

# VIOE - Vulnerability Intelligence & Orchestration Engine

---

## Integration & API Guide

---

**Document Version:** 1.0 **Classification:** Internal / Technical Partners **Last Updated:** January 2026

---

## Table of Contents

---

1. [Introduction](#)
  2. [Available Integrations](#)
  3. [Scanner Integrations](#)
  4. [Workflow Integrations](#)
  5. [Communication Integrations](#)
  6. [Directory Integrations](#)
  7. [API Overview](#)
  8. [Authentication Approach](#)
  9. [Webhooks](#)
  10. [Integration Examples](#)
  11. [Best Practices](#)
  12. [Troubleshooting](#)
- 

## 1. Introduction

---

### 1.1 Purpose

This guide provides comprehensive information about VIOE's integration capabilities, enabling organizations to connect their existing tools and extend platform functionality.

### 1.2 Integration Philosophy

VIOE is designed as an integration hub for security operations:

- **Data Ingestion:** Import vulnerability data from multiple sources
- **Workflow Automation:** Sync with task management systems
- **Communication:** Integrate with team collaboration tools
- **Identity:** Leverage existing directory services

### 1.3 Integration Categories

Category	Purpose	Examples
Scanner	Import vulnerability data	Snyk, Qualys, Tenable
Workflow	Task management	Jira
Communication	Team notifications	Slack, Email
Directory	User/team mapping	Okta, Active Directory

## 2. Available Integrations

### 2.1 Integration Matrix

Integration	Type	Direction	Status
Snyk	Scanner	Inbound	Supported
SonarQube	Scanner	Inbound	Supported
Checkmarx	Scanner	Inbound	Supported
Qualys	Scanner	Inbound	Supported
Tenable	Scanner	Inbound	Supported
Rapid7	Scanner	Inbound	Supported
Jira	Workflow	Bidirectional	Supported
Slack	Communication	Outbound	Supported
Email	Communication	Outbound	Supported
Okta	Directory	Inbound	Supported
Active Directory	Directory	Inbound	Supported

## 2.2 Integration Requirements

Integration	Requirements
All	Network connectivity to external service
API-based	Valid API credentials
OAuth-based	Authorization workflow completion
File-based	Supported file format

## 3. Scanner Integrations

### 3.1 Snyk Integration

**Overview:** Imports application security vulnerabilities from Snyk.

**Connection Method:** API

### Required Credentials:

Credential	Source
API Token	Snyk Dashboard → Account Settings
Organization ID	Snyk Dashboard → Organization Settings

### Data Imported:

- Vulnerability title and description
- CVE identifiers
- CVSS scores
- Affected package/dependency
- Remediation recommendations

### Configuration Steps:

1. Navigate to Settings → Integrations → Snyk
2. Enter API Token
3. Enter Organization ID
4. (Optional) Select specific projects to sync
5. Test connection
6. Save configuration
7. Configure sync schedule

### Sync Options:

Option	Description
Real-time	Immediate sync on new findings
Hourly	Sync every hour
Daily	Sync once per day
Manual	On-demand only

## 3.2 SonarQube Integration

**Overview:** Imports code quality and security issues from SonarQube.

**Connection Method:** API

### Required Credentials:

Credential	Source
Server URL	Your SonarQube instance URL
API Token	SonarQube → User → My Account → Security

### Data Imported:

- Security hotspots
- Vulnerability issues
- Code smell security implications
- Severity and effort estimates

### Configuration Steps:

1. Navigate to Settings → Integrations → SonarQube
2. Enter Server URL
3. Enter API Token
4. Select project key(s)
5. Test connection
6. Save and schedule

## 3.3 Checkmarx Integration

**Overview:** Imports SAST findings from Checkmarx.

**Connection Method:** API / File Import

### Required Credentials (API):

Credential	Source
Server URL	Checkmarx server URL
Username	Checkmarx account
Password	Checkmarx account

### Data Imported:

- Static analysis findings
- Severity classifications

- Code locations
- Remediation guidance

## 3.4 Qualys Integration

**Overview:** Imports infrastructure vulnerabilities from Qualys.

**Connection Method:** API / File Import

**Required Credentials:**

Credential	Source
API URL	Qualys platform URL
Username	Qualys account
Password	Qualys account

**Data Imported:**

- Infrastructure vulnerabilities
- QID identifiers
- CVE mappings
- CVSS scores
- Asset information

## 3.5 Tenable Integration

**Overview:** Imports vulnerability scan results from Tenable.io or Tenable.sc.

**Connection Method:** API / File Import

**Required Credentials:**

Credential	Source
Access Key	Tenable → Settings → API Keys
Secret Key	Tenable → Settings → API Keys

**Data Imported:**

- Plugin-based vulnerabilities

- CVE and CVSS information
- Asset details
- Severity ratings

## 3.6 Rapid7 Integration

**Overview:** Imports security assessment data from Rapid7 InsightVM.

**Connection Method:** API / File Import

**Required Credentials:**

Credential	Source
API URL	InsightVM console URL
API Key	InsightVM → Administration → API Keys

**Data Imported:**

- Vulnerability findings
- Risk scores
- Asset information
- Remediation steps

## 3.7 File Import (All Scanners)

For scanners without API integration or air-gapped environments:

**Supported Formats:**

Format	Extension	Notes
CSV	.csv	Column mapping required
JSON	.json	Schema-compliant
Excel	.xlsx, .xls	First sheet parsed
PDF	.pdf	Text extraction

**Required Fields:**

Field	Required	Description
Title	Yes	Vulnerability name
Severity	Yes	Critical/High/Medium/Low
CVE	No	CVE identifier
Description	No	Full description
Asset	No	Affected asset
Environment	No	Production/Staging/Dev

## 4. Workflow Integrations

### 4.1 Jira Integration

**Overview:** Bidirectional integration with Atlassian Jira for task management.

**Connection Method:** OAuth / API Token

**Capabilities:**

Function	Direction	Description
Create Issue	VIOE → Jira	Create Jira issue from task
Sync Status	Bidirectional	Status updates sync both ways
Link Issues	VIOE → Jira	Store Jira issue key
View in Jira	VIOE → Jira	Direct link to issue

**Required Credentials:**

Credential	Source
Jira URL	Your Atlassian instance URL
Email	Atlassian account email
API Token	Atlassian → Account Settings → API Tokens

### Configuration Steps:

1. Navigate to Settings → Integrations → Jira
2. Enter Jira instance URL
3. Enter email address
4. Enter API token
5. Select default project
6. Configure issue type
7. Map fields (optional)
8. Test connection
9. Save configuration

### Field Mapping:

VIOE Field	Jira Field	Mapping
Task Title	Summary	Direct
Description	Description	Direct
Priority	Priority	Configurable
Status	Status	Configurable
Team	Assignee	Optional

### Status Mapping Example:

VIOE Status	Jira Status
Todo	To Do
In Progress	In Progress
In Review	In Review
Completed	Done
Blocked	Blocked

### Sync Behavior:

- VIOE task update → Jira issue updated
- Jira issue update → VIOE task updated

- Sync frequency: Real-time or on-demand

---

## 5. Communication Integrations

---

### 5.1 Slack Integration

**Overview:** Send notifications to Slack channels for team awareness.

**Connection Method:** OAuth

**Capabilities:**

Function	Description
Team Notifications	New vulnerability assigned
Critical Alerts	Critical findings detected
SLA Warnings	Approaching deadlines
Daily Digest	Summary of activity

**Configuration Steps:**

1. Navigate to Settings → Integrations → Slack
2. Click "Connect to Slack"
3. Authorize VIOE in your Slack workspace
4. Select default notification channel
5. Map teams to channels
6. Configure notification types
7. Test notification
8. Save configuration

**Team-Channel Mapping:**

Team	Slack Channel
Platform Engineering	#platform-security
Backend Services	#backend-security
Frontend Team	#frontend-security

**Notification Types:**

Type	Trigger	Default
New Assignment	Vulnerability assigned to team	Enabled
Critical Finding	Critical severity imported	Enabled
SLA Warning	48 hours before deadline	Enabled
Daily Summary	End of day	Optional

**Message Format:**

Critical Vulnerability Assigned

Title: SQL Injection in Login Form

CVE: CVE-2025-12345

Assigned to: Backend Services

Confidence: 95% (High)

SLA: 7 days

[View in VIOE]

[Create Task]

**5.2 Email Integration**

**Overview:** Email notifications for alerts and reports.

**Configuration:**

- Uses organization's email delivery system
- Configurable per user
- HTML-formatted messages

**Email Types:**

Type	Recipients	Frequency
Critical Alerts	Assigned team, managers	Immediate
SLA Warnings	Task owner	48 hours before
Daily Digest	Subscribed users	Daily
Weekly Report	Leadership	Weekly

---

## 6. Directory Integrations

---

### 6.1 Okta Integration

**Overview:** Leverage Okta for user authentication and organizational mapping.

**Capabilities:**

Function	Description
SSO	Single sign-on authentication
User Provisioning	Automatic user creation
Group Mapping	Map Okta groups to VIOE teams
Profile Sync	Keep user data current

**Configuration Steps:**

1. Create VIOE application in Okta
2. Configure SAML or OIDC settings
3. Map Okta groups to VIOE teams
4. Enable user provisioning
5. Test authentication flow

**Group Mapping Example:**

Okta Group	VIOE Team
Engineering-Platform	Platform Engineering
Engineering-Backend	Backend Services
Security-Team	Security Analysts

### 6.2 Active Directory Integration

**Overview:** Integrate with on-premises Active Directory.

**Capabilities:**

Function	Description
Authentication	AD credentials for login
Group Sync	Map AD groups to teams
User Attributes	Import user metadata

**Connection Method:** LDAPS (Secure LDAP)

#### Configuration Requirements:

Requirement	Description
Domain Controller	AD server address
Base DN	Search base
Bind Account	Service account credentials
Group Filter	Groups to sync

## 7. API Overview

### 7.1 API Architecture

VIOE provides RESTful APIs for programmatic access.

**Base URL:** `https://[your-instance]/api/v1`

**API Domains:**

Domain	Base Path	Purpose
Vulnerabilities	/vulnerabilities	Vulnerability management
Tasks	/tasks	Remediation tasks
Assets	/assets	Asset inventory
Teams	/teams	Team management
Reports	/reports	Report generation
Import	/import	Data ingestion

## 7.2 API Capabilities

Operation	HTTP Method	Example
List	GET	GET /vulnerabilities
Retrieve	GET	GET /vulnerabilities/{id}
Create	POST	POST /vulnerabilities
Update	PUT/PATCH	PATCH /vulnerabilities/{id}
Delete	DELETE	DELETE /vulnerabilities/{id}

## 7.3 Response Format

All API responses use JSON format:

```
{
  "data": {
    // Response payload
  },
  "meta": {
    "total": 100,
    "page": 1,
    "per_page": 50
  }
}
```

**Error Response:**

```
{
  "error": {
    "code": "VALIDATION_ERROR",
    "message": "Invalid severity value",
    "details": {
      "field": "severity",
      "allowed": ["critical", "high", "medium", "low"]
    }
  }
}
```

## 7.4 Rate Limiting

Limit Type	Value
Requests per minute	100
Requests per hour	1000
Burst allowance	20

Rate limit headers returned:

- X-RateLimit-Limit
- X-RateLimit-Remaining
- X-RateLimit-Reset

## 8. Authentication Approach

### 8.1 API Authentication

**Method:** Bearer Token

**Obtaining Token:**

1. Navigate to Settings → API Keys
2. Click "Generate API Key"
3. Copy and securely store the key
4. Key is shown only once

## Using Token:

```
Authorization: Bearer <your-api-token>
```

## 8.2 Token Management

Action	Procedure
Generate	Settings → API Keys → Generate
Revoke	Settings → API Keys → Delete
Rotate	Generate new, update clients, revoke old

### Best Practices:

- Use separate tokens per integration
- Rotate tokens annually
- Revoke unused tokens
- Never share tokens in code repositories

## 8.3 OAuth Integrations

For user-facing integrations (Slack, Jira):

Step	Description
1	User initiates connection
2	Redirect to provider's auth page
3	User authorizes VIOE
4	Provider redirects with code
5	VIOE exchanges code for token
6	Token stored securely

## 9. Webhooks

---

### 9.1 Webhook Overview

VIOE can send webhooks for event-driven integrations.

#### Supported Events:

Event	Trigger
vulnerability.created	New vulnerability imported
vulnerability.updated	Vulnerability changed
vulnerability.resolved	Vulnerability resolved
task.created	New task created
task.completed	Task marked complete
incident.detected	New incident

### 9.2 Webhook Configuration

#### Configuration Steps:

1. Navigate to Settings → Webhooks
2. Click "Add Webhook"
3. Enter endpoint URL
4. Select events to subscribe
5. (Optional) Add secret for signature
6. Test webhook
7. Save

### 9.3 Webhook Payload

#### Example Payload:

```
{
  "event": "vulnerability.created",
  "timestamp": "2026-01-13T10:30:00Z",
  "data": {
```

```
"id": "vuln_123456",
"title": "SQL Injection Vulnerability",
"severity": "critical",
"cve_id": "CVE-2025-12345",
"assigned_team": "Backend Services",
"status": "open"
}
}
```

## 9.4 Webhook Security

### Signature Verification:

- HMAC-SHA256 signature in header
- Header: X-VIOE-Signature
- Verify against your webhook secret

### Verification Process:

1. Get raw request body
2. Compute HMAC-SHA256 with your secret
3. Compare to signature header
4. Reject if mismatch

---

## 10. Integration Examples

### 10.1 Import Vulnerabilities via API

#### Request:

```
POST /api/v1/import
Content-Type: application/json
Authorization: Bearer <token>

{
  "vulnerabilities": [
    {
      "title": "SQL Injection in User Search",
      "cve_id": "CVE-2025-12345",
      "severity": "critical",
```

```
    "description": "SQL injection vulnerability...",
    "asset": "user-service",
    "environment": "production"
  }
]
}
```

### Response:

```
{
  "data": {
    "imported": 1,
    "failed": 0,
    "duplicates": 0
  }
}
```

## 10.2 Create Task and Sync to Jira

### Request:

```
POST /api/v1/tasks
Content-Type: application/json
Authorization: Bearer <token>
```

```
{
  "title": "Fix SQL Injection in User Search",
  "vulnerability_id": "vuln_123456",
  "priority": "critical",
  "sync_to_jira": true
}
```

### Response:

```
{
  "data": {
    "id": "task_789",
    "title": "Fix SQL Injection in User Search",
    "status": "todo",
    "jira_issue_key": "SEC-1234",
    "jira_url": "https://company.atlassian.net/browse/SEC-1234"
  }
}
```

```
}  
}
```

## 10.3 Query Vulnerabilities with Filters

### Request:

```
GET /api/v1/vulnerabilities?severity=critical&status=open&team=backend  
Authorization: Bearer <token>
```

### Response:

```
{  
  "data": [  
    {  
      "id": "vuln_123456",  
      "title": "SQL Injection Vulnerability",  
      "severity": "critical",  
      "status": "open",  
      "assigned_team": "Backend Services"  
    }  
  ],  
  "meta": {  
    "total": 1,  
    "page": 1,  
    "per_page": 50  
  }  
}
```

---

## 11. Best Practices

### 11.1 Integration Setup

Practice	Description
Test in staging	Validate integrations before production
Use dedicated accounts	Service accounts for integrations
Document configurations	Record all integration settings
Monitor health	Regular integration health checks

## 11.2 Security Best Practices

Practice	Description
Rotate credentials	Annual rotation minimum
Least privilege	Grant minimum necessary permissions
Secure storage	Never store credentials in code
Audit access	Review integration activity

## 11.3 Performance Best Practices

Practice	Description
Batch imports	Group records for efficiency
Respect rate limits	Implement backoff
Cache appropriately	Reduce redundant calls
Use webhooks	Event-driven vs. polling

## 11.4 Maintenance Best Practices

Practice	Description
Monitor sync status	Check for failures
Update endpoints	Keep URLs current
Test after changes	Verify after updates
Document changes	Track modifications

---

# 12. Troubleshooting

---

## 12.1 Common Integration Issues

Issue	Cause	Resolution
Authentication failed	Invalid credentials	Regenerate token/key
Connection timeout	Network issue	Check connectivity
Rate limited	Too many requests	Implement backoff
Data format error	Schema mismatch	Verify format
Sync not updating	Webhook issue	Check endpoint

## 12.2 Diagnostic Steps

### For API Issues:

1. Verify token is valid
2. Check request format
3. Review error response
4. Test with minimal request
5. Check rate limit status

### For Webhook Issues:

1. Verify endpoint is accessible
2. Check firewall rules
3. Validate SSL certificate
4. Review signature verification
5. Check webhook logs

## 12.3 Getting Help

Resource	When to Use
This guide	First reference
Error messages	Check error code
Integration logs	Detailed debugging
Support team	Persistent issues

---

# Document Control

---

Version	Date	Author	Changes
1.0	January 2026	Documentation Team	Initial release

---

*This guide enables technical teams to successfully integrate VIOE with their ecosystem.*

**VIOE - Vulnerability Intelligence & Orchestration Engine** *Integration & API Guide*