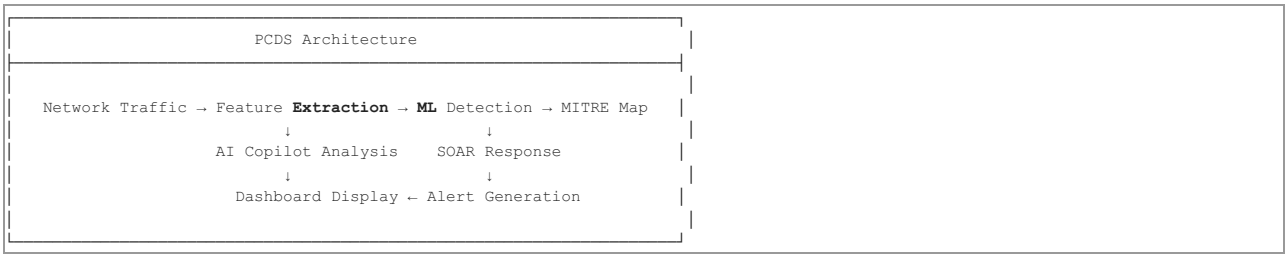
PCDS Core Functions - Step-by-Step Technical Guide

System Overview

PCDS (Predictive Cyber Defense System) is an AI-powered network security platform that detects, analyzes, and responds to cyber threats in real-time.



Core Function 1: Network Traffic Capture

What It Does

Captures and processes network packets flowing through the system.

Step-by-Step Flow

```
1. PACKET ARRIVES
   └── Network interface receives packet

2. PACKET PARSING
   └── Extract: Source IP, Dest IP, Ports, Protocol, Payload size

3. FLOW AGGREGATION
   └── Group packets into "flows" (same src/dst/port combination)

4. FEATURE EXTRACTION
   └── Calculate 40 statistical features per flow
```

Key Features Extracted

Feature	Description	Example
Flow Duration	Time from first to last packet	5.23 seconds
Total Fwd Packets	Packets sent forward	127
Total Bwd Packets	Packets sent backward	89
Flow Bytes/s	Data transfer rate	15,234 bytes/sec
Packet Length Mean	Average packet size	542 bytes
SYN Flag Count	TCP SYN flags	3
FIN Flag Count	TCP FIN flags	2

Code Location

- `backend/ml/feature_extractor.py`
- `backend/ml/real_log_collector.py`

Core Function 2: ML-Based Threat Detection

What It Does

Uses machine learning to classify network traffic as normal or attack.

Step-by-Step Flow

```
1. RECEIVE FEATURES
   └── 40-dimensional feature vector from extraction

2. PREPROCESSING
   └── StandardScaler normalization (zero mean, unit variance)

3. ENSEMBLE PREDICTION
   ├── XGBoost Classifier (40% weight)
   │   └── Returns: probability for each of 15 classes
   ├── Random Forest Classifier (35% weight)
   │   └── Returns: probability for each of 15 classes
   └── Autoencoder (25% weight)
       └── Returns: anomaly score (reconstruction error)

4. WEIGHTED VOTING
   └── Combine all predictions: final_prob = Σ(weight × prob)

5. CLASSIFICATION
   └── Attack class = argmax(final_prob)

6. OUTPUT
   └── {class: "DDoS", confidence: 0.94, anomaly_score: 0.23}
```

Detection Classes

ID	Attack Type	MITRE Technique
0	Normal	-
1-4	DoS Variants	T1498
5	DDoS	T1498
6	PortScan	T1046
7-8	Brute Force	T1110
9	Bot/C2	T1071
10-12	Web Attacks	T1190

Code Location

- backend/ml/ensemble_nids.py
- backend/ml/inference_engine.py

Core Function 3: MITRE ATT&CK Mapping

What It Does

Maps detected attacks to the MITRE ATT&CK framework for standardized threat intelligence.

Step-by-Step Flow

```
1. RECEIVE DETECTION
   └── Attack type: "SQL Injection", confidence: 0.91

2. LOOKUP TECHNIQUE
   └── Query: attack_to_mitre["SQL Injection"]
   └── Result: "T1190 - Exploit Public-Facing Application"

3. GET TECHNIQUE DETAILS
   └── Tactic: Initial Access
   └── Description: "Adversary exploits vulnerability..."
   └── Mitigations: ["M1042", "M1050"]

4. CALCULATE COVERAGE
   └── Total techniques: 191
   └── Covered: 49 (26%)

5. OUTPUT
   └── {technique_id: "T1190", tactic: "Initial Access", ...}
```

Covered Techniques (Examples)

Technique	Name	Tactic
T1046	Network Service Discovery	Discovery
T1110	Brute Force	Credential Access
T1190	Exploit Public-Facing App	Initial Access
T1498	Network Denial of Service	Impact
T1071	Application Layer Protocol	C2

Code Location

- backend/ml/mitre_attack.py
- backend/api/v2/hunt_mitre.py

Core Function 4: Entity Tracking & UEBA

What It Does

Tracks entities (users, hosts, IPs) and detects behavioral anomalies.

Step-by-Step Flow

```
1. ENTITY CREATION
   └── New IP detected: 192.168.1.100
   └── Create entity record with baseline metrics

2. BEHAVIOR PROFILING
   └── Track: login times, data volumes, destinations
   └── Build 30-day behavioral baseline

3. ANOMALY DETECTION
   └── Current: 500MB upload at 3AM
   └── Baseline: avg 50MB, never after midnight
   └── Anomaly score: 0.89 (HIGH)

4. THREAT SCORING
   └── Combine: anomaly score + detection history
   └── Final threat score: 78/100

5. URGENCY CLASSIFICATION
   └── Score 78 → Urgency: HIGH
   └── Trigger: immediate investigation
```

Entity Attributes

Attribute	Description
entity_id	Unique identifier
entity_type	user, host, ip, application
threat_score	0-100 risk score
urgency_level	low, medium, high, critical
first_seen	First detection timestamp
detection_count	Total detections

Code Location

- backend/entity_manager.py
- backend/ml/ueba.py

Core Function 5: AI Copilot (Azure OpenAI)

What It Does

Provides natural language threat analysis using GPT-4.

Step-by-Step Flow

```
1. USER QUERY
   └── "Analyze the SQL injection attack on web-server-01"

2. CONTEXT GATHERING
   └── Fetch: recent detections, entity info, MITRE data
   └── Build context prompt with relevant data

3. API CALL
   └── Azure OpenAI GPT-4 endpoint
   └── System prompt: "You are a security analyst..."
   └── User message: query + context

4. RESPONSE PROCESSING
   └── Parse GPT response
   └── Extract: analysis, recommendations, severity

5. OUTPUT
   └── Formatted analysis with actionable steps
```

Example Interaction

User: "What should I do about the DDoS attack?"

AI Response:

```
## DDoS Attack Analysis

**Immediate Actions:**
1. Enable rate limiting on edge firewall
2. Contact ISP for upstream filtering
3. Activate DDoS mitigation service

**Root Cause:**
- Source: 185.174.x.x/24 (known botnet range)
- Duration: 45 minutes
- Peak: 2.3 Gbps

**MITRE Mapping:** T1498 - Network Denial of Service
```

Code Location

- backend/ml/azure_ai_service.py
- backend/api/v2/copilot.py

Core Function 6: SOAR Orchestration

What It Does

Automates incident response through playbooks.

Step-by-Step Flow

```
1. DETECTION RECEIVED
   └── Attack: Ransomware, severity: CRITICAL

2. INCIDENT CREATION
   └── Generate incident_id: "INC-2024-001234"
   └── Status: NEW

3. PLAYBOOK MATCHING
   └── Query: playbooks.match("Ransomware")
   └── Match: "ransomware_response" playbook

4. PLAYBOOK EXECUTION
   ├── Action 1: Isolate affected host
   ├── Action 2: Block malicious IPs
   ├── Action 3: Create forensic snapshot
   └── Action 4: Alert SOC team

5. STATUS UPDATE
   └── Incident status: CONTAINED
   └── Actions taken: 4

6. DOCUMENTATION
   └── Log all actions with timestamps
```

Available Playbooks

Playbook	Trigger	Actions
ransomware_response	Ransomware detected	Isolate, snapshot, alert
ddos_mitigation	DDoS detected	Rate limit, block, escalate
brute_force_response	Failed logins > 10	Lock account, block IP
data_exfil_response	Large upload detected	Block, investigate, alert

Code Location

- backend/ml/soar_orchestrator.py
- backend/api/v2/soar.py

Core Function 7: Real-Time Dashboard

What It Does

Displays security status and metrics in real-time.

Data Flow

```
1. BACKEND POLLING
   └── API endpoints called every 5 seconds

2. DATA AGGREGATION
   ├── /api/v2/dashboard/overview
   │   └── Total entities, detections, campaigns
   ├── /api/v2/detections
   │   └── Latest security detections
   └── /api/v2/entities
       └── High-risk entities

3. FRONTEND RENDERING
   └── React components update with new data

4. USER INTERACTION
   └── Click detection → View details
   └── Click entity → Investigation view
```

Dashboard Widgets

Widget	Data Source	Refresh
Total Entities	/entities/count	10s
Active Detections	/detections	5s
Risk Score	/dashboard/risk	10s
Threat Distribution	/dashboard/threats	30s
MITRE Coverage	/mitre/stats	60s

Code Location

- `frontend/app/page.tsx`
- `backend/api/v2/dashboard.py`

Core Function 8: Alert Generation

What It Does

Creates and manages security alerts.

Step-by-Step Flow

```
1. DETECTION TRIGGER
   └── Severity >= "high" triggers alert

2. ALERT CREATION
   └── alert_id: "ALT-2024-005678"
   └── title: "Critical DDoS Attack Detected"
   └── severity: CRITICAL

3. ENRICHMENT
   └── Add: entity info, MITRE mapping, timeline

4. NOTIFICATION
   └── WebSocket push to connected clients
   └── Dashboard updates instantly

5. ACKNOWLEDGMENT
   └── Analyst clicks "Acknowledge"
   └── Status: INVESTIGATING

6. RESOLUTION
   └── Analyst resolves alert
   └── Status: CLOSED
```

Alert Priorities

Severity	Response Time	Auto-Escalate
Critical	Immediate	After 5 min
High	< 15 min	After 30 min
Medium	< 1 hour	After 2 hours
Low	< 24 hours	Never

Code Location

- `backend/api/v2/alerts.py`
- `frontend/app/alerts/page.tsx`

Core Function 9: Explainable AI (XAI)

What It Does

Explains WHY the ML model made a prediction.

Step-by-Step Flow

```
1. PREDICTION MADE
   └── Class: DDoS, Confidence: 94%

2. SHAP ANALYSIS
   └── Calculate feature contributions
   └── Which features pushed toward "DDoS"?

3. TOP FEATURES
   ├── flow_bytes_per_sec: +0.35 (high traffic)
   ├── syn_flag_count: +0.28 (SYN flood)
   └── packet_length_mean: +0.15 (small packets)

4. NATURAL LANGUAGE
   └── "Classified as DDoS because of unusually
        high traffic rate (15x normal) with
        excessive SYN flags indicating a flood attack."
```

XAI Output Example

```
{
  "prediction": "DDoS",
  "confidence": 0.94,
  "top_features": [
    {"feature": "flow_bytes_per_sec", "contribution": 0.35, "direction": "positive"},
    {"feature": "syn_flag_count", "contribution": 0.28, "direction": "positive"},
    {"feature": "packet_length_mean", "contribution": 0.15, "direction": "positive"}
  ],
  "explanation": "High traffic volume with SYN flood characteristics"
}
```

Code Location

- backend/ml/explainable_ai.py
- frontend/app/xai/page.tsx

Complete Data Flow Summary

```
┌─────────────────────────────────────────────────────────────────┐
│                    PCDS End-to-End Flow                         │
├─────────────────────────────────────────────────────────────────┤
│                                                                 │
│  1. Network Packet                                              │
│       ↓                                                         │
│  2. Feature Extraction (40 features)                            │
│       ↓                                                         │
│  3. ML Ensemble (XGBoost + RF + Autoencoder)                    │
│       ↓                                                         │
│  4. Classification (15 attack classes)                          │
│       ↓                                                         │
│  5. MITRE Mapping (technique + tactic)                          │
│       ↓                                                         │
│  6. Entity Update (threat score)                                │
│       ↓                                                         │
│  7. Alert Generation (if severity >= high)                      │
│       ↓                                                         │
│  8. SOAR Playbook (automated response)                          │
│       ↓                                                         │
│  9. Dashboard Update (real-time WebSocket)                      │
│       ↓                                                         │
│  10. AI Copilot (on-demand analysis)                            │
│                                                                 │
└─────────────────────────────────────────────────────────────────┘
```

API Endpoints Reference

Endpoint	Method	Purpose
/api/v2/dashboard/overview	GET	Dashboard stats
/api/v2/detections	GET	List detections
/api/v2/entities	GET	List entities
/api/v2/alerts	GET	List alerts
/api/v2/mitre/stats	GET	MITRE coverage
/api/v2/soar/simulate	POST	Trigger attack sim
/api/v2/copilot/query	POST	AI analysis
/api/v2/xai/explain	POST	Get explanation

Technology Stack

Layer	Technology
Frontend	Next.js 14, React, TypeScript, Tailwind
Backend	Python 3.11, FastAPI, SQLite
ML	PyTorch, XGBoost, Scikit-learn

AI Azure OpenAI GPT-4
Real-time WebSocket
