CyberSentinel Email Security Dashboard

API Manual

Table of Contents

Introduction

The CyberSentinel Email Security Dashboard provides a set of APIs for integrating email security features into your existing applications. This manual covers all available endpoints, authentication methods, and examples of how to use them.

API Overview

CyberSentinel APIs follow RESTful principles and use JSON for data exchange. All API requests are made to the base URL of your CyberSentinel installation.

Base URL

https://your-installation-url.com/api

API Versioning

API version is specified in the URL path:

https://your-installation-url.com/api/v1/[endpoint]

Response Format

All API responses use the following JSON structure:

```
{
  "success": true,
  "data": { ... },
  "message": "Operation successful"
}
```

Or for errors:

```
{
  "success": false,
  "error": {
    "code": "ERROR_CODE",
    "message": "Error description"
  }
}
```

Authentication

API Key Authentication

Most API endpoints require an API key for authentication.

1. Generate an API key in the admin dashboard

Include the API key in the `Authorization` header:
 Authorization: Bearer YOUR_API_KEY

2.

User Authentication

Some endpoints require user-level authentication:

1. Obtain a JWT token by calling the login endpoint

Include the token in the `Authorization` header:
 Authorization: Bearer YOUR_JWT_TOKEN

2.

Obtaining a JWT Token

POST /api/v1/auth/login
Content-Type: application/json

```
{
  "email": "user@example.com",
  "password": "secure_password"
}
```

Response:

```
{
  "success": true,
  "data": {
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "user": {
      "id": 1,
      "email": "user@example.com",
      "role": "admin"
    }
  },
  "message": "Authentication successful"
}
```

Email Analysis API

Analyze Email

Analyzes an email for potential security threats.

POST /api/v1/analyze-email
Authorization: Bearer YOUR_API_KEY
Content-Type: application/json

```
{
  "sender": "example@domain.com",
  "subject": "Email subject line",
  "content": "Full email content goes here..."
}
```

Response:

```
{
  "success": true,
  "data": {
    "riskLevel": "suspicious",
    "confidence": 85,
    "indicators": [
      "suspicious sender domain",
      "urgent language"
    ],
    "analysis": "This email contains elements commonly found in phishing attempts",
    "suspiciousLinks": [
      {
        "url": "https://suspicious-url.com",
        "reason": "Domain typosquatting detected"
      }
    ],
    "recommendedAction": "Review this email carefully before deciding whether to deliver it to the recipient"
  },
  "message": "Email analysis completed"
}
```


Get Email List

Retrieves a list of analyzed emails.

GET /api/v1/emails
Authorization: Bearer YOUR_JWT_TOKEN

Optional query parameters:

- `risk_level`: Filter by risk level (phishing, suspicious, safe)
- `status`: Filter by status (flagged, reviewing, blocked, cleared)
- `limit`: Number of results to return (default: 50)
- `offset`: Pagination offset (default: 0)

Response:


```
{
  "success": true,
  "data": {
    "emails": [
```

```
    {
      "id": 1,
      "sender": "example@domain.com",
      "subject": "Email subject",
      "receivedAt": "2025-05-07T12:34:56Z",
      "riskLevel": "suspicious",
      "status": "reviewing"
    }
   ],
   "total": 120,
   "limit": 50,
   "offset": 0
  },
  "message": "Emails retrieved successfully"
}
```


Get Email Details

Retrieves detailed information about a specific email.

```
GET /api/v1/emails/{email_id}
Authorization: Bearer YOUR_JWT_TOKEN
```

Response:


```
{
  "success": true,
  "data": {
    "id": 1,
    "sender": "example@domain.com",
    "subject": "Email subject",
    "receivedAt": "2025-05-07T12:34:56Z",
    "content": "Full email content...",
    "riskLevel": "suspicious",
    "status": "reviewing",
    "indicators": ["suspicious sender domain", "urgent language"],
    "recipient": "recipient@company.com",
    "links": [
      {
        "url": "https://example.com",
        "isSuspicious": false,
        "reason": ""
      }
```

```
    ],
    "attachments": []
  },
  "message": "Email details retrieved successfully"
}
```


Update Email Status

Updates the status of an email.

```
PUT /api/v1/emails/{email_id}/status
Authorization: Bearer YOUR_JWT_TOKEN
Content-Type: application/json

{
  "status": "blocked"
}
```

Response:


```
{
  "success": true,
  "data": {
    "id": 1,
    "status": "blocked"
  },
  "message": "Email status updated successfully"
}
```


Suggest Risk Level Correction

Suggests a correction to an email's risk level assessment.

```
POST /api/v1/emails/{email_id}/suggest-correction
Authorization: Bearer YOUR_JWT_TOKEN
Content-Type: application/json

{
  "newRiskLevel": "safe",
  "feedback": "This appears to be a legitimate email from our partner"
}
```

Response:


```
{
  "success": true,
  "data": {
    "id": 1,
    "previousRiskLevel": "suspicious",
    "suggestedRiskLevel": "safe",
    "status": "under_review"
  },
  "message": "Correction suggestion submitted successfully"
}
```


User Management API

Create User

Creates a new user account.

```
POST /api/v1/users
Authorization: Bearer YOUR_API_KEY
Content-Type: application/json
```


```
{
  "email": "newuser@example.com",
  "password": "secure_password",
  "role": "analyst"
}
```


Response:


```
{
  "success": true,
  "data": {
    "id": 2,
    "email": "newuser@example.com",
    "role": "analyst"
  },
  "message": "User created successfully"
}
```

Get User List

Retrieves a list of users.

GET /api/v1/users
Authorization: Bearer YOUR_JWT_TOKEN

Response:


```
{
  "success": true,
  "data": {
    "users": [
      {
        "id": 1,
        "email": "admin@example.com",
        "role": "admin"
      },
      {
        "id": 2,
        "email": "analyst@example.com",
        "role": "analyst"
      }
    ]
  },
  "message": "Users retrieved successfully"
}
```


Update User Role

Updates a user's role.

PUT /api/v1/users/{user_id}/role
Authorization: Bearer YOUR_JWT_TOKEN
Content-Type: application/json


```
{
  "role": "admin"
}
```


Response:

```
{
  "success": true,
  "data": {
    "id": 2,
    "email": "user@example.com",
    "role": "admin"
  },
  "message": "User role updated successfully"
}
```


Webhook Integration

CyberSentinel can send webhook notifications for important events.

Configure Webhook

Sets up a webhook endpoint.

```
POST /api/v1/webhooks
Authorization: Bearer YOUR_API_KEY
Content-Type: application/json

{
  "url": "https://your-application.com/webhooks/cybersentinel",
  "secret": "your_webhook_secret",
  "events": ["email.phishing", "email.suspicious", "email.blocked"]
}
```

Response:


```
{
  "success": true,
  "data": {
    "id": 1,
    "url": "https://your-application.com/webhooks/cybersentinel",
    "events": ["email.phishing", "email.suspicious", "email.blocked"]
  },
  "message": "Webhook configured successfully"
}
```


Webhook Payload Example

When a configured event occurs, CyberSentinel sends a JSON payload to your webhook URL:

```
{
  "event": "email.phishing",
  "timestamp": "2025-05-07T12:34:56Z",
  "data": {
    "emailId": 1,
    "sender": "suspicious@example.com",
    "subject": "Urgent: Verify your account",
    "riskLevel": "phishing",
    "confidence": 92
  }
}
```

Rate Limits

To ensure service stability, the API implements rate limiting:

- Standard tier: 60 requests per minute
- Professional tier: 300 requests per minute
- Enterprise tier: Custom limits based on requirements

When a rate limit is exceeded, the API returns a 429 Too Many Requests response with information about when you can retry:

```
{
  "success": false,
  "error": {
    "code": "RATE_LIMIT_EXCEEDED",
    "message": "Rate limit exceeded. Please try again in 37 seconds."
  }
}
```

Error Handling

Common Error Codes

Code	HTTP Status	Description
AUTHENTICATION_FAILED	401	Invalid API key or JWT token

AUTHORIZATION_FAILED	403	User doesn't have permission for this action
RESOURCE_NOT_FOUND	404	Requested resource doesn't exist
VALIDATION_ERROR	422	Invalid request parameters
RATE_LIMIT_EXCEEDED	429	Too many requests
SERVER_ERROR	500	Internal server error

Error Response Example

```
{
  "success": false,
  "error": {
    "code": "VALIDATION_ERROR",
    "message": "Invalid email format",
    "details": {
      "field": "sender",
      "constraint": "email"
    }
  }
}
```