INTEGRATING SYHUNT INTO GITHUB

The information in this document applies to **version 6.9.8.2** of Syhunt Hybrid.

INTRODUCTION

Launching Syhunt scans from within a GitHub Actions 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. You can also configure GitHub issue trackers in Syhunt, allowing vulnerabilities to be submitted to the issues area of projects.

The following example GitHub Actions YML script will scan the current repository source code, failing the job if medium or high vulnerabilities are identified. In addition to this, and it attaches a PDF vulnerability report to the pipeline job artifacts.

```
name: syhunt test
on:
  push:
    branches:
      - main
jobs:
  build:
    runs-on: [self-hosted, Windows, X64]
    steps:
      - uses: actions/checkout@v2
      - run: Start-CodeScan -pfcond 'fail-if:risk=mediumup' -output 'report.pdf'
      - name: 'Syhunt Report'
        uses: actions/upload-artifact@v2
        if: failure()
        with:
         name: syhunt-report
         path: report.pdf
         if-no-files-found: error
```

ACTIVATING A SYHUNT RUNNER

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

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, install with its default settings Git for Windows, which can be downloaded at https://gitforwindows.org/
2. Secondly, install with its default settings Syhunt Hybrid (**syhunt-hybrid-6.9.14.1.exe**)
3. Finally, on GitHub.com, navigate to the main page of the repository:
    1. Under your repository name, click **Settings**.
    2. In the left sidebar, click **Actions**.
    3. In the left sidebar, under Actions, click **Runners**
    4. Click **New self-hosted runner**.
    5. Select Windows operating system and x64 architecture
    6. You will see instructions showing you how to download the runner application and install it on your self-hosted runner machine.

4. Follow the given instructions to activate the runner with the repository's token.
    1. When you execute **config.cmd**, it will be asked to enter the name of the group to which the runner will be added. Press enter for Default group.
    2. Enter a name to identify the runner. Example: **syhunt**
    3. Enter a label for the runner. Example: **syhunt**
    4. Finally you will see the messages below, which indicate success:
    5. **v Runner successfully added**
    6. **v Runner connection is good**
    7. Now press enter to keep **_work** as the work directory.
    8. Specify if you wish to execute the runner as a Windows service (YES or NO)

5. Now you must execute **run.cmd**. You should see the following message:
    1. **v Connected to GitHub**
    2. 2022-02-04 19:50:21Z: Listening for Jobs

Syhunt is now ready to be called from GitHub Actions scripts! See examples below

Note: A self-hosted runner is automatically removed from GitHub if it has not connected to GitHub Actions for more than 30 days. In addition to this, each workflow run is limited to 72 hours. If a workflow run reaches this limit, the workflow run is cancelled.

ADDING SYHUNT TO YOUR ACTIONS YML SCRIPT

If you don't have a workflow YML file in your repository, create a file like ".github/workflows/blank.yml" at your project's root path.

SAST Example - The following example Github Actions YML script will scan the current repository source code, failing the job if medium or high vulnerabilities are identified. In addition to this, it attaches a PDF vulnerability report to the pipeline job artifacts.

```
name: syhunt test
on:
  push:
    branches:
      - main
jobs:
  build:
    runs-on: [self-hosted, Windows, X64]
    steps:
      - uses: actions/checkout@v2
      - run: Start-CodeScan -pfcond 'fail-if:risk=mediumup' -output 'report.pdf'
      - name: 'Syhunt Report'
        uses: actions/upload-artifact@v2
        if: failure()
        with:
         name: syhunt-report
         path: report.pdf
         if-no-files-found: error
```

DAST Example - The following example Github Actions 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 attaches a PDF vulnerability report to the pipeline job artifacts.

```
name: syhunt test
on:
  push:
    branches:
      - main
jobs:
  build:
    runs-on: [self-hosted, Windows, X64]
    steps:
      - uses: actions/checkout@v2
      - run: Start-DynamicScan -target 'www.productionurl.com' -pfcond 'fail-if:risk=mediumup' -output
      - name: 'Syhunt Report'
        uses: actions/upload-artifact@v2
        if: failure()
        with:
         name: syhunt-report
         path: report.pdf
         if-no-files-found: error
```

Additional examples:

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

# SAST Example - Scan a remote project repository
$MyProject = @{
  target = 'https://github.com/syhunt/vulnphp.git';
  branch = 'main';
  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
```

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.
- **tracker** (optional) - the name of previously created tracker or a dynamically generated tracker that will receive a summary of identified vulnerabilities at the end of the scan. Examples
- **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.
- **timelimit** (optional) - sets the maximum scan time limit (default: no limit). If the time is reached, the scan is aborted. Examples: 1d, 3h, 2h30m, 50m

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 project 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 git client will fetch the default branch.
- **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.
- **tracker** (optional) - the name of previously created tracker or a dynamically generated tracker that will receive a summary of identified vulnerabilities at the end of the scan. Examples
- **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.
- **timelimit** (optional) - sets the maximum scan time limit (default: no limit). If the time is reached, the scan is aborted. Examples: 1d, 3h, 2h30m, 50m

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:

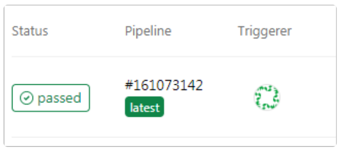
```
# Example 1 - Scan the current directory/repository
Start-CodeScan -pfcond "fail-if:risk=mediumup"

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

# Example 2 - Scan a remote Azure DevOps Services project
Start-CodeScan -target "https://dev.azure.com/user/projectname" -huntmethod "normal" -pfcond "fail-if:r

# Example 4 - Scan a specific local directory
Start-CodeScan -target "C:\www\" -huntmethod "normal" -pfcond "fail-if:risk=mediumup"
```

PASS/FAIL CONDITIONS



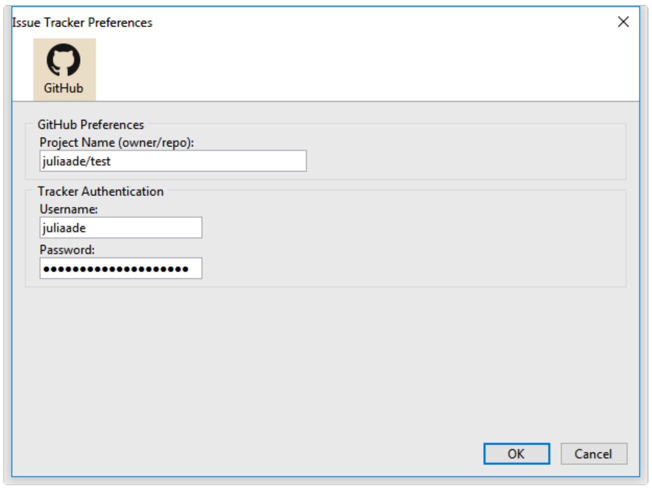
Status	Pipeline	Triggerer
passed	#161073142 latest	

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

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

INTEGRATING WITH GITHUB ISSUES

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


Issue Tracker Preferences

GitHub Preferences

Project Name (owner/repo): juliaade/test

Tracker Authentication

Username: juliaade

Password: ••••••••••••••••••••

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