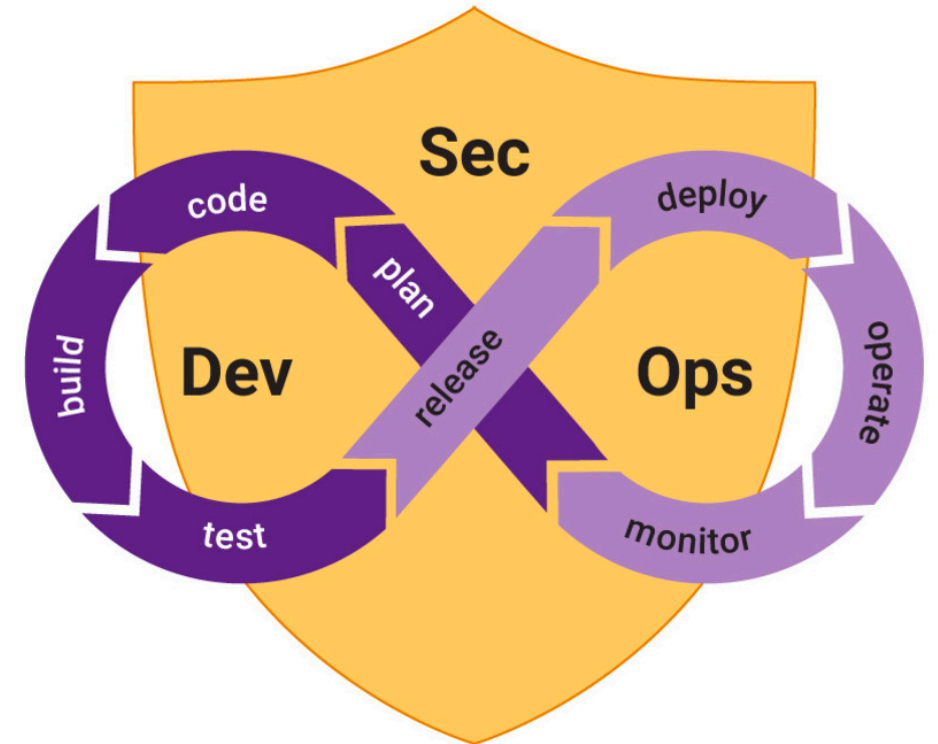


DevSecOps in Mobile Sec

DevSecOps

DevSecOps aims to implement Security as a part of the build process by integrating **Security tools as part of the Build process**, **Creating Security as a Culture** in the organization and **Automating as many processes** as possible



Source: <https://www.synopsys.com/glossary/what-is-devsecops.html>

Security in SDLC

Pre-Commit Hooks: Detect any hard coded passwords, API keys etc in the source code during the commit itself. For e.g *Talisman, git-secret*

IDE Plugins: Integrate Plugins to perform security analysis within the IDE itself, for e.g *Address Sanitizer in Xcode, SonarLint, FindBugs* etc

Secrets Management: Manage credentials such that can be integrated into the development toolset, for e.g *Ansible Vault*

Source Code Composition Analysis: Identify any vulnerable open-source libraries being use, for e.g *Owasp dependency checker*

Static Application Security Testing: Analyze source code or compiled version to find security flaws, for e.g *SonarQube*

Dynamic Application Security Testing: Scan the application in a running environment

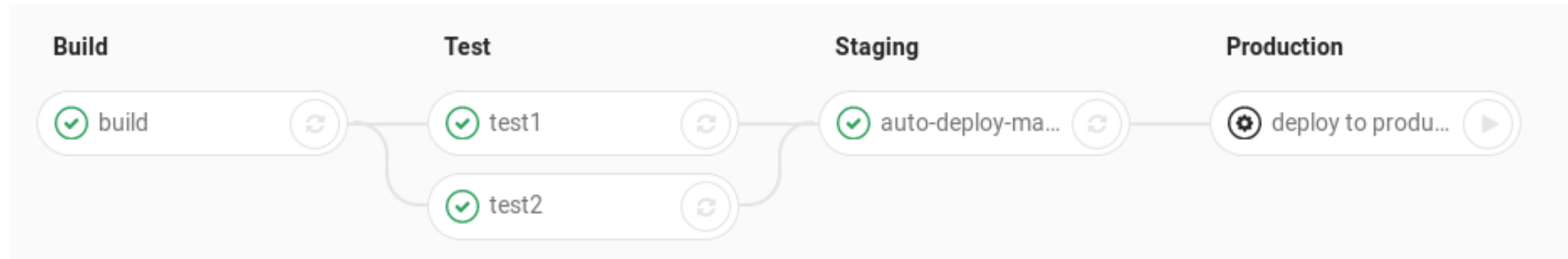
Container Scanning: Scan Docker images for vulnerabilities

Security in SDLC

1. Tools should support **CLI**
2. Preferably should have a docker image
3. Should finish execution in 10 minutes
4. Much more customizable (remove false positives, exclude files etc)
5. Should support Parsable output (XML, JSON etc)

Gitlab Pipelines

A pipeline is a group of **jobs** that get executed in stages. All of the **jobs** in a stage are executed in parallel (if there are enough concurrent Runners), and if they all succeed, the pipeline moves on to the next stage. If one of the **jobs** fails, the next stage is not (usually) executed. <https://forge.etsi.org/rep/help/ci/pipelines.md>



In Gitlab, the pipeline is put inside a file named **.gitlab-ci.yml** in the main repository

Sample CI/CD pipelines: <https://docs.gitlab.com/ee/ci/examples/>

.gitlab-ci.yml keywords

image: Use a specific docker image e.g image: python:3.6

before_script: Override a set of commands that are executed before job

services: Use Docker service images

allow_failure: Allow a job to fail. A failed job does not cause the pipeline to fail

artifacts: List of files and directories to attach to a job on success

when: when to run job (for e.g manual, delayed)

only: Run only on that specific branch

Some keywords can be set globally as the default for all jobs with the `default:` keyword. Default keywords can then be overridden by job-specific configuration.

Source: <https://docs.gitlab.com/ee/ci/yaml/>

Sample .gitlab-ci.yml (C++)

C++ .gitlab-ci.yml 702 Bytes

Edit

Web IDE

Lock

```
1 # use the official gcc image, based on debian
2 # can use versions as well, like gcc:5.2
3 # see https://hub.docker.com/_/gcc/
4 image: gcc
5
6 build:
7   stage: build
8   # instead of calling g++ directly you can also use some build toolkit like make
9   # install the necessary build tools when needed
10  # before_script:
11  #   - apt update && apt -y install make autoconf
12  script:
13    - g++ helloworld.cpp -o mybinary
14  artifacts:
15    paths:
16      - mybinary
17      # depending on your build setup it's most likely a good idea to cache outputs to reduce the build time
18      # cache:
19      #   paths:
20      #     - "*.o"
21
22 # run tests using the binary built before
23 test:
24   stage: test
25   script:
26     - ./runmytests.sh
```

Secrets scanning example - Trufflehog

Scan through git repositories for secrets, looking into commit history and branches. For e.g **trufflehog**

```
prateek:: trufflehog --regex --entropy=False https://github.com/dxa4481/trufflehog.git
~~~~~
Reason: Generic Secret
Date: 2021-01-31 00:20:09
Hash: 288f35ed2e73145e3578207e896a4cf1600bea45
Filepath: test_all.py
Branch: origin/dev
Commit: Fixed since_commit and tests (#219)

* Fixed since_commit and tests

* Updated testing comment

Co-authored-by: Dylan Ayrey <dxa4481@rit.edu>
secret = '9ed54617547cfca783e0f81f8dc5c927e3d1e345'
```

Secrets scanning example - gitrob

Can be used to scan an organization, and its members public repos

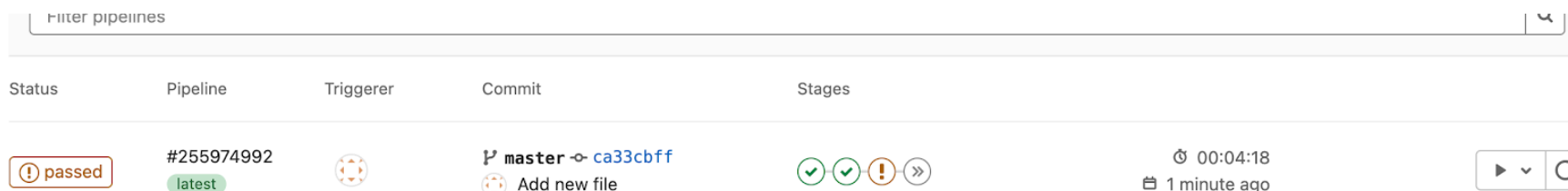
<https://github.com/michenriksen/gitrob>

```
prateek@kali:~$ gitrob samsung  
  
  _ _ _  
  ( ) / _ _ _ _ _ _ _ _ / /  
 / _ \ / / _ _ \ _ _ \ _ \ \  
 \ , / _ ^ _ \ / \ _ _ \ . _ \  
 / _ _ \ by @michenriksen  
  
gitrob v2.0.0-beta started at 2021-03-11T02:52:31-05:00  
Loaded 91 signatures  
Web interface available at http://127.0.0.1:9393  
Gathering targets...  
Retrieved 120 repositories from Samsung  
Retrieved 3 repositories from ajkxyz  
Retrieved 18 repositories from arthur-flam  
Retrieved 2 repositories from clamp03
```

Task 5 mins: Create your project for DevSecOps

We will be using the vulnerable python application Django.nv for our labs today <https://github.com/nVisium/django.nv>

1. On Gitlab, **Create a New Project -> Import Project -> Repo by Url -> Paste <https://github.com/nVisium/django.nv>** , Make sure it's a private project, Rename the project **Djangonv-devsecops** and click **Create**
2. In the repo, add a **new File** by Clicking on **+** -> **New File**
3. Name it **.gitlab-ci.yml**
4. Paste the code from the file **1.txt** under **Day 2 -> DevSecOps**
5. Click **Commit Changes**
6. On the Left Section go to **CI/CD -> Pipelines** and watch as all the stages complete execution

































The screenshot shows the GitLab Pipelines interface. At the top, there is a search bar labeled "Filter pipelines". Below it, a table displays pipeline information. The table has columns for Status, Pipeline, Triggerer, Commit, Stages, and a final column with controls. The first row shows a pipeline with a "passed" status, ID "#255974992", triggerer "latest", commit "master -> ca33cbff" with the message "Add new file", and stages represented by three icons (two green checkmarks and one red exclamation mark). The pipeline duration is "00:04:18" and it was completed "1 minute ago".

Status	Pipeline	Triggerer	Commit	Stages	
passed	#255974992 latest		master -> ca33cbff Add new file		00:04:18 1 minute ago

Task 15 mins: Adding SAST, SCA and DAST

1. Edit `.gitlab-ci.yml`
2. Paste the code from the following URL **Day 2 -> DevSecOps -> 2.txt** and **Commit Changes**
3. Notice the addition of **dast** job
4. Go to **CI/CD -> Pipelines** and check the results once pipeline finishes execution
5. Download the artifacts file and analyze the output of the tool

Pipeline	Triggerer	Commit	Stages	Duration	Actions
#255982804 latest		 master  6a4c0312  Update .gitlab-ci.yml	    	00:06:24 just now	  
#255979838		 master  7894071e  Update .gitlab-ci.yml	   	00:04:19 22 minutes ago	
#255977366		 master  60899b7d  Update .gitlab-ci.yml	   	00:04:25 34 minutes ago	

- Download zap-baseline:archive artifact
- Download oast-frontend:archive artifact
- Download oast:archive artifact
- Download sast:archive artifact

Task 10 mins

1. What is a ZAP baseline scan in DevSecOps ?
2. How do you ensure some jobs run only when changes are committed to master and not any other branch ?
3. In the same pipeline, add the following tools in the integration stage (target: **demo.testfire.net**)
 1. Nikto for Web Server Scanning: <https://github.com/sullo/nikto>
 2. SSLyze for SSL/certificate related issues <https://github.com/nabla-c0d3/sslyze>
 3. Nmap for port Scanning <https://nmap.org/book/port-scanning-tutorial.html>

Solution: **Day 2 -> DevSecOps -> 3.txt**

GitLab DevSecOps

How GitLab enables DevSecOps

- **Every piece of code** is tested upon commit for security threats, without incremental cost.
- **The developer** can remediate now, while they are still working in that code, or create an issue with one click.
- **The security pro** can see and manage unresolved vulnerabilities captured as a by-product of software development.
- **Single source of truth** can focus collaboration on remediation, eliminating translation and finger pointing.
- **A single tool** reduces cost to buy, integrate and maintain point solutions throughout the DevOps pipeline.



<https://about.gitlab.com/solutions/dev-sec-ops/>
<https://t.me/learningnets>

Gitlab DevSecOps

docs.gitlab.com/ee/user/application_security/

GitLab Docs Search the docs...

GitLab Docs > User Docs > Application security

Application security ULTIMATE

GitLab can check your application for security vulnerabilities that may lead to unauthorized access, data leaks, denial of services, and more. GitLab reports vulnerabilities in the merge request so you can fix them before you merge.

- The [Security Dashboard](#) provides a high-level view of vulnerabilities detected in your projects, pipeline, and groups.
- The [Threat Monitoring](#) page provides runtime security metrics for application environments. With the information provided, you can immediately begin risk analysis and remediation.

For an overview of GitLab application security, see [Security Deep Dive](#).

Quick start

Get started quickly with Dependency Scanning, License Scanning, Static Application Security Testing (SAST), and Secret Detection by adding the following to your `.gitlab-ci.yml`:

```
include:
  - template: Security/Dependency-Scanning.gitlab-ci.yml
  - template: Security/License-Scanning.gitlab-ci.yml
  - template: Security/SAST.gitlab-ci.yml
  - template: Security/Secret-Detection.gitlab-ci.yml
```

To add Dynamic Application Security Testing (DAST) scanning, add the following to your `.gitlab-ci.yml` and

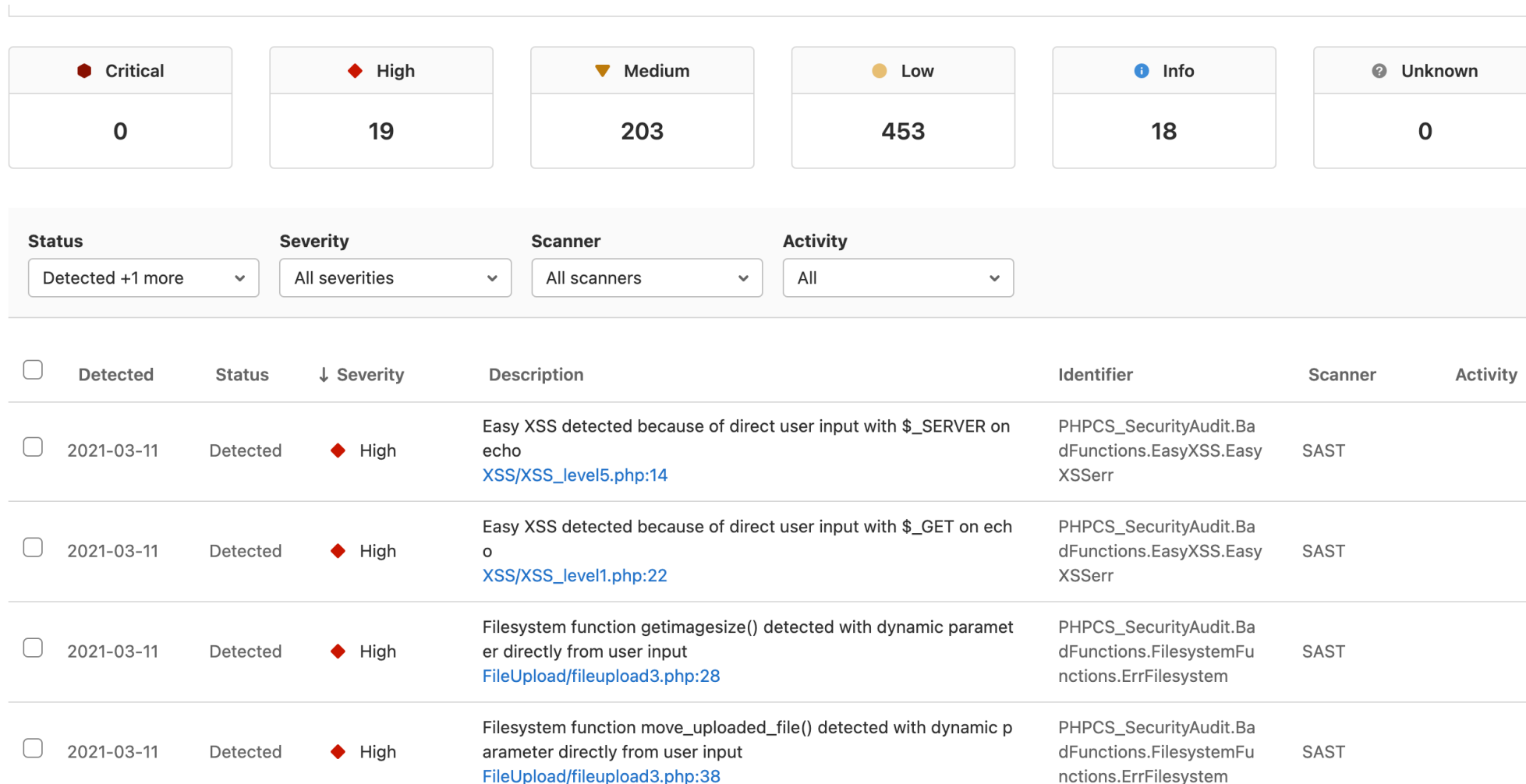
https://docs.gitlab.com/ee/user/application_security/

<https://t.me/learningnets>

Task 10 mins: Adding Gitlab's DevSecOps to CI/CD

1. On Gitlab, **Create a New Project -> Import Project -> Repo by Url -> <https://github.com/OWASP/Vulnerable-Web-Application>** Rename the project **Owasp-VWA**, make sure it's a **Public Project** and click **Create**
2. In the repo, add a new File
3. Name it **.gitlab-ci.yml**
4. Paste the code from the file **4.txt** under **Day 2 -> DevSecOps**
5. Click **Commit Changes**
6. On the Left Section go to **CI/CD -> Pipelines** and watch as all the stages complete execution
7. Once finished, on the left, go to **Security & Compliance -> Vulnerability Report**

Task 10 mins: Adding Gitlab's DevSecOps to CI/CD



References

<https://www.practical-devsecops.com/>

<https://medium.com/faun/gitlab-and-devsecops-pipeline-implementation-part-i-3e03e2e2a919>

<https://www.youtube.com/watch?v=NScuPB5uwg4>

<https://about.gitlab.com/blog/2019/06/20/announcing-gitlab-devsecops/>

-