PhishIQ Plus – API & Integration Overview

Introduction

PhishIQ Plus is designed for enterprise security teams that operate centralized monitoring environments such as Security Operations Centers (SOC), SIEM-centric architectures, and MSSPs. The API accepts events containing suspicious links and returns enriched, risk-scored, structured security events in JSON format, optimized for direct ingestion into SIEM systems and SOC investigation workflows.

Unlike PhishIQ API, which is intended for developers embedding phishing detection into applications and products during development, PhishIQ Plus is purpose-built for operational security: it focuses on structured security event generation, prioritization, and SOC-level management. Both products are powered by the same core detection engine but are optimized for different integration models and use cases.

Prerequisites

- Valid PhishIQ Plus API access (API key provided for your environment).
- Ability to send HTTPS requests from your infrastructure (e.g., SIEM connector, SOAR platform, or internal server).
- Understanding of the event structure you send and the structured security event format you receive.
- (Optional) A SIEM or event pipeline to validate integration and event flow.

Authentication

All requests to the PhishIQ Plus API must include your API key for authentication. The API key is passed in the request header.

Passing the API Key in Requests

Include the `x-api-key` header in every request with your API key as the value.

Example using cURL:

```
curl -X POST "https://api.example.com/phishiq-plus/v1/analyze" \
  -H "x-api-key: YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{"url": "https://example.com/link-to-check", "event_id": "optional-correlation
```

Security note: Use the API only from the server side (e.g., SIEM connector or backend). Never expose your API key in client-side code, logs, or public repositories.

Endpoints and Request Structure

The following describes the typical structure for submitting a link for analysis and receiving a structured security event. Base URL, path, and parameter names may vary according to your deployment; refer to your provisioning documentation for exact values.

Element	Description
Base URL	Provided with your API credentials (e.g., `https://api.example.com/phishiq-plus/v1`).
Request method	POST (recommended for sending event payloads with URL and optional metadata).
Request body	JSON object containing at least the URL to analyze; may include `event_id` , `source` , or other optional fields as specified in your API documentation.

Example request body (JSON):

```
{
  "url": "https://example.com/suspicious-link",
  "event_id": "optional-correlation-id",
  "source": "siem-connector"
}
```

Response Format

The API returns a JSON object representing a structured security event, designed for direct ingestion into SIEM and SOC workflows. Typical fields include:

Field	Description
`risk_score`	Numeric or categorical risk score for prioritization.
`threat_classification`	Classification of the threat (e.g., phishing, credential harvesting, fraud).
`safety_status`	Overall safety indication (e.g., safe, suspicious, malicious).
`recommended_action`	Suggested action for the SOC (e.g., block, investigate, allow).
`technical_indicators`	Technical details useful for correlation and investigation.
`event_id / timestamp`	Correlation and timing information as applicable.

Exact field names and values are defined in your API specification. The structure is SIEM-optimized so that events can be ingested without additional transformation.

Example Response

```
{
  "event_id": "optional-correlation-id",
  "risk_score": 85,
  "threat_classification": "phishing",
  "safety_status": "malicious",
  "recommended_action": "block",
  "technical_indicators": { ... },
  "timestamp": "2025-02-21T12:00:00Z"
}
```

Error Handling

Requests may fail due to invalid input, authentication issues, or server errors. Implement proper error handling so that your SIEM pipeline remains reliable and you can detect and respond to failures quickly.

Common HTTP Status Codes

Code	Meaning
200	Success; response body contains the structured security event.
400	Bad request (e.g., missing or invalid parameters).
401	Unauthorized (e.g., missing or invalid API key).
429	Too many requests; consider backoff and retry.
500	Server error; retry may succeed later.

Error Response Format

When an error occurs, the API returns a JSON response with the following fields:

- `error_code` – A short string identifying the type of error.
- `message` – A human-readable description of the error.
- `details` (optional) – Additional information that may help resolve the issue.

Example Error Response

```
{
  "error_code": "INVALID_API_KEY",
  "message": "The API key provided is invalid or has expired."
}
```

Guidelines for Proper and Secure Use of the API

- **Use server-side only.** Send requests only from your backend, SIEM connector, or trusted server. Never expose API keys in client-side code, browser extensions, or public repositories.
- **Separate environments.** Use a dedicated API key for each environment (e.g., production and test) to avoid mistakes and limit blast radius.
- **Handle error responses.** Implement robust handling of non-2xx responses and parse error payloads so that your pipeline can log, alert, or retry as appropriate.
- **Keep your keys secure.** Store API keys in a secure secrets manager or configuration store, and follow accepted security practices for credential management.
- **Integrate in an event-oriented way.** Treat the API as part of an event pipeline: send events containing links and consume the returned structured security events in your SIEM or SOC workflow.

Stay Updated on Changes

Improvements, changes, and alerts are published periodically. Follow updates to the API and documentation to ensure continued compatibility with your integration.

Support

For any issues or questions, please contact our support at:

support@ntrigo.com