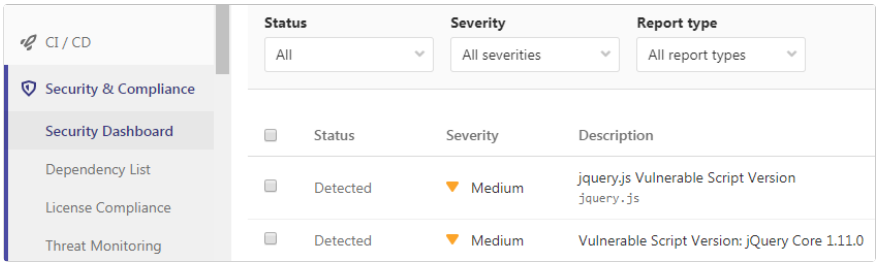
INTEGRATING SYHUNT INTO GITLAB

The information in this document applies to **version 6.8.6** of Syhunt Hybrid.

INTRODUCTION

Launching Syhunt scans from within a GitLab CI YML script is simple and straightforward, allowing you to integrate the Syhunt Dynamic, Syhunt Code and Syhunt Mobile security testing tools into your continuous delivery pipeline and security dashboard, schedule scans and more. You can also configure GitLab issue trackers in Syhunt, allowing vulnerabilities to be submitted to the issues area of projects.



The following example Gitlab CI YML script will scan the current repository source code, failing the job if medium or high vulnerabilities are identified. In addition to this, it attachs a PDF vulnerability report to the pipeline job artifacts and adds the identified vulnerabilities to GitLab's security dashboard.

```
syhunt_test:
  script:
          - Start-CodeScan -pfcond 'fail-if:risk=mediumup' -output 'report.pdf' -outputex 'gl-sast-repo
  artifacts:
    reports:
      sast: gl-sast-report.json
    paths:
    - report.pdf
    when: on_failure
  tags:
   - syhunt
```

ACTIVATING A SYHUNT RUNNER

Syhunt Runner is a CI service that will receive scan requests, execute scans and communicate the scan

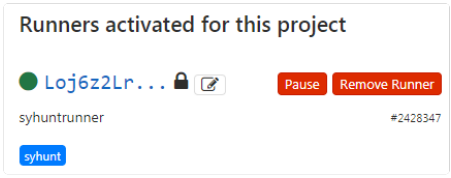
results with GitLab.

IMPORTANT: For security and performance reasons, it is advised that the Runner is installed on a separated virtual machine or dedicated physical machine.

1. First, go into your GitLab project settings and access the CI (Continuous Integration) options:
    1. Click **Settings**
    2. Click **CI / CD**
    3. Expand **Runners**
    4. Scroll down to **Set up a specific Runner manually**
    5. Save the registration token to a safe location. You will need it later below.

2. Install with its default settings Git for Windows, which can be downloaded at https://gitforwindows.org/
3. Install with its default settings Syhunt Hybrid (**syhunt-hybrid-6.9.0.exe**)
4. Download and run the Syhunt Runner (**syhunt-runner-13.1.0.exe**). After launching the setup, you will be prompted for the runner registration information.
    1. Paste the registration token you copied in the token text field and click Next to continue and complete the install.

After finishing installing, return to GitLab, and the **Runners** section we were just on, and press F5 to refresh the screen. The syhuntrunner should now be registered for this project. You will see it listed under Runners activated for this project at the bottom of the page, as shown in the screenshot below.



Syhunt is now ready to be called from CI YML scripts! See examples below

ADDING SYHUNT TO YOUR CI YML SCRIPT

If you don't have a CI YML file in your repository, go to **Project overview** and click **Set up CI/CD**. This will create a .gitlab-ci.yml file within the repository.

SAST Example - The following example Gitlab CI YML script will scan the current repository source code, failing the job if medium or high vulnerabilities are identified. In addition to this, it attachs a PDF vulnerability report to the pipeline job artifacts and adds the identified vulnerabilities to GitLab's security dashboard.

```
syhunt_test:
  stage: test
  script:
          - Start-CodeScan -pfcond 'fail-if:risk=mediumup' -output 'report.pdf' -outputex 'gl-sast-repo
  artifacts:
    reports:
      sast: gl-sast-report.json
    paths:
    - report.pdf
    when: on_failure
  tags:
   - syhunt
```

DAST Example - The following example Gitlab CI YML script will scan the live web application after going into production, failing the job if medium or high vulnerabilities are identified. In addition to this, it attachs a PDF vulnerability report to the pipeline job artifacts and adds the identified vulnerabilities to GitLab's security dashboard.

```
production:
stage: deploy
  script:
          - Start-DynamicScan -target 'www.productionurl.com' -pfcond 'fail-if:risk=mediumup' -output '
  artifacts:
    reports:
      sast: gl-dast-report.json
    paths:
    - report.pdf
    when: on_failure
  tags:
   - syhunt
only:
  - tags
```

Additional examples:

```
# SAST Example - Scan local directory/repository
Start-CodeScan -pfcond "fail-if:risk=mediumup"

# SAST Example - Scan a remote GIT repository
$MyProject = @{
  target = 'https://github.com/syhunt/vulnphp.git';
  branch = 'master';
  pfcond = 'fail-if:risk=mediumup';
  output = 'report.pdf'
}
Start-CodeScan @MyProject

# DAST Example - Scan URL
$MyWebsite= @{
  target = 'https://www.somewebsite.com';
  pfcond = 'fail-if:risk=mediumup';
  output = 'report.pdf'
}
Start-DynamicScan @MyWebsite
```

INTEGRATING WITH GITLAB'S SECURITY DASHBOARD

Syhunt will generate a GitLab-compatible vulnerability report file if the outputex parameter is simply set to:

- SAST: **gl-sast-report.json** or **anyfilename.gls.json**
- DAST: **gl-dast-report.json** or **anyfilename.gld.json**

Remember to add the filename to artifacts.reports.sast or artifacts.reports.dast to your CI YML file like shown in the first example above to activate the integration.

Important tip: If this is the first scan against a large codebase, it is recommended to scan your application without the dashboard integration activated to make sure you don't have a large number of vulnerabilities. If a large number of vulnerabilities is found, improve the security state of the application by fixing the initial batch of vulnerabilities reported by Syhunt and only then enable the dashboard integration.

START-DYNAMICSCAN FUNCTION

Syhunt Dynamic must be launched through the Start-DynamicScan() function. The following parameters must be provided when calling the Start-DynamicScan() function:

- **target** (required) - the target URL to be scanned (eg. http://www.somesite.com)
- **huntmethod** (optional) - the Hunt Method to be used during the scan, If omitted, the default method will be used.

- **pfcond** (optional) - allows the script to fail with proper exit code if a certain condition is met. See below a list of available pass/fail conditions.
- **output** (optional) - allows to set an output filename (eg. report.pdf or report.html).
- **outputex** (optional) - allows to set a second output filename (eg. export.json).
- **verbmode** (optional) - $false by default. If changed to true, turns on verbose mode allowing information other than error and basic info to be printed.
- **genrep** (optional) - $true by default. If changed to false, Syhunt will not generate an output file.
- **redirIO** (optional) - $true by default. If changed to false, input and output from the Syhunt scanner will not be redirected to the console.

When using the output or outputex parameters, all output formats supported by Syhunt are available. The report or export will be saved to the current working directory, unless a full path name is provided.

Examples:

```
# Example 1 - Scan URL with single line
Start-DynamicScan -target 'https://www.somewebsite.com' -pfcond 'fail-if:risk=mediumup'

# Example 2 - Scan URL
$MyWebsite= @{
  target = 'https://www.somewebsite.com';
  pfcond = 'failifriskmedium';
  output = 'report.pdf'
}
Start-DynamicScan @MyWebsite
```

START-CODESCAN FUNCTION

Syhunt Code must be launched through the Start-CodeScan() function. The following parameters can be provided when calling the Start-CodeScan() function, all of which are optional:

- **target** - the target URL of a GIT repository to be scanned, or a local source code directory or file. If the target parameter is omitted, the current working directory is scanned.
- **branch** - the repository branch to be scanned. If the branch parameter is omitted, the master branch is scanned.
- **huntmethod** - the Hunt Method to be used during the scan, If omitted, the default method will be used.
- **pfcond** - allows the script to fail with proper exit code if a certain condition is met. See below a list of available pass/fail conditions.
- **output** - allows to set an output filename (eg. report.pdf or report.html).
- **outputex** - allows to set a second output filename (eg. export.json).
- **verbmode** - $false by default. If changed to true, turns on verbose mode allowing information other than

error and basic info to be printed.

- **genrep** - $true by default. If changed to false, Syhunt will not generate an output file.
- **redirIO** - $true by default. If changed to false input and output from the Syhunt scanner will not be redirected to the console.

When using the output or outputex parameters, all output formats supported by Syhunt are available. The report or export will be saved to the current working directory, unless a full path name is provided.

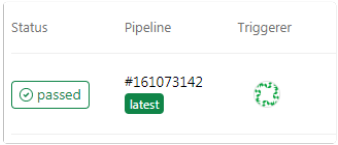
Examples:

```
# Example 1 - Scan the current directory/repository
Start-CodeScan -pfcond "fail-if:risk=mediumup"

# Example 2 - Scan a remote GIT project
Start-CodeScan -target "https://github.com/someuser/somerepo.git" -huntmethod "normal" -pfcond "fail-if

# Example 3 - Scan a specific local directory
Start-CodeScan -target "C:\www\" -huntmethod "normal" -pfcond "fail-if:risk=mediumup"
```

PASS/FAIL CONDITIONS



The following are the pass/fail conditions currently supported by Syhunt:

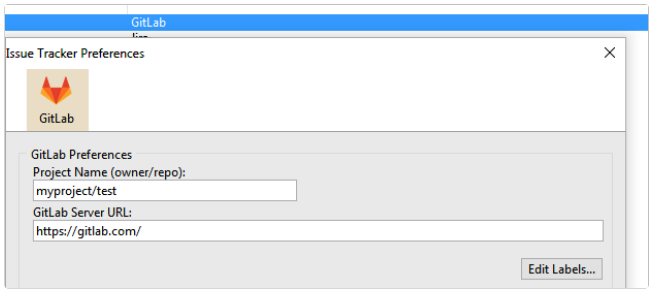
- `fail-if:risk=high` - Fail if a High risk vulnerability is found
- `fail-if:risk=mediumup` - Fail if a Medium or High risk vulnerability is found
- `fail-if:risk=lowup` - Fail if a Low, Medium or High risk vulnerability is found

ADVANCED RUNNER SETTINGS

Update the concurrent value for Runners in C:\SyhuntRunner\config.toml to allow multiple concurrent jobs as detailed in advanced configuration details.

INTEGRATING WITH GITLAB ISSUES

Configuring an issue tracker is an easy task and vulnerabilities can be submitted to a specific project with the click of a button.



Firstly, If you have not done so already, you have to create a personal access token with API access permission:

1. Log in to GitLab.
2. In the upper-right corner, click your avatar and select **Settings**.
3. On the User Settings menu, select **Access Tokens**.
4. Choose a name and optional expiry date for the token.
5. Choose the API scope.
6. Click the **Create personal access token** button.
7. Save the personal access token somewhere safe. Once you leave or refresh the page, you won't be able to access it again.

Finally, you have to add a GitLab tracker:

1. Click the Issue Trackers icon in the launcher toolbar in Syhunt. The Issue Trackers screen will open.
2. Click the Add Tracker icon in the Issue Trackers screen toolbar and choose the Add tracker: GitLab menu option.
3. Enter a reference name for the new tracker (like MyProjectName) and hit **OK**. A preferences dialog window will open.
4. Enter the GitLab project name. Format must be **owner/repo**.
5. Enter the GitLab Server URL, eg: `https://gitlab.com/` or your own server URL.
6. Enter your GitLab personal access token and click the **OK** button.

The tracker is ready! Right click the item you just edited in the list and click the **Submit Test Issue** option. If you configured everything properly, a test issue item should be created at `https://[gitlab_server]/[owner]/[repo]/issues`. If not, you will see an error message giving a hint of what needs to be done.

For more details on how to submit vulnerabilities to the GitLab tracker, see: Submitting Vulnerabilities To a Tracker.
