

## Static Code Analysis

In this lab your task is simple, you have a list of vulnerable projects and you are required an automated static code analysis (SAST). Below you will a list of vulnerable applications/projects as well as a list of suggested SAST solutions. pick at least 2 applications/projects and scan them with one of the suggested SAST solution.

DVWA -> <https://github.com/digininja/DVWA>

DVNA -> <https://github.com/appsecco/dvna>

we will scan this 2 repository with with DeepSource and Snyk

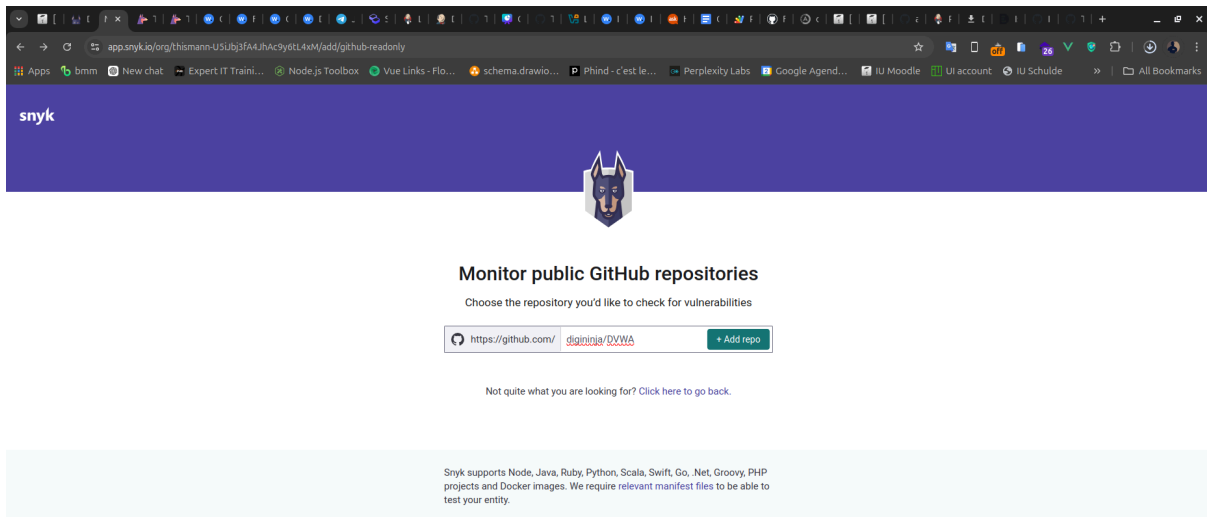
create a account to snyk and deepsource

- open the snyk.io
- sign-up to the snyk
- link the github account to snyk
- we can also monitorize external repository by add the respo name to the synk
- create the new project with snyk

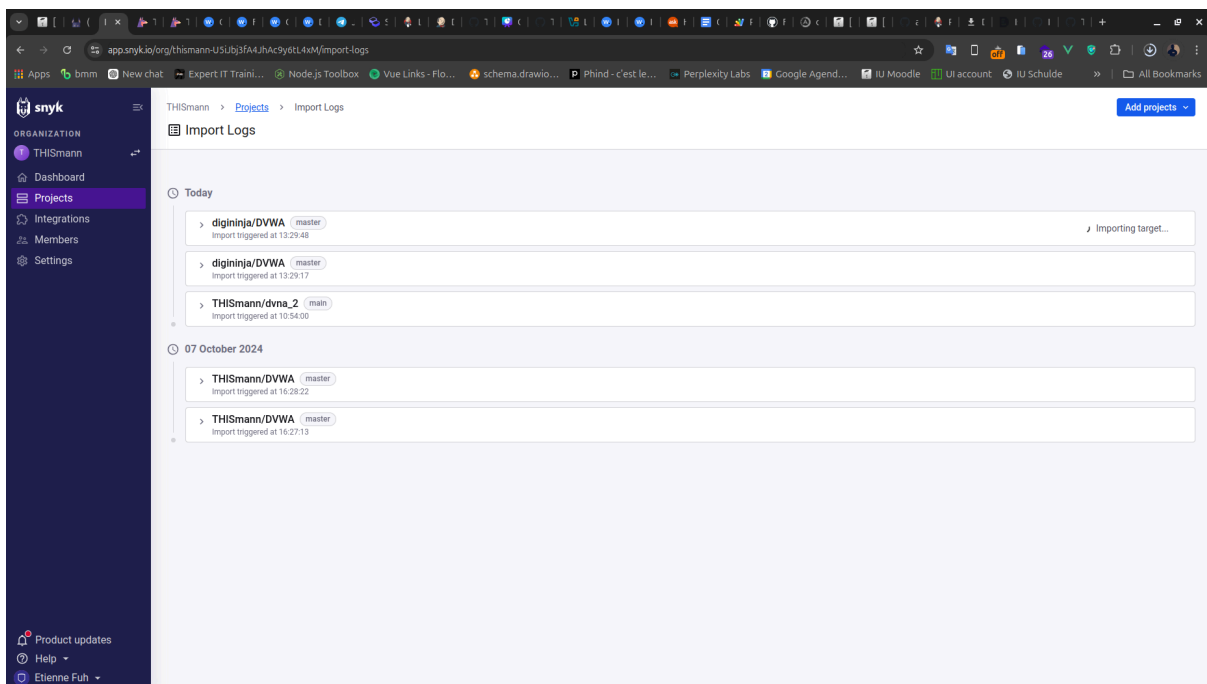
The screenshot displays the Snyk dashboard for the organization 'THISmann'. The left sidebar contains navigation links for Organization, Dashboard, Projects, Integrations, Members, and Settings. The main content area features a 'Security is a team sport' banner with a 'Copy invite link' button. Below this, the 'Top pending tasks' section shows two entries for 'THISmann/dvna\_2', each with a 'Fix vulnerabilities' button. The 'Top vulnerable projects' section lists four projects: 'THISmann/dvna\_2-Dockerfile', 'THISmann/dvna\_2-Dockerfile-dev', 'THISmann/dvna\_2-package.json', and 'digininja/DVWA'. Each project entry includes a 'Tested' timestamp, a table of issue counts (Critical, High, Medium, Low), and a 'Fix vulnerabilities' button. The browser's address bar shows the URL: <https://app.snyk.io/org/thismann-USUbj3FA4JhAc9y6tL4xM/add/github-readonly>.

Project	Tested	Critical	High	Medium	Low	Actions
THISmann/dvna_2-Dockerfile	3 hours ago	52	217	242	559	Fix vulnerabilities
THISmann/dvna_2-Dockerfile-dev	3 hours ago	52	217	242	559	Fix vulnerabilities
THISmann/dvna_2-package.json	3 hours ago	5	20	20	1	Fix vulnerabilities
digininja/DVWA	8 minutes ago	0	31	27	36	Fix vulnerabilities

- add a repository <https://github.com/digininja/DVWA> to snyk



- select you project and open the code analytics



- and now we have all the list of vulnerability detected

snyk

ORGANIZATION

THISmann

Dashboard

Projects

Integrations

Members

Settings

Product updates

Help

Etienne Fuh

Import triggered at 13:29:48

Project	Status
Snyk Code supported files	Project created <a href="#">View project</a>
Dockerfile	Project created <a href="#">View project</a>

> digininja/DVWA

master

Import triggered at 13:29:17

> THISmann/dvna\_2

main

Import triggered at 10:54:00

07 October 2024

> THISmann/DVWA

master

Import triggered at 16:28:22

Project	Status
Dockerfile	Project created <a href="#">View project</a>
Snyk Code supported files	Project created <a href="#">View project</a>

> THISmann/DVWA

master

Import triggered at 16:27:13

Project	Status
Dockerfile	Project created <a href="#">View project</a>
Snyk Code supported files	Project created <a href="#">View project</a>

snyk

ORGANIZATION

THISmann

Dashboard

Projects

Integrations

Members

Settings

Product updates

Help

Etienne Fuh

THISmann > Projects > THISmann/dvna\_2

[Open on GitHub](#)

Dockerfile

Overview History Settings

C

imagemagick/imagemagick-6-common - NULL Pointer Dereference

VULNERABILITY

CWE-476

CVE-2017-14626

CVSS 9.8

CRITICAL

SNYK-DEBIAN9-IMAGEMAGICK-400971

SCORE

714

Introduced through

imagemagick/imagemagick-6-common@8.6.9.7.4+dfsg-11+deb9u7, imagemagick/imagemagick-6.q16@8.6.9.7.4+dfsg-11+deb9u7 and others

Exploit maturity

NO KNOWN EXPLOIT

Fixed in

imagemagick/imagemagick-6-common@8.6.9.7.4+dfsg-11+deb9u10, @8.6.9.7.4+dfsg-11+deb9u10

Show more detail

Learn about this type of vulnerability

Ignore

C

imagemagick/imagemagick-6-common - NULL Pointer Dereference

VULNERABILITY

CWE-476

CVE-2017-14625

CVSS 9.8

CRITICAL

SNYK-DEBIAN9-IMAGEMAGICK-400978

SCORE

714

Introduced through

imagemagick/imagemagick-6-common@8.6.9.7.4+dfsg-11+deb9u7, imagemagick/imagemagick-6.q16@8.6.9.7.4+dfsg-11+deb9u7 and others

Exploit maturity

NO KNOWN EXPLOIT

Fixed in

imagemagick/imagemagick-6-common@8.6.9.7.4+dfsg-11+deb9u10, @8.6.9.7.4+dfsg-11+deb9u10

Show more detail

Learn about this type of vulnerability

Ignore

snky

ORGANIZATION

THISmann

Dashboard

Projects

Integrations

Members

Settings

Product updates

Help

Etienne Fuh

THISmann > Projects > THISmann/dvna\_2 (main)

Dockerfile

OverviewHistorySettings

C

imagemagick/imagemagick-6-common - Out-of-bounds Write

SCORE 714

VULNERABILITY

CWE-787 + 1 MORE

CVE-2018-14551

CVSS 9.8

CRITICAL

SNYK-DEBIAN9-IMAGEMAGICK-402497

Introduced through

imagemagick/imagemagick-6-common@8.6.9.7.4+dfsg-11+deb9u7, imagemagick/imagemagick-6-q16@8.6.9.7.4+dfsg-11+deb9u7 and others

Exploit maturity

NO KNOWN EXPLOIT

Fixed in

imagemagick/imagemagick-6-common@8.6.9.7.4+dfsg-11+deb9u9, @8.6.9.7.4+dfsg-11+deb9u9

Show more detail

Ignore

C

libexif/libexif-dev - Integer Overflow or Wraparound

SCORE 714

VULNERABILITY

CWE-190

CVE-2020-0452

CVSS 9.8

CRITICAL

SNYK-DEBIAN9-LIBEXIF-1037315

Introduced through

libexif/libexif-dev@0.6.21-2+b2 and libexif/libexif12@0.6.21-2+b2

Exploit maturity

NO KNOWN EXPLOIT

Fixed in

libexif/libexif-dev@0.6.21-2+deb9u5, @0.6.21-2+deb9u5

Show more detail

Ignore

<< < 1 2 3 4 5 - > >>

94 of 94 issues

Group by none Sort by highest severity

H

SQL Injection

SCORE 810

SNYK CODE

CWE-89

```
9 | $pass = md5( $pass );
10 |
11 | // Check the database
12 | $query = "SELECT * FROM `users` WHERE user = '$user' AND password = '$pass'";
13 | $result = mysqli_query($GLOBALS["__mysqli_ston"], $query ) or die( '<pre>' . ((is_object($GLOBALS["__mysqli_ston"])) ? mysqli_error($
14 |
```

Unsanitized input from an HTTP parameter flows into `mysqli_query`, where it is used in an SQL query. This may result in an SQL Injection vulnerability.  
[vulnerabilities/brute/source/low.php](#) 5 steps in 1 file

Learn about this type of vulnerability and how to fix it

Ignore Full details

H

SQL Injection


SCORE 810

SNYK CODE

CWE-89

```
7 | switch ($DVWA['SQLI_DB']) {
8 |     case MYSQL:
9 |         // Check database
10 |         $query = "SELECT first name, last name FROM users WHERE user id = '$id'";
```

now let run the scan on the second resp <https://github.com/appsecco/dvna>



## Monitor public GitHub repositories

Choose the repository you'd like to check for vulnerabilities

+ Add repo

appsecco/dvna

✓ Supported manifest file found

Import 1 repository

Not quite what you are looking for? [Click here to go back.](#)

- and we can select the code analytics

Targets 4

> appsecco/dvna

109 C 459 H 531 M 1.1k L ...

> THISmann/dvna\_2

109 C 459 H 531 M 1.1k L ...

▼ digininja/DVWA

0 C 31 H 27 M 36 L ...

Project

Imported

Tested

Issues ↓

<input type="checkbox"/> Code analysis	2 hours ago	2 hours ago	0 C 31 H 27 M 36 L ...
<input type="checkbox"/> Dockerfile	2 hours ago	2 hours ago	0 C 0 H 0 M 0 L ...

> THISmann/DVWA

0 C 31 H 27 M 36 L ...

Ready to import another project?

Secure your entire stack with Snyk

- now we have a list of differents vulnerabilities found

Code Analysis

OverviewHistorySettings

SEVERITY

High31

Medium27

Low36

PRIORITY SCORE

Scored between 0 - 1000

STATUS

Open94

Ignored0

LANGUAGES

PHP94

VULNERABILITY TYPES

Use of Password Hash...23

Information Exposure22

SQL Injection9

Path Traversal8

Command Injection6

Sensitive Cookie Withou...5

94 of 94 issues

Group by noneSort by highest severity

H

SQL Injection

SNYK CODE | CWE-89

SCORE810

```
9 | case MYSQL:
10 | // Check database
11 | $query = "SELECT first_name, last_name FROM users WHERE user_id = '$id' LIMIT 1;";
12 | try {
13 |     $result = mysqli_query($GLOBALS["__mysqli_ston"], $query ); // Removed 'or die' to suppress mysql errors
14 | }
```

Unsanitized input from cookies flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.  
vulnerabilities/sqli\_blind/source/high.php5 steps in 1 file

Learn about this type of vulnerability and how to fix it

IgnoreFull details

H

SQL Injection

SNYK CODE | CWE-89

SCORE810

```
31 | global $sqlite_db_connection;
32 |
33 | $query = "SELECT first_name, last_name FROM users WHERE user_id = '$id' LIMIT 1;";
34 | try {
```

Code Analysis

OverviewHistorySettings

H

Path Traversal

SNYK CODE | CWE-23

SCORE809

```
56 | default:
57 |     $vuln = "Unknown Vulnerability";
58 | }
59 |
60 | $source = @file_get_contents( DVWA_WEB_PAGE_TO_ROOT . "vulnerabilities/{$id}/source/{$security}.php" );
61 |
```

Unsanitized input from an HTTP parameter flows into file\_get\_contents, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to read arbitrary files.  
vulnerabilities/view\_source.php6 steps in 1 file

Learn about this type of vulnerability and how to fix it

IgnoreFull details

H

Path Traversal

SNYK CODE | CWE-23

SCORE809

```
61 | $source = str_replace( array( 'html .' ), array( 'echo' ), $source );
62 |
63 | $js_html = "";
64 | if (file_exists( DVWA_WEB_PAGE_TO_ROOT . "vulnerabilities/{$id}/source/{$security}.js" )) {
65 |     $js_source = @file_get_contents( DVWA_WEB_PAGE_TO_ROOT . "vulnerabilities/{$id}/source/{$security}.js" );
66 | }
```

Code Analysis

OverviewHistorySettings

Use of Password Hash...

Information Exposure22

SQL Injection9

Path Traversal8

Command Injection6

Sensitive Cookie Withou...5

Sensitive Cookie in HTT...5

Cross-site Scripting (XSS)4

Open Redirect4

Hardcoded Secret3

Use of Hardcoded Crede...2

Code Injection2

Use of a Broken or Risky...1

M

Information Exposure

SNYK CODECWE-200

SCORE374

```
57         $pass_new = md5( $pass_new );
58
59         // Update database
60         $insert = "UPDATE `users` SET password = '$pass_new' WHERE user = '" . $dwaCurrentUser() . "'";
61         $result = mysqli_query($GLOBALS["__mysqli_ston"], $insert ) or die( '
```

An MySQL error object flows to die/exit and is leaked to the attacker. This may disclose important information about the application to an attacker.

[vulnerabilities/captcha/source/low.php](#)

4 steps in 1 file

Ignore

Full details

M

Information Exposure

SNYK CODECWE-200

SCORE374

```
11     $pass = md5( $pass );
12
13     // Check the database
14     $query = "SELECT * FROM `users` WHERE user = '$user' AND password = '$pass'";
15     $result = mysqli_query($GLOBALS["__mysqli_ston"], $query ) or die( '
```

An MySQL error object flows to die/exit and is leaked to the attacker. This may disclose important information about the application to an attacker.

[vulnerabilities/brute/source/medium.php](#)

4 steps in 1 file

Code Analysis

OverviewHistorySettings

Use of Password Hash...

Information Exposure22

SQL Injection9

Path Traversal8

Command Injection6

Sensitive Cookie Withou...5

Sensitive Cookie in HTT...5

Cross-site Scripting (XSS)4

Open Redirect4

Hardcoded Secret3

Use of Hardcoded Crede...2

Code Injection2

Use of a Broken or Risky...1

L

Use of Password Hash With Insufficient Computational Effort

SNYK CODECWE-916

SCORE424

```
12     // Sanitise password input
13     $pass = $_GET[ 'password' ];
14     $pass = stripslashes( $pass );
15     $pass = ((isset($GLOBALS["__mysqli_ston"]) && is_object($GLOBALS["__mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["__mysqli_stc
16     $pass = md5( $pass );
17
```

MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

[vulnerabilities/brute/source/high.php](#)

1 step in 1 file

Learn about this type of vulnerability and how to fix it

Ignore

Full details

L

Use of Password Hash With Insufficient Computational Effort

SNYK CODECWE-916

SCORE424

```
53     // Check to see if both password match
54     if( $pass_new == $pass_conf ) {
55         // They do!
56         $pass_new = ((isset($GLOBALS["__mysqli_ston"]) && is_object($GLOBALS["__mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["my
57         $pass_new = md5( $pass_new );
58
```

MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.