Spark 1 Security

nOs1 Documentation

VERSION 1.0.29 SEPTEMBER 2025

About nOs1

n0s1 is an open-source secret scanner designed for ticketing and messaging tools such as Slack, Jira, Confluence, Asana, Wrike, Zendesk, Linear, GitHub, and GitLab. It scans all channels/tickets/items/issues within the chosen platform in search of any leaked secrets in the titles, bodies, messages and comments.

These secrets are identified by comparing them against an adaptable configuration file named regex.yaml. Alternative TOML format is also supported: regex.toml. The scanner specifically looks for sensitive information, which includes:

- Github Personal Access Tokens
- GitLab Personal Access Tokens
- AWS Access Tokens
- PKCS8 private keys
- RSA private keys
- SSH private keys
- npm access tokens

Currently supported target platforms:

Slack	Jira	Confluence	Asana
Wrike	Zendesk	Linear	GitHub
GitLab			

Quick Start

Get n0s1

n0s1 is available in the most common distribution channels. The complete list of installation options is available in the Integration section. Here are a few popular examples:

- python3 -m pip install n0s1 --upgrade
- docker run spark1security/n0s1
- Download source code from https://github.com/spark1security/n0s1
- See Integration for more

n0s1 is integrated with many popular platforms and applications. Here are a few popular options examples:

- GitHub Actions
- Docker

General Usage

```
n0s1 -h
usage: n0s1 [-h] COMMAND …
.                         .
positional arguments:
  COMMAND            Subcommands
    slack_scan       Scan Slack messages
    asana_scan       Scan Asana tasks
    zendesk_scan     Scan Zendesk tickets
    github_scan      Scan GitHub repos
    gitlab_scan      Scan GitLab repos
    wrike_scan       Scan Wrike tasks
    linear_scan      Scan Linear issues
    jira_scan        Scan Jira tickets
    confluence_scan  Scan Confluence pages
```

```
n0s1 jira_scan [-h] [--regex-file [REGEX_FILE]]
                    [--config-file [CONFIG_FILE]]
                    [--report-file [REPORT_FILE]] [--post-comment]
                    [--skip-comment] [--server [SERVER]] [--email [EMAIL]]
                    [--api-key [API_KEY]]
```

Examples:

Jira scan

```
n0s1 jira_scan --server "https://myjira.atlassian.net"
```

Result: scan Jira server myjira.atlassian.net using $JIRA_TOKEN env variable as API key. The report will be saved to n0s1_report.json

---

Slack scan

```
n0s1 slack_scan --api-key $SLACK_TOKEN
```

Result: scan Slack messages using $SLACK_TOKEN env variable as API key. The report will be saved to n0s1_report.json

---

Slack scan limiting the scope with Slack search

```
n0s1 slack_scan --api-key $SLACK_TOKEN --scope "search:database"
```

Result: scan all Slack messages with the word database

GitLab scan

```
n0s1 gitlab_scan --api-key $GITLAB_TOKEN --owner "spark1.us" --branch
"default"
```

Result: scan only the default branch of all repos under GitLab group spark1.us. Use PAT $GITLAB_TOKEN env variable as API key

---

GitHub scan

```
n0s1 github_scan --api-key $GITHUB_TOKEN --owner spark1security --repo
anthillpy
```

Result: scan public GitHub repository https://github.com/spark1security/anthillpy using PAT $GITHUB_TOKEN env variable as API key. The report will be saved to n0s1_report.json

---

GitHub scan limiting the scope with GitHub search

```
n0s1 github_scan --api-key $GITHUB_TOKEN --scope "search:org:spark1security
action in:name"
```

Result: scan all GitHub repositories with the word action in the name from the org spark1security

---

Confluence scan

```
n0s1 confluence_scan --server "https://myjira.atlassian.net"
```

Result: scan Confluence server myjira.atlassian.net/wiki using $CONFLUENCE_TOKEN env variable as API key. The report will be saved to n0s1_report.json

---

Confluence scan with CQL query

```
n0s1 confluence_scan --server "https://myjira.atlassian.net"--scope "cql:space=KB and type=page"
```

Result: scan all pages from workspace KB only on Confluence server myjira.atlassian.net/wiki

---

Jira scan API key paramater

```
n0s1 jira_scan --server "https://myjira.atlassian.net" --api-key "<YOUR_JIRA_API_TOKEN>"
```

Result: scan Jira server myjira.atlassian.net passing API key as an argument.

Jira scan with JQL query

```
n0s1 jira_scan --server "https://myjira.atlassian.net" --scope "jql:project=MAR OR project=\"Auto Service\""
```

Result: scan only Jira projects MAR and Auto Service.

Jira scan for large sites - Split scan in 5 parts

```
n0s1 jira_scan --server "https://myjira.atlassian.net" --map --map-file jira_map.json
n0s1 jira_scan --server "https://myjira.atlassian.net" --scope 1/5 --map-file jira_map.json –report-file jira_report_1_5.json
n0s1 jira_scan --server "https://myjira.atlassian.net" --scope 2/5 --map-file jira_map.json –report-file jira_report_2_5.json
n0s1 jira_scan --server "https://myjira.atlassian.net" --scope 3/5 --map-file jira_map.json –report-file jira_report_3_5.json
n0s1 jira_scan --server "https://myjira.atlassian.net" --scope 4/5 --map-file jira_map.json –report-file jira_report_4_5.json
n0s1 jira_scan --server "https://myjira.atlassian.net" --scope 4/5 --map-file jira_map.json –report-file jira_report_5_5.json
```

Result: map the Jira server myjira.atlassian.net and save the mapping to “jira_map.json”. Run 5 Jira scans covering 1/5 of the Jira site for each scan, and save the results to report files “jira_report_1_5.json” to “jira_report_5_5.json”.

Asana scan

```
n0s1 asana_scan
```

Result: scan Asana using $ASANA_TOKEN env variable as API key.

Zendesk scan

```
n0s1 zendesk_scan
```

Result: scan Zendesk using $ZENDESK_TOKEN env variable as API key.

Zendesk scan with query to limit scope

```
n0s1 zendesk_scan --scope "query:type:ticket subject:AWS"
```

Result: scan only Zendesk tickets that have the word AWS in the title.

Wrike scan

```
n0s1 wrike_scan
```

Result: scan Wrike using $WRIKE_TOKEN env variable as API key.

Linear scan

```
n0s1 linear_scan
```

Result: scan Linear server using $LINEAR_TOKEN env variable as API key.

Jira scan custom regex file

```
curl https://raw.githubusercontent.com/spark1security/n0s1/main/src/n0s1/config/regex.yaml -o my_regex.yaml
vi my_regex.yaml
n0s1 jira_scan --server "https://myjira.atlassian.net" --regex-file my_regex.yaml --report-file my_jira_report.json
```

Result: scan Jira server myjira.atlassian.net using $JIRA_TOKEN env variable as API key matching regexes from my_regex.yaml file and writing a report to my_jira_report.json.

Jira scan post comment to ticket

```
n0s1 jira_scan --server "https://myjira.atlassian.net" --post-comment
```

Result: scan Jira server myjira.atlassian.net and add a comment to Jira tickets warning users about potential secret exposure. A report will be also saved to my_jira_report.json.

Jira scan using docker and custom TOML regex file

```
docker run -v /home/user/my_regex.toml:/my_regex.toml spark1security/n0s1
jira_scan --server "https://myjira.atlassian.net" --email <MY_EMAIL> --api-key
$JIRA_TOKEN --post-comment --regex-file /my_regex.toml
```

Result: run n0s1 from a docker container to scan Jira server myjira.atlassian.net using a custom regex file “/home/user/my_regex.toml”

Integration

You can use n0s1 directly from your local machine by installing the pip package, or by downloading the source code. You can also use integration options such as Docker and GitHub Actions. See a few examples below:

Pip package

```
python3 -m pip install n0s1
n0s1 jira_scan --server "https://<YOUR_JIRA_SERVER>.atlassian.net" --api-key
"<YOUR_JIRA_API_TOKEN>"
```

From Source Code

```
git clone https://github.com/spark1security/n0s1.git
cd n0s1/src/n0s1
python3 -m venv n0s1_python
source n0s1_python/bin/activate
python3 -m pip install -r ../../requirements.txt
python3 n0s1.py jira_scan --server "https://<YOUR_JIRA_SERVER>.atlassian.net"
--api-key "<YOUR_JIRA_API_TOKEN>"
Deactivate
```

Docker

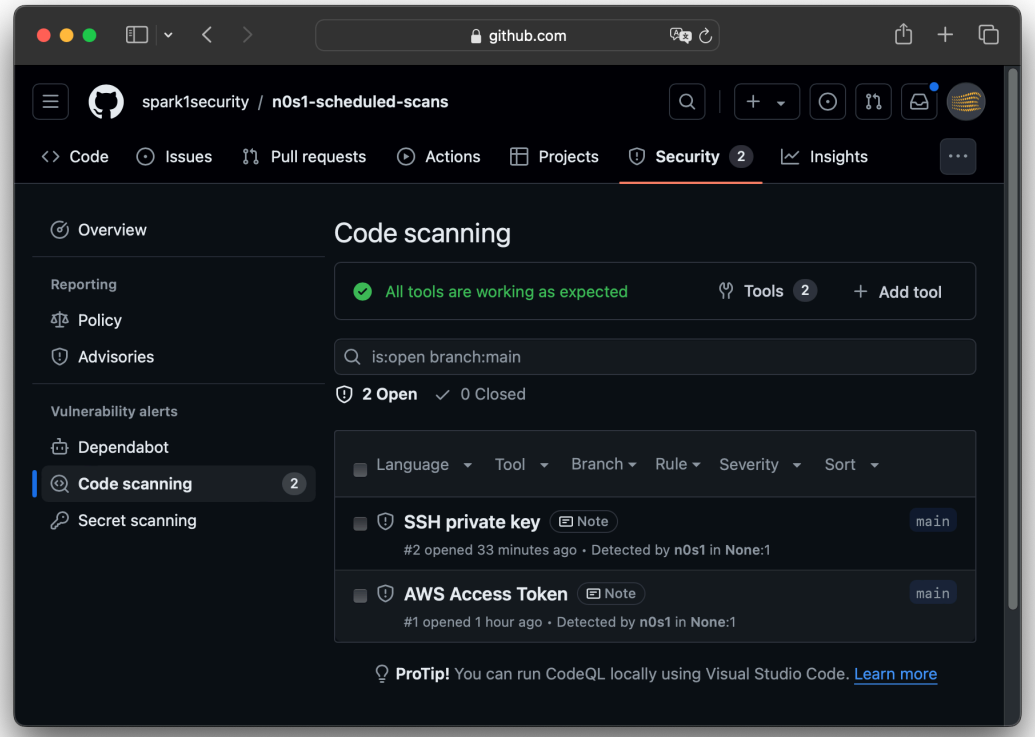
```
docker run spark1security/n0s1 jira_scan --server
"https://<YOUR_JIRA_SERVER>.atlassian.net" --api-key "<YOUR_JIRA_API_TOKEN>"
```

GitHub Actions

See n0s1.yml for a live example.

```
jobs:
```

```yaml
  jira_secret_scanning:
    steps:
      - name: Run n0s1 secret scanner on Jira
        uses: spark1security/n0s1-action@main
        env:
          JIRA_TOKEN: ${{ secrets.JIRA_API_TOKEN }}
        with:
          scan-target: 'jira_scan'
          user-email: 'service_account@<YOUR_COMPANY>.atlassian.net'
          platform-url: 'https://<YOUR_COMPANY>.atlassian.net'
          report-file: 'jira_leaked_secrets.sarif'
          report-format: 'sarif'
      - name: Upload n0s1 secret scan results to GitHub Security Codescanning
        uses: github/codeql-action/upload-sarif@v2
        with:
          sarif_file: jira_leaked_secrets.sarif
      - uses: actions/upload-artifact@v3
        with:
          name: n0s1-artifact
          path: jira_leaked_secrets.json
          retention-days: 5
```

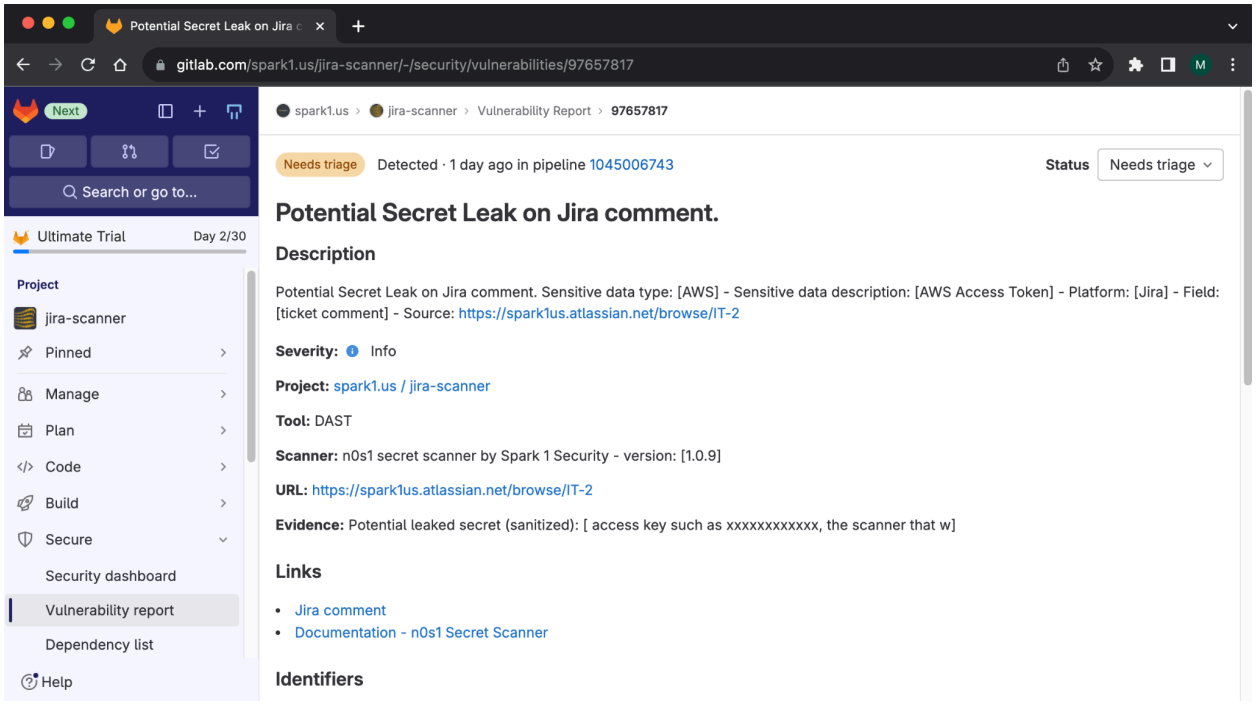


GitLab CI

Add the following job to your *.gitlab-ci.yml* file:

```yaml
jira-scan:
  stage: test
  image:
```

```
    name: spark1security/n0s1
    entrypoint: [""]
  script:
    - n0s1 jira_scan --email "service_account@<YOUR_COMPANY>.atlassian.net"
--api-key $JIRA_TOKEN --server "https://<YOUR_COMPANY>.atlassian.net"
    - apt-get update
    - apt-get -y install jq
    - cat n0s1_report.json | jq
```



User Manual

Inputs

Find the available command line CLI inputs parameters and GitHub Actions parameters available. The CLI inputs definition can be found here, and the GitHub Actions parameters list can be found here.

***Note:** `CLI inputs arguments in orange` and `GitHub Actions in rose`.

Name	Type	Default	Example	Description
`COMMAND` **`scan-target`**	String	None	jira_scan	*Command to be executed. It currently supports: slack_scan, jira_scan, confluence_scan, asana_scan, wrike_scan, zendesk and linear_scan*

--regex-file regex-file	String	<PIP-PKG>/config/regex.yaml	--regex-file ~/my_regs.yaml	*File .yaml or .toml with a list of regexes to be matched.*
--config-file config-file	String	<PIP-PKG>/config/config.yaml	--config-file ~/my_conf.yaml	*Configuration file (YAML format) to be used.*
--report-file report-file	String	./n0s1_report.json	--report-file report.json	*Output report file for the leaked secrets.*
--report-format report-format	String	n0s1	--report-format sarif	*Output report format. Supported formats: n0s1, sarif, gitlab.*
--post-comment post-comment	flag	disabled		*By default, scans only flag leaked secrets; this adds a warning comment to every ticket with a potential secret leak*
--skip-comment skip-comment	flag	disabled		*By default, scans check the ticket title, description and comments; this flag disables ticket comment scanning*
--api-key password-key	String	None		*Jira, Confluence or Linear API key. If not provided, n0s1 will try to use $JIRA_TOKEN, CONFLUENCE_TOKEN or $LINEAR_TOKEN env variables*
--server platform-url	String	None	--server "https://myjira.atlassian.net"	*Jira/Confluence server uri.*
--email user-email	String	None	--email svc@mycompany.com	*Jira/Confluence user email.*
--show-matched-secret-on-logs show-matched-secret-on-logs	flag	disabled		*By default, only a sanitized version of the leak is shown on logs. This flag makes the actual leaked secret to be displayed on logs. Be extra careful when enabling this flag because you might make the leak worse by sending sensitive info to logs.*
--secret-manager secret-manager	String	"a secret manager tool"	--secret-manager 1Password	*Secret manager tool name to be suggested when leaks are found.*
--contact-help contact-help	String		--contact-help “Security Team”	*Contact information for assistance when leaks are detected.*
--label label	String		--label fa4b101f-1ad5-4c18-9ad0-a2b89186f641	*Unique identifier to be added to the comments so that the n0s1 bot can recognize if the leak has been previously flagged.*

--debug **debug**	flag	disabled		*Debug mode. Warning it may further expose sensitive data.*
--insecure **insecure**	flag	disabled		*Insecure mode. Ignore SSL certificate verification.*
--timeout **timeout**	integer	disabled		*HTTP request timeout in seconds (currently supported only by Jira and Confluence scans).*
--limit **limit**	integer	disabled		*The limit of the number of pages to return per HTTP request (currently supported only by Jira and Confluence scans).*
--map **map**	integer	-1		*Generate a JSON file with a list of all tickets/pages/issues organized hierarchically for the target site. (currently supported only by Asana, Jira, and Confluence).*
--map-file **map-file**	String	./n0s1_map.json	--map-file jira_map.json	*Output JSON map file.*
--scope **scope**	String	disabled	--scope “jql:project=MAR” --scope 2/5	*Use a query to limit the scan scope.* *Or split the scan coverage in equal parts. c/n where "c" is the chunk number you want to scan, and "n" the total number of chunks you are splitting the map.*
--owner **owner**	String	Logged user	--owner spark1security	*GitHub owner (e.g. user or organization).*
--repo **repo**	String	All repos	--repo anthillpy	*GitHub repository to be scanned. If omitted all accessible repos under the owner will be scanned.*
--branch **branch**	String	All branches	--branch “test”	*GitHub/GitLab comma separated list of branches to be scanned. Ex: --branch “main,test”. If omitted all branches are scanned. If you provide --branch “default”, the branch marked as default is scanned, regardless its name.*

Community

n0s1 is a Spark 1 open source project.

Learn about our open source work and portfolio here.

Contact us about any matter by opening a GitHub Discussion here

Appendix A - Common use cases

Use case: One-off scan for passwords in Jira and Confluence

Scan Jira and Confluence using GitHub Actions with a customized regex set that includes "*(?i)password ?[=:] ?.**". Run the one-off scan from a separate branch "*password-search*" and add the results to GitHub Code scanning tab.

Make a local copy of the n0s1 regex config:

```
cd YOUR_REPO
curl https://raw.githubusercontent.com/spark1security/n0s1/main/src/n0s1/config/regex.toml -o my_regex.toml
```

Edit the customized my_regex.toml file to include the "*(?i)password ?[=:] ?.**" pattern. Create a "config" folder under your repo ".github/workflows" folder. Copy my_regex.toml to .github/workflows/config/my_regex.toml:

.github/workflows/config/my_regex.toml

```
[[rules]]
id = "plaintext_password"
description = "Plaintext Password"
regex = '''(?i)password ?[=:] ?.*'''
keywords = ["password"]

[[rules]]
id = "gitlab_personal_access_token"
description = "GitLab Personal Access Token"
.
.
.
```

Create a new GitHub Action secret-scan.yml:

.github/workflows/secret-scan.yml

```
name: secret-scan
on:
 workflow_dispatch:
 push:
   branches:
     - "password-search"
jobs:
 secret-scan:
   name: Scan secret from Jira, Confluence
   runs-on: docker
   steps:
```

```yaml
    - name: Checkout config
      uses: actions/checkout@v3
      with:
        ref: password-search
        sparse-checkout: |
          .github/workflows/config/my_regex.toml
    - name: Scan Jira for leaked credentials
      uses: spark1security/n0s1-action@main
      env:
        JIRA_TOKEN: ${{ secrets.JIRA_TOKEN }}
      with:
        scan-target: "jira_scan"
        user-email: "service_account@yourdomain.com"
        platform-url: "https://yourdomain.atlassian.net"
        regex-file: ".github/workflows/config/my_regex.toml"
        report-format: "sarif"
        report-file: "jira_secret_report.sarif"
    - name: Scan Confluence for leaked credentials
      uses: spark1security/n0s1-action@main
      env:
        CONFLUENCE_TOKEN: ${{ secrets.CONFLUENCE_TOKEN }}
      with:
        scan-target: "confluence_scan"
        platform-url: "https://confluence.yourdomain.com"
        regex-file: ".github/workflows/config/my_regex.toml"
        report-format: "sarif"
        report-file: "confluence_secret_report.sarif"
    - name: Upload scan results to GitHub Security tab
      uses: github/codeql-action/upload-sarif@v2
      with:
        sarif_file: "confluence_secret_report.sarif"
        category: confluence_secret_report
    - name: Upload scan results to GitHub Security tab
      uses: github/codeql-action/upload-sarif@v2
      with:
        sarif_file: "jira_secret_report.sarif"
        category: jira_secret_report
    - name: Store scan results as artifact
      uses: actions/upload-artifact@v3
      with:
        name: secret-scan-report
        path: |
          jira_secret_report.sarif
          confluence_secret_report.sarif
        retention-days: 5
```

Push the 2 new files to branch “password-search” to trigger the action:

```
git branch password-search
git add .github/workflows/config/my_regex.toml .github/workflows/secret-scan.yml
git commit -m “Scan Jira and Confluence for passwords”
git push
```

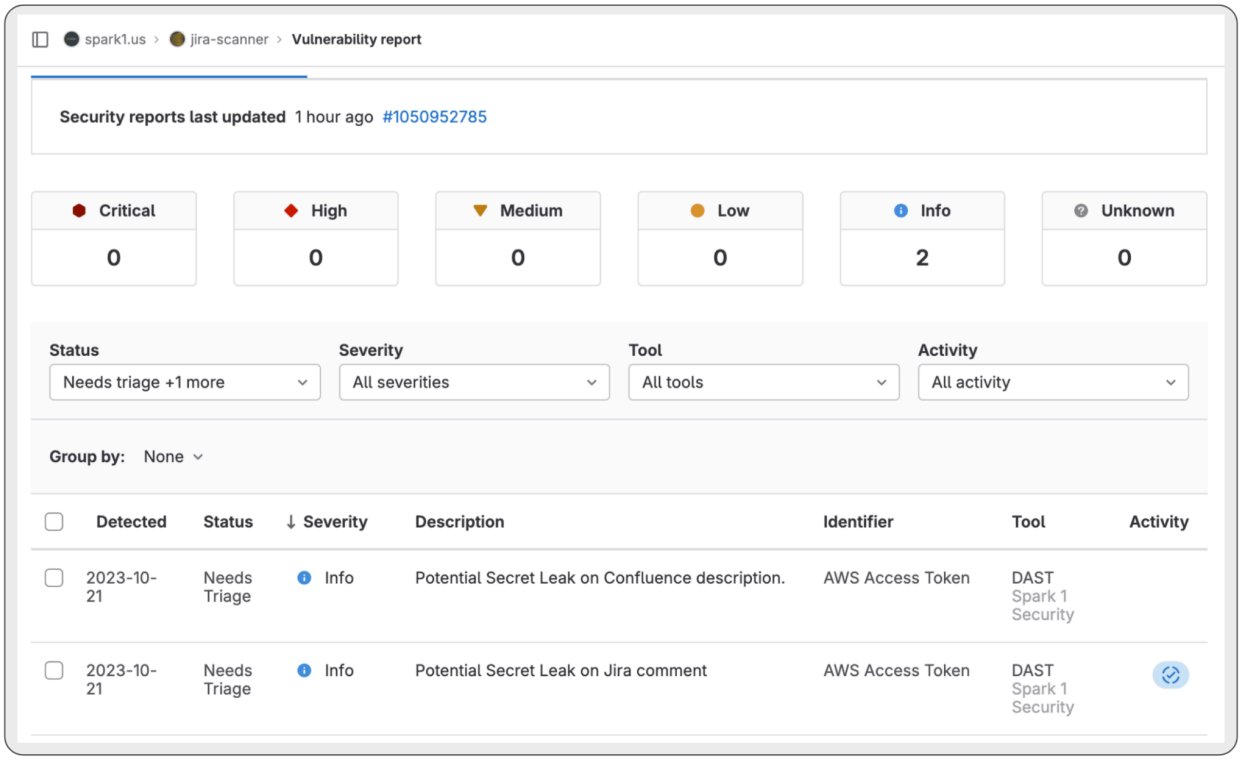
Use case: scan Jira and Confluence with GitLab CI

Scan Jira and Confluence using GitLab CI, display the results on CI job standard output and add the findings to GitLab Vulnerability report:

.gitlab-ci.yml

```
jira-scan:
 stage: test
 image:
   name: spark1security/n0s1
   entrypoint: [""]
 script:
   - n0s1 jira_scan --email "service_acc@yourdomain.com" --api-key $JIRA_TOKEN
--server "https://yourdomain.atlassian.net" --report-file gl-dast-report.json
--report-format gitlab
   - apt-get update
   - apt-get -y install jq
   - cat gl-dast-report.json | jq
 artifacts:
   reports:
     dast:
       - gl-dast-report.json

confluence-scan:
 stage: test
 image:
   name: spark1security/n0s1
   entrypoint: [""]
 script:
   - n0s1 confluence_scan --email "service_acc@yourdomain.com" --api-key
$JIRA_TOKEN --server "https://yourdomain.atlassian.net" --report-file
gl-dast-report.json --report-format gitlab
   - apt-get update
   - apt-get -y install jq
   - cat gl-dast-report.json | jq
 artifacts:
   reports:
     dast:
       - gl-dast-report.json
```


spark1.us › jira-scanner › Vulnerability report
Security reports last updated 1 hour ago #1050952785
Critical 0
High 0
Medium 0
Low 0
Info 2
Unknown 0
Status
Needs triage +1 more
Severity
All severities
Tool
All tools
Activity
All activity
Group by: None
Detected
Status
Severity
Description
Identifier
Tool
Activity
2023-10-21
Needs Triage
Info
Potential Secret Leak on Confluence description.
AWS Access Token
DAST Spark 1 Security
2023-10-21
Needs Triage
Info
Potential Secret Leak on Jira comment
AWS Access Token
DAST Spark 1 Security