local language models can now process organizational communications at scale, extracting psychological indicators in real-time.
2. **Data Availability**: Modern organizations generate rich behavioral data through digital communications, authentication logs, and collaboration tools.
3. **Breach Economics**: With average breach costs reaching $4.88M, even 20% reduction justifies significant investment in predictive capabilities.

This is not about proving psychology works in cybersecurity—the evidence is overwhelming. This is about engineering practical systems to capture and act on psychological signals we know exist.

2 Core Implementation Architecture

2.1 Data Pipeline

Listing 1: Core CPF Processing Pipeline

```
import numpy as np
from transformers import AutoModelForSequenceClassification
from datetime import datetime, timedelta
import pandas as pd

class CPFAnalyzer:
    def __init__(self, llm_model='microsoft/deberta-v3-small'):
```

```
        self.llm = AutoModelForSequenceClassification.from_pretrained(llm_model)
        self.indicators = self.initialize_indicators()
        self.baseline = {}

    def analyze_temporal_exhaustion(self, email_df):
        """CPF-Indicator-2.6:-Temporal-Exhaustion-Pattern"""

        # Response time degradation over day
        email_df['hour'] = pd.to_datetime(email_df['timestamp']).dt.hour
        email_df['response_time'] = email_df['response_timestamp'] - email_df['r

        # Calculate degradation curve
        hourly_avg = email_df.groupby('hour')['response_time'].mean()
        degradation = (hourly_avg.iloc[-1] - hourly_avg.iloc[0]) / hourly_avg.il

        # Threshold: >50% degradation = HIGH risk
        if degradation > 0.5:
            return 2  # Red
        elif degradation > 0.2:
            return 1  # Yellow
        return 0  # Green

    def detect_cognitive_overload(self, text_corpus):
        """CPF-Indicator-5.3:-Cognitive-Overload-via-Linguistic-Markers"""

        features = []
        for text in text_corpus:
            # Linguistic complexity
            sentences = text.split('.')
            avg_length = np.mean([len(s.split()) for s in sentences])

            # Cognitive markers
            confusion_markers = ['confused', 'lost', 'overwhelmed', 'too-much']
            marker_density = sum(text.lower().count(m) for m in confusion_marker

            # LLM sentiment
            sentiment = self.llm(text).logits.argmax().item()

            features.append({
                'complexity': avg_length,
                'confusion': marker_density,
                'sentiment': sentiment
            })

        # Statistical anomaly detection
        df = pd.DataFrame(features)
        threshold = df['confusion'].mean() + 2 * df['confusion'].std()

        current = df['confusion'].iloc[-100:].mean()  # Last 100 messages

        return 2 if current > threshold else 0
```

```
    def calculate_convergence_index(self):
        """Critical state when multiple categories align"""

        active_categories = sum(1 for score in self.indicators.values() if score

        if active_categories >= 3:
            # Multiple systems failing = exponential risk
            return np.exp(active_categories / 10)
        return 1.0
```

2.2 Integration Points

The framework integrates with existing infrastructure without disruption:

Listing 2: SIEM Integration Example

```
class CPFSIEMConnector:
    def __init__(self, siem_api):
        self.siem = siem_api
        self.cpf = CPFAnalyzer()

    def process_real_time(self):
        """Process streaming events for psychological indicators"""

        for event in self.siem.stream_events():
            if event.type == 'authentication_failure':
                self.update_stress_indicator(event)

            elif event.type == 'email_metadata':
                exhaustion = self.cpf.analyze_temporal_exhaustion(event.data)
                if exhaustion > 1:
                    self.siem.create_alert({
                        'severity': 'MEDIUM',
                        'title': 'Temporal Exhaustion Detected',
                        'description': f'Team showing {exhaustion} exhaustion le
                        'recommendation': 'Increase phishing detection sensitivi
                    })

            # Convergence check every hour
            if event.timestamp.minute == 0:
                convergence = self.cpf.calculate_convergence_index()
                if convergence > 2:
                    self.trigger_defensive_mode(convergence)
```

3 Measurable Indicators

3.1 Tier 1: Immediately Available (Week 1)

These indicators use data every organization already collects:

Indicator	Data Source	Metric	Threshold
Temporal Exhaustion	Email timestamps	Response degradation	¿50% slower
Alert Fatigue	SIEM acks	Dismissal rate	¿70% ignored
Password Chaos	AD/LDAP	Reset frequency	¿2x baseline
Ticket Velocity	ITSM	Volume spike	¿3σ deviation
Auth Failures	Logs	Failure rate	¿5x normal

3.2 Tier 2: LLM-Enhanced (Week 2-4)

Requiring text analysis via local LLM:

Listing 3: LLM-Based Stress Detection

```
def detect_organizational_stress(slack_export, llm_model):
    """
    Analyze Slack/Teams for stress indicators
    Privacy: No message content stored, only scores
    """

    stress_indicators = {
        'temporal': ['deadline', 'urgent', 'asap', 'now'],
        'emotional': ['frustrated', 'confused', 'worried', 'stressed'],
        'cognitive': ['cant think', 'too much', 'overwhelming'],
        'social': ['nobody helping', 'on my own', 'where is everyone']
    }

    daily_scores = []

    for day_messages in slack_export.group_by_day():
        embeddings = llm_model.encode(day_messages)

        # Aggregate stress without storing content
        stress_score = llm_model.classify_batch(
            embeddings,
            labels=['calm', 'normal', 'stressed', 'crisis']
        )

        daily_scores.append({
            'date': day_messages.date,
            'stress_mean': np.mean(stress_score),
            'stress_std': np.std(stress_score),
            'message_volume': len(day_messages)
        })

    return pd.DataFrame(daily_scores)
```

4 Pilot Protocol

4.1 Phase 1: Baseline (Days 1-30)

1. Deploy collectors (read-only, no PII storage)
2. Establish organizational baseline patterns

3. Train LLM on organization-specific language
4. Identify top 5 relevant CPF indicators

4.2 Phase 2: Correlation (Days 31-60)

1. Daily CPF scoring
2. Correlate with:
    - Security incidents
    - Help desk tickets
    - Error rates
    - Productivity metrics
3. Identify predictive lead time

4.3 Phase 3: Prediction (Days 61-90)

1. Enable predictive alerts
2. Measure:
    - True positive rate
    - False positive rate
    - Lead time accuracy
    - SOC team feedback
3. ROI calculation

5 Privacy and Ethics

Non-negotiable principles:

- No individual profiling - minimum 10-person aggregation
- No message content storage - only statistical scores
- No punitive use - framework identifies systemic issues, not "problem employees"
- Full transparency - teams know what's measured and why
- Opt-out capability - for research/pilot phase

6 Expected Outcomes

6.1 Conservative Estimate

- 48-hour warning for high-risk periods
- 20% reduction in human-factor incidents
- 30% faster incident response

6.2 Optimistic Projection

- 14-day predictive capability
- 50% incident reduction
- Automated defensive adjustments
- 3x ROI within first year

7 Call for Collaboration

We seek three types of partners:

7.1 Enterprise Pilots

Organizations with:

- 500+ employees
- Mature SOC operations
- SIEM/SOAR infrastructure
- Willingness to share anonymized outcomes

7.2 Academic Validation

Universities/institutions for:

- Rigorous statistical validation
- Peer review publication
- Ethical framework development
- Cross-cultural validation

7.3 Technology Partners

SIEM/SOAR vendors for:

- Native integration development
- Scaling to enterprise level
- ML model optimization
- Commercial deployment

8 Implementation Roadmap

9 Conclusion

CPF is not experimental psychology—it's engineering implementation of established psychological principles. The convergence of LLMs, behavioral data availability, and breach economics makes this the right moment for deployment.

We're not asking "does psychology matter in security?" We know it does. We're asking "how can we operationalize this knowledge to prevent breaches?"

Join us in answering that question.

Quarter	Milestone	Deliverable
Q1 2025	5 pilot deployments	Validation data
Q2 2025	Academic publication	Peer-reviewed paper
Q3 2025	V2.0 framework	Refined indicators
Q4 2025	Commercial release	Enterprise product

Contact

- Email: kaolay@gmail.com
- Framework: `https://cpf3.org`
- Pilot Application: `https://cpf3.org/pilot`
- GitHub: `https://github.com/cpf-framework` [Coming Q1 2025]

A Sample CPF Indicators for Quick Reference

1. **Authority Vulnerabilities**: Executive exception patterns
2. **Temporal Vulnerabilities**: End-of-day degradation
3. **Social Influence**: Peer pressure compliance
4. **Affective States**: Stress/fear indicators
5. **Cognitive Overload**: Information processing limits
6. **Group Dynamics**: Collective blind spots
7. **Stress Response**: Fight/flight/freeze patterns
8. **Unconscious Process**: Shadow projections
9. **AI-Specific**: Automation bias
10. **Convergent States**: Perfect storm conditions