Contents

C2/C3 Attack Monitoring System **5**
Comprehensive User Guide 5

Table of Contents **6**

1. Introduction **7**
1.1 Purpose 7
1.2 Key Features 7
1.3 Target Audience 7

2. System Overview **8**
2.1 Architecture 8
2.2 Component Description 8
Frontend (Next.js 14) 8
Backend (FastAPI) 8
Detection Engine 9
Machine Learning Stack 9
Data Storage 9
2.3 Data Flow 9

3. Getting Started **10**
3.1 Accessing the Dashboard 10
3.2 Interface Overview 10
3.3 Connection Status 10
3.4 Initial Configuration 10

4. Dashboard Components **11**
4.1 Statistics Panel 11
Total Alerts 11
Critical Alerts 11
Active Threats 11
Detection Rate 11
4.2 Live Alerts Feed 11
Alert Card Information 11
Interactions 12
4.3 Threat Timeline 12
Time Range Selection 12
Chart Features 12
Interpreting the Timeline 12
4.4 Network Topology 12
Node Types 12
Node Size 12

Link Visualization . . . 12
Controls . . . 13
Interactions . . . 13
4.5 Alert Severity Chart . . . 13
4.6 Alerts Table . . . 13
Columns . . . 13
Features . . . 13

5. Detection Capabilities **15**
5.1 Beaconing Detection . . . 15
What It Detects . . . 15
Detection Methods . . . 15
Configuration Parameters . . . 15
Alert Information . . . 15
5.2 DNS Tunneling Detection . . . 16
What It Detects . . . 16
Detection Methods . . . 16
Configuration Parameters . . . 16
Alert Information . . . 16
5.3 Data Exfiltration Detection . . . 16
What It Detects . . . 16
Detection Methods . . . 16
Configuration Parameters . . . 17
Alert Information . . . 17
5.4 Encrypted C2 Detection . . . 17
What It Detects . . . 17
Detection Methods . . . 17
Configuration Parameters . . . 18
Alert Information . . . 18
5.5 DGA Domain Detection . . . 18
What It Detects . . . 18
Detection Methods . . . 18
Configuration Parameters . . . 18
Alert Information . . . 19
5.6 Machine Learning Detection . . . 19
What It Detects . . . 19
Algorithms Used . . . 19
Ensemble Scoring . . . 19
Features Analyzed . . . 19
Configuration Parameters . . . 20

6. Alert Management **21**
6.1 Alert Lifecycle . . . 21
Generation . . . 21
States . . . 21
6.2 Severity Levels . . . 21
Critical (Severity 4) . . . 21
High (Severity 3) . . . 21
Medium (Severity 2) . . . 22
Low (Severity 1) . . . 22
6.3 MITRE ATT&CK Mapping . . . 22
6.4 Investigation Workflow . . . 22
Step 1: Initial Triage . . . 22
Step 2: Context Gathering . . . 22

Step 3: Analysis . . . 22
Step 4: Response . . . 23
Step 5: Resolution . . . 23

7. Network Topology Visualization **24**
7.1 Understanding the Graph . . . 24
Reading the Visualization . . . 24
Use Cases . . . 24
7.2 Interactive Features . . . 25
Navigation . . . 25
Node Interaction . . . 25
Filtering . . . 25
7.3 Best Practices . . . 25
Regular Review . . . 25
During Incidents . . . 25
Documentation . . . 25

8. API Reference **26**
8.1 Base URL . . . 26
8.2 Authentication . . . 26
8.3 Endpoints . . . 26
Health Check . . . 26
Alerts . . . 26
Statistics . . . 28
Topology . . . 28
WebSocket . . . 29
8.4 API Documentation . . . 29

9. Configuration **30**
9.1 Configuration File . . . 30
9.2 Detection Thresholds . . . 30
Beaconing Detection . . . 30
DNS Tunneling Detection . . . 30
Exfiltration Detection . . . 30
Encrypted C2 Detection . . . 31
DGA Detection . . . 31
9.3 Machine Learning Configuration . . . 31
9.4 Integration Configuration . . . 31
Zeek Integration . . . 31
Suricata Integration . . . 32
9.5 Redis Configuration . . . 32
9.6 Database Configuration . . . 32
9.7 Applying Configuration Changes . . . 32

10. Troubleshooting **33**
10.1 Connection Issues . . . 33
WebSocket Disconnection . . . 33
API Not Responding . . . 33
10.2 Detection Issues . . . 33
No Alerts Generated . . . 33
Too Many False Positives . . . 34
10.3 Performance Issues . . . 34
Slow Dashboard Loading . . . 34
High Memory Usage . . . 34
10.4 Common Error Messages . . . 34

"Database connection failed" . . . 34
"Redis connection refused" . . . 34
"Detection engine initialization failed" . . . 35
10.5 Log Locations . . . 35

11. Security Considerations **36**
11.1 Access Control . . . 36
Current State . . . 36
Recommended Improvements . . . 36
11.2 Network Security . . . 36
TLS Configuration . . . 36
Security Headers . . . 36
11.3 Data Protection . . . 36
Sensitive Data . . . 36
Recommendations . . . 36
11.4 Operational Security . . . 37
Container Security . . . 37
Host Security . . . 37

Appendix **38**
A. Keyboard Shortcuts . . . 38
B. Glossary . . . 38
C. File Locations . . . 38
D. Docker Commands Reference . . . 39
E. Support . . . 39

C2/C3 Attack Monitoring System

Comprehensive User Guide

Table of Contents

1. Introduction
2. System Overview
3. Getting Started
4. Dashboard Components
5. Detection Capabilities
6. Alert Management
7. Network Topology Visualization
8. API Reference
9. Configuration
10. Troubleshooting
11. Security Considerations
12. Appendix

---

1. Introduction

1.1 Purpose

The C2/C3 Attack Monitoring System is a comprehensive, real-time security monitoring platform designed to detect Command and Control (C2) communication patterns and other malicious network activities. This system combines advanced statistical analysis, machine learning algorithms, and integration with industry-standard security tools to provide enterprise-grade threat detection capabilities.

1.2 Key Features

- **Real-Time Detection**: Continuous monitoring of network traffic with sub-second alert generation
- **Multi-Vector Analysis**: Detection of beaconing, DNS tunneling, data exfiltration, encrypted C2, and DGA domains
- **Machine Learning Integration**: Ensemble detection using Isolation Forest, Autoencoder, and Local Outlier Factor algorithms
- **Interactive Visualization**: D3.js-powered network topology graphs and Recharts timeline visualization
- **Security Tool Integration**: Native support for Zeek logs, Suricata EVE JSON, and packet capture
- **WebSocket Communication**: Real-time bidirectional updates with subscription-based filtering
- **RESTful API**: Comprehensive API with auto-generated Swagger documentation

1.3 Target Audience

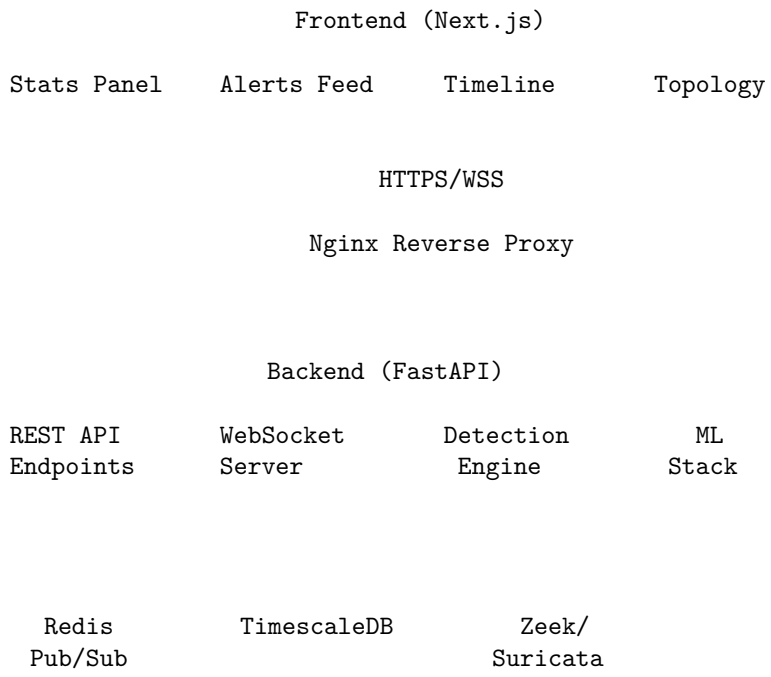
This guide is intended for: - Security Operations Center (SOC) analysts - Incident response teams - Network security engineers - System administrators - Threat hunters

---

2. System Overview

2.1 Architecture

The C2 Monitor system consists of several integrated components:



2.2 Component Description

Frontend (Next.js 14)

- Static export served directly by Nginx
- React-based single-page application
- Real-time updates via WebSocket
- Responsive design for various screen sizes

Backend (FastAPI)

- Asynchronous Python web framework
- RESTful API with automatic OpenAPI documentation

- WebSocket server for real-time communication
- Detection engine orchestration

Detection Engine

- Five specialized detectors for different attack vectors
- Fortran-accelerated numerical computations (with Python fallbacks)
- Configurable thresholds and sensitivity

Machine Learning Stack

- Ensemble detector combining multiple algorithms
- Adaptive training on network baselines
- Model persistence and versioning

Data Storage

- **Redis**: Real-time pub/sub, caching, and session management
- **TimescaleDB**: Time-series optimized PostgreSQL for historical data

2.3 Data Flow

1. Network data ingested from Zeek, Suricata, or packet capture
2. Detection engine analyzes traffic patterns
3. ML models score anomalous behavior
4. Alerts generated and stored in database
5. WebSocket broadcasts alerts to connected clients
6. Dashboard updates in real-time

---

3. Getting Started

3.1 Accessing the Dashboard

Open your web browser and navigate to:

`https://dude.pmarines.org`

The dashboard loads automatically and establishes a WebSocket connection for real-time updates.

3.2 Interface Overview

Upon loading, you will see the main dashboard with the following sections:

1. **Header Bar**: System title, connection status indicator
2. **Statistics Panel**: Four cards showing key metrics
3. **Live Alerts Feed**: Real-time alert stream
4. **Threat Timeline**: Historical visualization of alerts
5. **Network Topology**: Interactive graph of network connections
6. **Alert Severity Chart**: Distribution of alerts by severity
7. **Alerts Table**: Detailed, searchable list of all alerts

3.3 Connection Status

The header displays the current connection status:

- **Connected** (Green): WebSocket connection active, receiving real-time updates
- **Disconnected** (Red): No connection to backend, data may be stale
- **Reconnecting** (Yellow): Attempting to re-establish connection

If disconnected, the system automatically attempts reconnection every 5 seconds.

3.4 Initial Configuration

For first-time setup, ensure:

1. Backend services are running (check Docker container status)
2. Network data sources (Zeek/Suricata) are configured
3. Detection thresholds are appropriate for your environment

---

4. Dashboard Components

4.1 Statistics Panel

The statistics panel displays four key metrics in card format:

Total Alerts

- Count of all alerts generated
- Color-coded by trend (increasing = red, stable = gray)

Critical Alerts

- Number of critical severity alerts requiring immediate attention
- Prominent red highlighting when active

Active Threats

- Currently ongoing suspicious activities
- Threats that haven’t been resolved or expired

Detection Rate

- Percentage of traffic analyzed that triggered alerts
- Helps gauge overall network health

4.2 Live Alerts Feed

The left sidebar shows a real-time stream of incoming alerts:

Alert Card Information

- **Severity Badge**: Critical (red), High (orange), Medium (yellow), Low (blue)
- **Alert Type**: Category of detection (Beaconing, DNS Tunneling, etc.)
- **Source/Destination**: IP addresses involved
- **Timestamp**: When the alert was generated
- **Confidence Score**: ML model confidence (0-100%)

Interactions

- Click an alert to view full details
- Alerts auto-scroll as new ones arrive
- Maximum of 50 alerts displayed (older ones archived)

4.3 Threat Timeline

The central chart visualizes alert frequency over time:

Time Range Selection

Use the dropdown to select: - Last 6 hours - Last 12 hours - Last 24 hours (default) - Last 48 hours - Last 7 days

Chart Features

- **Area Chart**: Stacked areas by severity
- **Hover Tooltips**: Exact counts at each time point
- **Trend Lines**: Visual indication of attack patterns

Interpreting the Timeline

- Spikes indicate potential attack campaigns
- Sustained elevation suggests persistent threats
- Periodic patterns may indicate beaconing behavior

4.4 Network Topology

Interactive D3.js force-directed graph showing network relationships:

Node Types

- **Blue**: Internal hosts
- **Orange**: External hosts
- **Purple**: DNS servers
- **Red**: Malicious/suspicious hosts

Node Size

Proportional to alert count - larger nodes have more associated alerts.

Link Visualization

- **Gray lines**: Normal connections
- **Red lines**: Suspicious connections
- **Line thickness**: Connection frequency/volume

Controls

- **Zoom In/Out**: Mouse wheel or buttons
- **Pan**: Click and drag background
- **Move Nodes**: Click and drag individual nodes
- **Reset View**: Button to restore default zoom/position
- **Suspicious Only**: Toggle to filter view

Interactions

- Hover over nodes to see IP address, alert count, and connection count
- Drag nodes to rearrange the graph
- Double-click to isolate a node's connections

4.5 Alert Severity Chart

Pie chart showing distribution of alerts by severity:

- **Critical**: Immediate action required
- **High**: Urgent investigation needed
- **Medium**: Should be reviewed
- **Low**: Informational, monitor for patterns

4.6 Alerts Table

Comprehensive table with full alert details:

Columns

- **Time**: Alert timestamp
- **Type**: Detection category
- **Severity**: Alert severity level
- **Source IP**: Origin of suspicious traffic
- **Destination IP**: Target of suspicious traffic
- **Description**: Detailed alert message
- **Confidence**: ML confidence score

Features

Searching

Type in the search box to filter alerts by any field content.

Filtering

- **Severity Filter**: Show only specific severity levels
- **Type Filter**: Show only specific detection types

Sorting

Click column headers to sort ascending/descending.

Pagination

Navigate through large result sets with page controls.

Export

Click “Export CSV” to download all filtered alerts for external analysis.

---

5. Detection Capabilities

5.1 Beaconing Detection

What It Detects

Periodic communication patterns typical of malware calling home to C2 servers.

Detection Methods

Interval Analysis

- Calculates time intervals between connections
- Computes coefficient of variation (CV)
- Low CV indicates regular, automated beaconing

Statistical Measures

- **Mean Interval**: Average time between connections
- **Standard Deviation**: Variance in intervals
- **Bowley Skewness**: Asymmetry of interval distribution

FFT Periodicity

- Fast Fourier Transform analysis
- Identifies dominant frequencies in connection patterns
- Detects even sophisticated jittered beacons

Configuration Parameters

```yaml
beaconing:
  min_connections: 10       # Minimum connections to analyze
  cv_threshold: 0.3         # Max coefficient of variation
  periodicity_threshold: 0.7  # FFT periodicity score
  time_window: 3600         # Analysis window (seconds)
```

Alert Information

- Detected interval pattern
- Periodicity score
- Connection count and timespan
- Destination IP and port

5.2 DNS Tunneling Detection

What It Detects

Covert data transfer through DNS queries, often used to bypass firewalls.

Detection Methods

Entropy Analysis

- Shannon entropy of subdomain strings
- High entropy indicates encoded/encrypted data
- Normal domains have lower entropy

Query Characteristics

- Subdomain length analysis
- Query frequency per domain
- TXT record abuse detection

Behavioral Patterns

- Unusual query volumes
- Non-standard record types
- Direct IP DNS servers

Configuration Parameters

```yaml
dns_tunneling:
  entropy_threshold: 3.5   # Shannon entropy threshold
  length_threshold: 50     # Subdomain length threshold
  frequency_threshold: 100 # Queries per minute threshold
  txt_record_threshold: 10 # TXT queries threshold
```

Alert Information

- Domain queried
- Entropy score
- Query type and frequency
- Subdomain characteristics

5.3 Data Exfiltration Detection

What It Detects

Large or unusual data transfers that may indicate data theft.

Detection Methods

Volume Analysis

- Monitors outbound data volumes

- Compares against baseline patterns
- Detects sudden large transfers

Ratio Analysis

- Upload/download ratio per host
- Identifies hosts sending more than receiving
- Flags unusual upload patterns

Timing Analysis

- After-hours transfer detection
- Burst transfer patterns
- Sustained high-volume transfers

Configuration Parameters

```yaml
exfiltration:
  volume_threshold: 104857600  # 100MB threshold
  ratio_threshold: 5.0         # Upload/download ratio
  time_window: 3600            # Analysis window
  baseline_multiplier: 3.0     # Times baseline threshold
```

Alert Information

- Data volume transferred
- Transfer duration
- Source and destination
- Baseline comparison

5.4 Encrypted C2 Detection

What It Detects

Malicious encrypted communications using TLS/SSL fingerprinting.

Detection Methods

JA3/JA3S Fingerprinting

- TLS client fingerprint (JA3)
- TLS server fingerprint (JA3S)
- Comparison against known malware fingerprints

Certificate Analysis

- Self-signed certificate detection
- Certificate validity periods
- Issuer reputation checking

Connection Patterns

- Unusual port usage
- Certificate/SNI mismatches
- Expired certificate usage

Configuration Parameters

```yaml
encrypted_c2:
  ja3_blacklist_path: /opt/c2monitor/data/ja3_blacklist.txt
  check_self_signed: true
  check_expired: true
  min_validity_days: 7
```

Alert Information

- JA3/JA3S fingerprint
- Certificate details
- Match reason (blacklist, self-signed, etc.)
- Connection metadata

5.5 DGA Domain Detection

What It Detects

Domain Generation Algorithm (DGA) domains used by malware for C2 resilience.

Detection Methods

Linguistic Analysis

- Consonant/vowel ratio
- N-gram frequency analysis
- Dictionary word matching

Statistical Features

- Character distribution entropy
- Domain length analysis
- Numeric character ratio

Pattern Recognition

- Known DGA family patterns
- Randomness scoring
- TLD analysis

Configuration Parameters

```yaml
dga:
  entropy_threshold: 3.0
  length_threshold: 15
```

```
consonant_ratio_threshold: 0.7
ngram_threshold: 0.5
```

Alert Information

- Suspicious domain
- DGA probability score
- Linguistic features
- Associated IP addresses

5.6 Machine Learning Detection

What It Detects

Anomalous patterns not matching specific signatures, using ensemble ML.

Algorithms Used

Isolation Forest

- Unsupervised anomaly detection
- Identifies outliers in feature space
- Effective for high-dimensional data

Autoencoder

- Neural network reconstruction
- High reconstruction error = anomaly
- Learns normal traffic patterns

Local Outlier Factor (LOF)

- Density-based detection
- Compares local density to neighbors
- Identifies local anomalies

Ensemble Scoring

```
Final Score = (w1 × IF_score + w2 × AE_score + w3 × LOF_score) / (w1 + w2 + w3)
```

Default weights: IF=0.4, AE=0.35, LOF=0.25

Features Analyzed

- Connection duration
- Bytes transferred (in/out)
- Packet counts
- Protocol distribution
- Port entropy
- Time-based features

Configuration Parameters

```yaml
ml:
  isolation_forest:
    contamination: 0.1
    n_estimators: 100
  autoencoder:
    encoding_dim: 16
    threshold_percentile: 95
  lof:
    n_neighbors: 20
    contamination: 0.1
  ensemble:
    threshold: 0.7
    weights:
      isolation_forest: 0.4
      autoencoder: 0.35
      lof: 0.25
```

---

6. Alert Management

6.1 Alert Lifecycle

Generation

1. Detection engine identifies suspicious activity
2. Alert created with severity, type, and details
3. Stored in TimescaleDB
4. Published via Redis pub/sub
5. WebSocket broadcasts to clients

States

- **New**: Just generated, not yet reviewed
- **Acknowledged**: Analyst has seen the alert
- **Investigating**: Under active investigation
- **Resolved**: Issue addressed or false positive
- **Escalated**: Sent to higher tier or incident response

6.2 Severity Levels

Critical (Severity 4)

- **Description**: Confirmed active threat requiring immediate response
- **Examples**:
  - Active data exfiltration
  - Known malware C2 communication
  - Ransomware beaconing
- **Response Time**: Immediate (< 15 minutes)

High (Severity 3)

- **Description**: Likely threat requiring urgent investigation
- **Examples**:
  - High-confidence beaconing detection
  - DNS tunneling with data transfer
  - Blacklisted JA3 fingerprint
- **Response Time**: Urgent (< 1 hour)

Medium (Severity 2)

- **Description**: Suspicious activity requiring review
- **Examples**:
  - Moderate ML anomaly scores
  - Potential DGA domains
  - Unusual certificate usage
- **Response Time**: Standard (< 4 hours)

Low (Severity 1)

- **Description**: Informational, may indicate policy violations
- **Examples**:
  - Low-confidence detections
  - Single anomalous connection
  - Borderline threshold violations
- **Response Time**: Best effort (< 24 hours)

6.3 MITRE ATT&CK Mapping

Alerts are mapped to MITRE ATT&CK techniques:

Detection Type	Technique ID	Technique Name
Beaconing	T1071	Application Layer Protocol
DNS Tunneling	T1071.004	DNS
Exfiltration	T1041	Exfiltration Over C2 Channel
Encrypted C2	T1573	Encrypted Channel
DGA	T1568.002	Domain Generation Algorithms

6.4 Investigation Workflow

Step 1: Initial Triage

1. Review alert details in the alerts table
2. Check severity and confidence score
3. Examine source and destination IPs
4. Look for related alerts (same IPs, timeframe)

Step 2: Context Gathering

1. Check network topology for host relationships
2. Review timeline for related activity
3. Query historical data via API
4. Cross-reference with threat intelligence

Step 3: Analysis

1. Examine raw network data if available
2. Check endpoint logs for corroboration
3. Analyze any captured payloads

4. Determine scope of potential compromise

Step 4: Response

1. Isolate affected hosts if necessary
2. Block malicious IPs/domains
3. Collect forensic evidence
4. Document findings

Step 5: Resolution

1. Mark alert as resolved
2. Update detection rules if needed
3. Create incident report
4. Implement preventive measures

---

7. Network Topology Visualization

7.1 Understanding the Graph

The network topology provides a visual representation of network relationships and suspicious connections.

Reading the Visualization

Node Interpretation

- **Position**: Automatically calculated by force-directed algorithm
- **Color**: Indicates node type (internal, external, suspicious)
- **Size**: Proportional to alert count
- **Border**: Red border indicates suspicious activity

Edge Interpretation

- **Color**: Gray (normal) or red (suspicious)
- **Thickness**: Indicates connection frequency/volume
- **Direction**: Lines connect source to destination

Use Cases

Identifying C2 Infrastructure

Look for: - Single external node connected to many internal hosts - Star patterns with suspicious central node - Unusual port concentrations

Finding Lateral Movement

Look for: - Chains of internal-to-internal connections - Hosts with unusual peer connections - New connections from compromised hosts

Detecting Data Staging

Look for: - Large data flows to single internal host - Internal host with many connections before external transfer - Unusual time-of-day activity patterns

7.2 Interactive Features

Navigation

- **Zoom**: Mouse wheel or +/- buttons
- **Pan**: Click and drag empty space
- **Reset**: Click reset button to restore default view

Node Interaction

- **Hover**: Display tooltip with node details
- **Click**: Select node for detailed view
- **Drag**: Reposition node (other nodes adjust)

Filtering

- **Suspicious Only**: Toggle to show only nodes with alerts
- **Time Range**: Filter by activity timeframe
- **Alert Type**: Filter by specific detection types

7.3 Best Practices

Regular Review

- Check topology daily for new patterns
- Compare against baseline network map
- Investigate new external connections

During Incidents

- Filter to suspicious only
- Focus on affected timeframe
- Track lateral movement paths

Documentation

- Export screenshots for reports
- Note unusual patterns
- Track topology changes over time

---

8. API Reference

8.1 Base URL

`https://dude.pmarines.org/api`

8.2 Authentication

Currently, the API does not require authentication. In production, implement appropriate authentication mechanisms.

8.3 Endpoints

Health Check

`GET /api/health`

Response:

```
{
  "status": "healthy",
  "timestamp": "2026-01-10T14:30:00.000Z",
  "version": "1.0.0",
  "services": {
    "api": "running",
    "database": "connected",
    "redis": "connected"
  }
}
```

Alerts

List Alerts

`GET /api/alerts`

Query Parameters: | Parameter | Type | Description | |———–|——|———–| | limit | int | Max results (default: 100) | | offset | int | Pagination offset | | severity | string | Filter by severity | | type | string | Filter by alert type | | start_time | datetime | Start of time range | | end_time | datetime | End of time range | | source_ip | string | Filter by source IP | | dest_ip | string | Filter by destination IP |

Response:

```
{
  "alerts": [
    {
      "id": "alert_123",
      "timestamp": "2026-01-10T14:30:00.000Z",
      "type": "beaconing",
      "severity": "high",
      "source_ip": "192.168.1.100",
      "dest_ip": "45.33.32.156",
      "description": "Periodic beaconing detected",
      "confidence": 0.87,
      "details": {
        "interval": 300,
        "cv": 0.15,
        "periodicity": 0.92
      },
      "mitre_technique": "T1071"
    }
  ],
  "total": 150,
  "limit": 100,
  "offset": 0
}
```


Get Alert by ID

```
GET /api/alerts/{alert_id}
```

Get Alert Summary

```
GET /api/alerts/summary
```

Response:


```
{
  "total": 1523,
  "by_severity": {
    "critical": 12,
    "high": 89,
    "medium": 456,
    "low": 966
  },
  "by_type": {
    "beaconing": 234,
    "dns_tunneling": 89,
    "exfiltration": 45,
    "encrypted_c2": 123,
    "dga": 67,
    "ml_detection": 965
  }
}
```


Get Alert Timeline

```
GET /api/alerts/timeline
```

Query Parameters: | Parameter | Type | Description | |———–|——|———–| | hours | int | Hours to look back (default: 24) | | interval | string | Grouping interval (hour, day) |

Statistics

System Statistics

```
GET /api/statistics
```

Response:

```
{
  "alerts": {
    "total": 1523,
    "last_24h": 234,
    "critical_active": 3
  },
  "traffic": {
    "bytes_analyzed": 1073741824,
    "flows_analyzed": 45678,
    "packets_analyzed": 987654
  },
  "detection": {
    "rate": 0.034,
    "false_positive_rate": 0.12
  }
}
```

Detector Statistics

```
GET /api/statistics/detectors
```

Topology

Get Network Topology

```
GET /api/topology
```

Query Parameters: | Parameter | Type | Description | |———–|——|———–| | max_nodes | int | Maximum nodes (default: 100) | | suspicious_only | bool | Only suspicious nodes | | time_range | int | Hours to look back |

Response:

```
{
  "nodes": [
    {
      "id": "192.168.1.100",
      "ip": "192.168.1.100",
      "label": "workstation-1",
      "type": "internal",
      "alert_count": 5,
      "connections": 12,
      "is_suspicious": true
    }
  ],
```

```
  "links": [
    {
      "source": "192.168.1.100",
      "target": "45.33.32.156",
      "value": 23,
      "is_suspicious": true
    }
  ]
}
```

WebSocket

Connect

```
const ws = new WebSocket('wss://dude.pmarines.org/ws');
```

Subscribe to Alerts

```
{
  "action": "subscribe",
  "channel": "alerts",
  "filters": {
    "severity": ["critical", "high"],
    "types": ["beaconing", "exfiltration"]
  }
}
```

Receive Alerts

```
{
  "type": "alert",
  "data": {
    "id": "alert_124",
    "timestamp": "2026-01-10T14:31:00.000Z",
    "type": "beaconing",
    "severity": "high",
    "source_ip": "192.168.1.100",
    "dest_ip": "45.33.32.156"
  }
}
```

8.4 API Documentation

Full interactive API documentation is available at:

`https://dude.pmarines.org/api/docs`

This Swagger UI allows you to: - Browse all endpoints - View request/response schemas - Test API calls directly

---

9. Configuration

9.1 Configuration File

The main configuration file is located at:

`/var/www/nextjs/dude/config/settings.yaml`

9.2 Detection Thresholds

Beaconing Detection

```yaml
detection:
  beaconing:
    enabled: true
    min_connections: 10
    cv_threshold: 0.3
    periodicity_threshold: 0.7
    time_window: 3600
    min_interval: 10
    max_interval: 86400
```

DNS Tunneling Detection

```yaml
detection:
  dns_tunneling:
    enabled: true
    entropy_threshold: 3.5
    length_threshold: 50
    frequency_threshold: 100
    txt_record_threshold: 10
```

Exfiltration Detection

```yaml
detection:
  exfiltration:
    enabled: true
    volume_threshold: 104857600
    ratio_threshold: 5.0
    time_window: 3600
    baseline_multiplier: 3.0
```

Encrypted C2 Detection

```yaml
detection:
  encrypted_c2:
    enabled: true
    ja3_blacklist_path: /opt/c2monitor/data/ja3_blacklist.txt
    check_self_signed: true
    check_expired: true
```

DGA Detection

```yaml
detection:
  dga:
    enabled: true
    entropy_threshold: 3.0
    length_threshold: 15
    consonant_ratio_threshold: 0.7
```

9.3 Machine Learning Configuration

```yaml
ml:
  enabled: true
  model_path: /opt/c2monitor/models

  isolation_forest:
    contamination: 0.1
    n_estimators: 100
    max_samples: auto

  autoencoder:
    encoding_dim: 16
    hidden_layers: [64, 32]
    epochs: 50
    threshold_percentile: 95

  lof:
    n_neighbors: 20
    contamination: 0.1

  ensemble:
    threshold: 0.7
    weights:
      isolation_forest: 0.4
      autoencoder: 0.35
      lof: 0.25
```

9.4 Integration Configuration

Zeek Integration

```yaml
integrations:
  zeek:
```

```
    enabled: true
    log_path: /var/log/zeek
    watch_files:
      - conn.log
      - dns.log
      - ssl.log
      - http.log
```

Suricata Integration

```
integrations:
  suricata:
    enabled: true
    eve_path: /var/log/suricata/eve.json
```

9.5 Redis Configuration

```
redis:
  host: 127.0.0.1
  port: 6379
  db: 0
  password: null
  channels:
    alerts: c2monitor:alerts
    stats: c2monitor:stats
```

9.6 Database Configuration

```
database:
  host: 127.0.0.1
  port: 5433
  name: c2monitor
  user: c2monitor
  password: ${DB_PASSWORD}
  pool_size: 10
```

9.7 Applying Configuration Changes

After modifying configuration:

1. Validate syntax:

   ```
   python -c "import yaml; yaml.safe_load(open('/var/www/nextjs/dude/config/settings.yaml'))"
   ```

2. Restart backend:

   ```
   cd /opt/docker-apps/c2monitor
   docker compose restart backend
   ```

3. Verify changes applied:

   ```
   docker compose logs backend --tail=20
   ```

---

10. Troubleshooting

10.1 Connection Issues

WebSocket Disconnection

Symptoms: - “Disconnected” status in header - No real-time updates - Stale alert data

Solutions: 1. Check backend status: `bash docker compose ps`

2. Verify WebSocket endpoint:

```
curl -i -N -H "Connection: Upgrade" -H "Upgrade: websocket" \
  https://dude.pmarines.org/ws
```

3. Check Nginx logs:

```
sudo tail -f /var/log/nginx/dude.error.log
```

4. Restart backend:

```
docker compose restart backend
```

API Not Responding

Symptoms: - Empty dashboard components - API errors in browser console - Timeout errors

Solutions: 1. Check backend health: `bash curl https://dude.pmarines.org/api/health`

2. Check backend logs:

```
docker compose logs backend --tail=50
```

3. Verify database connection:

```
docker compose exec timescaledb pg_isready
```

10.2 Detection Issues

No Alerts Generated

Symptoms: - Alert count stays at zero - Empty alerts table - Flat timeline

Possible Causes: 1. No network data being ingested 2. Detection thresholds too high 3. Integration not configured

Solutions: 1. Verify data ingestion: `bash docker compose logs backend | grep -i "processing"`

2. Check integration status:

```
curl https://dude.pmarines.org/api/health | jq '.services'
```

3. Lower detection thresholds temporarily for testing

Too Many False Positives

Symptoms: - Alert flood - Low confidence scores - Benign traffic flagged

Solutions: 1. Raise detection thresholds in configuration 2. Add known-good hosts to whitelist 3. Retrain ML models with cleaner baseline 4. Adjust severity weights

10.3 Performance Issues

Slow Dashboard Loading

Solutions: 1. Reduce max_nodes in topology: `yaml topology: max_nodes: 50`

2. Limit timeline range to 6-12 hours
3. Use pagination in alerts table

High Memory Usage

Solutions: 1. Limit Redis cache size: `bash docker compose exec redis redis-cli CONFIG SET maxmemory 256mb`

2. Reduce backend workers:

```
# In docker-compose.yml
command: uvicorn backend.api.main:app --workers 2
```

3. Implement data retention policies

10.4 Common Error Messages

“Database connection failed”

```
# Check TimescaleDB status
docker compose ps timescaledb
docker compose logs timescaledb --tail=20

# Restart database
docker compose restart timescaledb
```

“Redis connection refused”

```
# Check Redis status
docker compose exec redis redis-cli ping

# Restart Redis
docker compose restart redis
```

“Detection engine initialization failed”

```
# Check for missing dependencies
docker compose exec backend pip list | grep -E "scipy|numpy|sklearn"

# Rebuild backend image
docker compose build backend
docker compose up -d backend
```

10.5 Log Locations

Component	Log Location
Nginx	/var/log/nginx/dude.access.log, /var/log/nginx/dude.error.log
Backend	docker compose logs backend
Redis	docker compose logs redis
TimescaleDB	docker compose logs timescaledb

---

11. Security Considerations

11.1 Access Control

Current State

The system currently does not implement authentication. This is suitable for: - Internal network deployment - Lab/testing environments - VPN-protected access

Recommended Improvements

1. Implement API key authentication
2. Add OAuth2/OIDC integration
3. Role-based access control (RBAC)
4. IP whitelisting

11.2 Network Security

TLS Configuration

The system uses TLS 1.2/1.3 with strong cipher suites: - ECDHE key exchange - AES-GCM encryption - Perfect forward secrecy

Security Headers

Nginx adds security headers: - Strict-Transport-Security (HSTS) - X-Frame-Options - X-Content-Type-Options - X-XSS-Protection - Referrer-Policy

11.3 Data Protection

Sensitive Data

The system processes: - IP addresses (source and destination) - DNS queries - Connection metadata - TLS fingerprints

Recommendations

1. Implement data retention policies
2. Anonymize IP addresses for long-term storage
3. Encrypt database at rest
4. Regular backup procedures

11.4 Operational Security

Container Security

- Run containers as non-root
- Use read-only file systems where possible
- Limit container capabilities
- Regular image updates

Host Security

- Keep system packages updated
- Monitor Docker daemon
- Implement host-based firewall
- Regular security audits

Appendix

A. Keyboard Shortcuts

Shortcut	Action
R	Refresh data
F	Focus search box
Esc	Clear filters
T	Toggle topology view
?	Show help

B. Glossary

Term	Definition
Beaconing	Regular, periodic network communication typical of malware C2
C2/C&C	Command and Control - infrastructure used to control malware
DGA	Domain Generation Algorithm - creates random domain names for C2
DNS Tunneling	Using DNS queries to exfiltrate data or establish C2
JA3	TLS client fingerprinting method
MITRE ATT&CK	Knowledge base of adversary tactics and techniques

C. File Locations

File/Directory	Purpose
/var/www/nextjs/dude/frontend/out	Static frontend files
/var/www/nextjs/dude/backend	Backend Python code
/var/www/nextjs/dude/config	Configuration files
/opt/docker-apps/c2monitor	Docker Compose files
/opt/c2monitor/models	ML model storage
/var/log/c2monitor	Application logs

D. Docker Commands Reference

```
# View status
docker compose ps

# View logs
docker compose logs -f backend

# Restart service
docker compose restart backend

# Rebuild and restart
docker compose up -d --build backend

# Enter container shell
docker compose exec backend /bin/bash

# View resource usage
docker stats
```

E. Support

For issues and feature requests: - GitHub: https://github.com/your-org/c2monitor - Documentation: https://dude.pmarines.org/docs

---

Document Version: 1.0.0 Last Updated: January 2026