Faraday

Integrating Faraday in the software development process

Part Three Travis CI

This white paper produced by our technical team, shares important information to attack vulnerabilities from the first stage when developing software.

About Us

Faraday's mission is to **make security simple and accessible to everyone**, using our experience and passion to enable SMB/SME companies reducing their gap between exposure and remediation.

We truly believe that a clear understanding of your security posture is the primary key to reduce your attack surface, allowing you to make smarter decisions to protect your most valuable assets.

Security is a world-class engineering challenge and we want to help. We are a passionate leading team that wants to transform the way security works.

Outstanding research results published

Constant contribution to the global security community

+15 years working with F500

Speakers at the best security conferences

+60 employees worldwide

Get to know us at **faradaysec.com**

Integrating Faraday in the software development process - Part Three

Introduction

From a while we are explaining to you how to integrate our Faraday instance to our development process of Python applications integrated with **Heroku** easily. In our last posts, we did this using **Github Actions** and **Jenkins Pipelines**.

Today we are going to implement this using **Travis CI** as a CD/CI tool because this is one with good market share.

Previous considerations

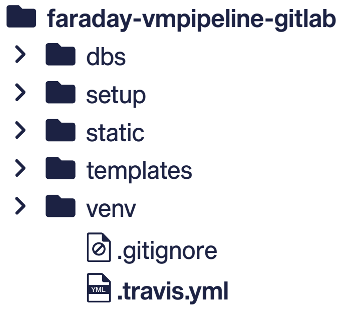
As we did in the last post, we are going to assume that you already have a python repository in some git server that already is setup using **Heroku**. You can read our first post if you want to know how to set this and get more context.

Travis CI is a CD/CI tool that has two versions: The paid version allows us to use private git repositories and the free version allows us only to use public repositories.

We are not going to go deep into the config of each version because each one has a different way to configure it but it is not hard, just read the official documentation. We are going to focus on the **.travis.yml** file and how must be configured to integrate it with our **Faraday** instance.

1. Creating the .travis.ylm file

The first step is to create a file called **.travis.yml** in the root of our repository, just as we did with the **Jenkinsfile** file in our last post. This file will be read by Travis on each push and will allow us to build, test, deploy our application and upload the result to our **Faraday** instance.



```
faraday-vmpipeline-gitlab
  > dbs
  > setup
  > static
  > templates
  > venv
      .gitignore
      .travis.yml
```

Once the file is created, we need to write the following in it:

```
language: python
services:
 - docker
python:
  - ‘3.9’
env:
  - PROJECT=faraday-vmpipelines
before_install
  - docker pull owasp/zap2docker-stable
  - docker build https://github.com/flopezluksenberg/docker-faraday-
report-uploader.git#bandit -t faraday-uploader
script:
  -
```

Let’s explain what means each line written in previous code block:

- The first line is pretty clear: this is telling to **Travis** that we are going to build an application written in Python.

- The **services** section allows us to tell to **Travis** that our execution flow will require some external **service** while it is executing our task like **mongodb**, **redis**, etc. In our case we are going to use **docker** because we need it to run zap later.

- Under **python** section we are going to set the python version. In this case the version is **3.9**.
- The **env** section allows us to specify environment variables that we need to use during execution. In our case we are going to create the **PROJECT** variable because we need it to build the workspace name in our **Faraday** instance later.

- In the **before_install** section we tell to **Travis** what need to do before install our dependencies. In our case we are using this section to pre-setup the docker images that we will use later.

- The **script** section is the most important in this file. Here we are going to write each step to be executed by **Travis** sequentially. You will notice that is pretty similar as we did in our last posts using **Github Actions** and **Jenkins Pipelines**.

*Note: All sections before **script** are necessary for general setup. In our case the dependencies of our repository will be installed automatically because we said to Travis that the language to be used is python so **Travis** will find the **requirements.txt** file to install the required dependencies before execute the **script** section.*

2. Defining the script section

As we said, the **script** section is the most important because here is where **Travis** will execute each line of our building process sequentially. Hopefully All the environment were set up in the previous sections so we can only focus on the execution sequence.

As we did in the previous post, we will run **bandit** over our code, we will deploy our app in **Heroku**, we will run a remote scan using **Zap** over our just deployed app and we will finish this process uploading the report status to our **Faraday** instance.

```
 language: python
services:
 - docker
python:
  - ‘3.9’
env:
  - PROJECT=faraday-vmpipelines
before_install
  - docker pull owasp/zap2docker-stable
  - docker build https://github.com/flopezluksenberg/docker-far-
aday-report-uploader.git#bandit -t faraday-uploader
```

```
script:
  - bandit -r $TRAVIS_BUILD_DIR -f xml -o $TRAVIS_BUILD_
DIR/flaskapp_faraday_bandit.xml --exit-zero
  - git remote rm heroku || true
  - git remote add heroku
    https://heroku:$HEROKU_API_KEY@git.heroku.com/$HEROKU_APP_NAME.git > /dev/null
  - git push heroku HEAD:master -f
  - export CURRENT_DATE=$(date +'%Y-%m-%d')
  - docker run -u root -v $TRAVIS_BUILD_DIR:/zap/wrk/:rw --network=host -t owasp/
zap2docker-stable zap-baseline.py -t $ZAP_SCAN_URL -x flaskapp_faraday_zap.xml || true
  - docker run --name faraday-uploader --rm -v $TRAVIS_BUILD_DIR:
$TRAVIS_BUILD_DIR -e HOST=$FARADAY_URL -e USERNAME=$FARADAY_USERNAME
-e PASSWORD=$FARADAY_PASSWORD -e WORKSPACE=$PROJECT-$CURRENT_DATE-$TRAVIS_BUILD_NUMBER
-e FILES=$TRAVIS_BUILD_DIR/flaskapp_faraday_bandit.xml faraday-uploader
  - docker run --name faraday-uploader --rm -v $TRAVIS_BUILD_DIR:
$TRAVIS_BUILD_DIR -e HOST=$FARADAY_URL -e USERNAME=$FARADAY_USERNAME
-e PASSWORD=$FARADAY_PASSWORD -e WORKSPACE=$PROJECT-$CURRENT_DATE-$TRAVIS_BUILD_
NUMBER -e FILES=$TRAVIS_BUILD_DIR/flaskapp_faraday_zap.xml faraday-uploader
```

Let's read step by step of previous code block to understand what we did on the **script** section:

- In the first line we are running **bandit** over our repository. This is not different in comparison with the previous posts.
- The following three steps are used to setup and deploy our app to **Heroku**. Again, we did it equally as we did in our previous posts. Here is important to notice the custom environment variables called **$HEROKU_API_KEY and $HEROKU_APP_NAME**. We will explain later how to declare custom variables in **Travis**.
- Then we create a variable to store the current date because we will need later to create the **workspace** in our **Faraday** instance. The variable name is **$CURRENT_DATE**.
- Then we use **docker**. The first docker run is to scan and identify vulns on our deployed app using **Zap**. Please notice that we are using another custom environment variable called **$ZAP_SCAN_URL**.
- When the previous scan finishes we just need to upload both generated reports to our **Faraday** instance. This is not different in comparison as we did in the previous posts because **docker** let us do it easily. Notice that we have used two environment variables provided by **Travis** and some custom environment variables too. The **Travis** environment variables are the following:
    - **$TRAVIS_BUILD_DIR**: This is pretty clear, it is the build dir that **Travis** is using to run the task

- **$TRAVIS_BUILD_NUMBER**: This variable tell us what running number is the current one.

The custom environment variables are the following:

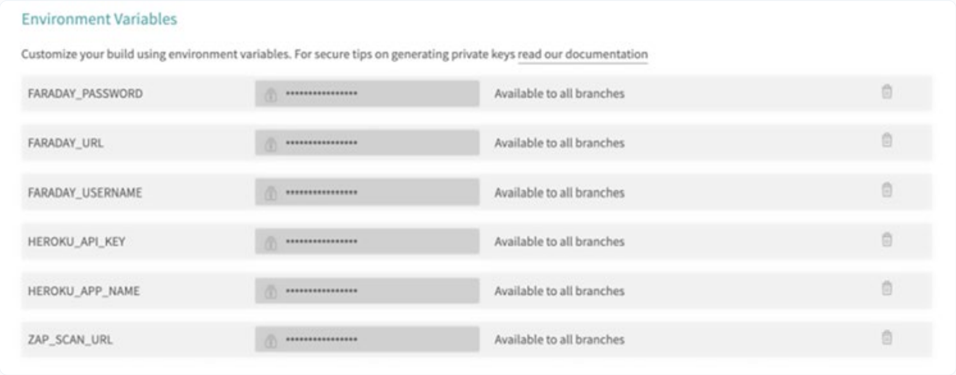
- **$FARADAY_URL**: This is our **Faraday** instance url.
- **$FARADAY_USERNAME**: Faraday username that will upload the reports
- **$FARADAY_PASSWORD**: Faraday password of the previous username that will upload the reports
- **$PROJECT**: We have defined this environment variable in the **env** section previously. This is used to create the workspace name.
- **$CURRENT_DATE**: We have created this variable previously in the **script** section. This returns the day with the following format yyyy-MM-dd.

Note: We are not going to go deep here because we've cover a lot in the previous posts. If you can't follow this post please go to read the previous ones to get more context.

3. Creating the custom environment variables in Travis

We already set up the **.travis.yml** file but to get a successful build we still need to set up the custom environment variables that we have used previously.

To do this, we need to go to the settings of our repository through the **Travis** website (doesn't matter what version you have). Once inside this, you need to find the **Environment Variables** section and add all the variables that we used previously one by one.



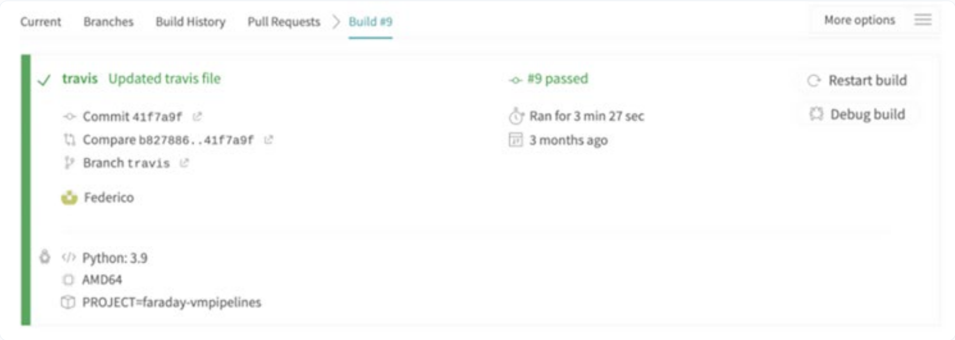
Environment Variables

Customize your build using environment variables. For secure tips on generating private keys read our documentation

FARADAY_PASSWORD	••••••••••••••••	Available to all branches
FARADAY_URL	••••••••••••••••	Available to all branches
FARADAY_USERNAME	••••••••••••••••	Available to all branches
HEROKU_API_KEY	••••••••••••••••	Available to all branches
HEROKU_APP_NAME	••••••••••••••••	Available to all branches
ZAP_SCAN_URL	••••••••••••••••	Available to all branches

The **Travis** UI is pretty simple so we are not going to go deep here. It is important to notice that you can set variable values by branch, so you could set up different **Faraday** instances based on the branch.

When you finish it you can see your build in **Travis**. If all went ok, you will see something like the following:



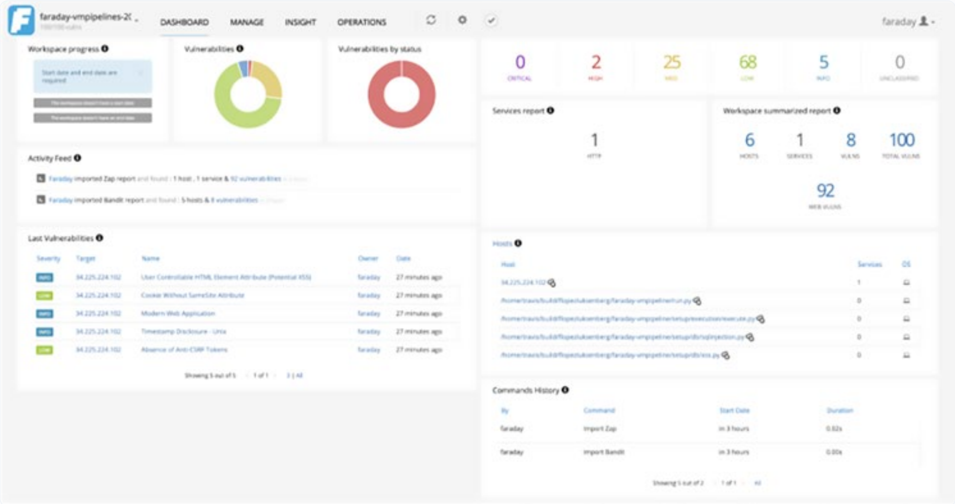
✓ travis Updated travis file

Commit 41f7a9f
Compare b827886..41f7a9f
Branch travis
Federico

#9 passed
Ran for 3 min 27 sec
3 months ago

Python: 3.9
AMD64
PROJECT=faraday-vmpipelines

And that ‘s all. Now you can go to your **Faraday** instance and check the new workspace as we did in the previous posts.



Activity Feed

Faraday imported Zap report and found : 1 host , 1 service & 92 vulnerabilities
Faraday imported Bandit report and found : 5 hosts & 8 vulnerabilities

Last Vulnerabilities

Severity	Target	Name	Owner	Date
INFO	34.225.224.102	User Controllable HTML Element Attribute (Potential XSS)	faraday	27 minutes ago
LOW	34.225.224.102	Cookie Without SameSite Attribute	faraday	27 minutes ago
INFO	34.225.224.102	Modern Web Application	faraday	27 minutes ago
INFO	34.225.224.102	Timestamp Disclosure - Unix	faraday	27 minutes ago
LOW	34.225.224.102	Absence of Anti-CSRF Tokens	faraday	27 minutes ago

Showing 5 out of 5 1 of 1

Hosts

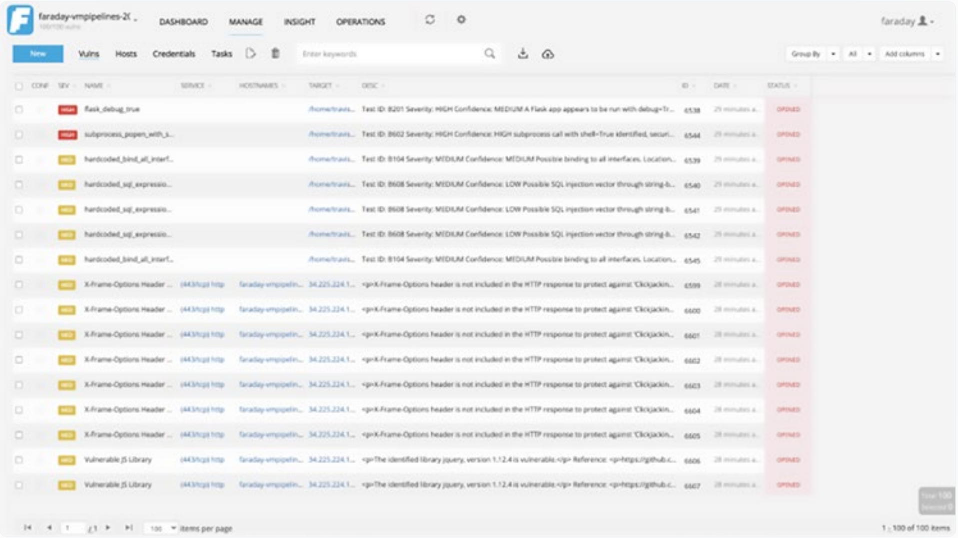
Host	Services	OS
34.225.224.102	1	
[illegible]run.py	0	
[illegible]execute.py	0	
[illegible]sqlinjection.py	0	
[illegible]xss.py	0	

Commands History

By	Command	Start Date	Duration
faraday	import Zap	in 3 hours	0.02s
faraday	import Bandit	in 3 hours	0.00s

Showing 5 out of 2 1 of 1

If you enter to the Status Report section you can see the detected vulns:



☐	CONF	SEV	NAME	SERVICE	HOSTNAMES	TARGET	DESC	ID	DATE	STATUS
☐		HIGH	flask_debug_true			/home/travis...	Test ID: B201 Severity: HIGH Confidence: MEDIUM A Flask app appears to be run with debug=Tr...	6538	29 minutes a...	OPENED
☐		HIGH	subprocess_popen_with_s...			/home/travis...	Test ID: B602 Severity: HIGH Confidence: HIGH subprocess call with shell=True identified, securi...	6544	29 minutes a...	OPENED
☐		MED	hardcoded_bind_all_interf...			/home/travis...	Test ID: B104 Severity: MEDIUM Confidence: MEDIUM Possible binding to all interfaces. Location...	6539	29 minutes a...	OPENED
☐		MED	hardcoded_sql_expressio...			/home/travis...	Test ID: B608 Severity: MEDIUM Confidence: LOW Possible SQL injection vector through string-b...	6540	29 minutes a...	OPENED
☐		MED	hardcoded_sql_expressio...			/home/travis...	Test ID: B608 Severity: MEDIUM Confidence: LOW Possible SQL injection vector through string-b...	6541	29 minutes a...	OPENED
☐		MED	hardcoded_sql_expressio...			/home/travis...	Test ID: B608 Severity: MEDIUM Confidence: LOW Possible SQL injection vector through string-b...	6542	29 minutes a...	OPENED
☐		MED	hardcoded_bind_all_interf...			/home/travis...	Test ID: B104 Severity: MEDIUM Confidence: MEDIUM Possible binding to all interfaces. Location...	6545	29 minutes a...	OPENED
☐		MED	X-Frame-Options Header ...	(443/tcp) http	faraday-vmpipelin...	34.225.224.1...	<p>X-Frame-Options header is not included in the HTTP response to protect against 'Clickjackin...	6599	28 minutes a...	OPENED
☐		MED	X-Frame-Options Header ...	(443/tcp) http	faraday-vmpipelin...	34.225.224.1...	<p>X-Frame-Options header is not included in the HTTP response to protect against 'Clickjackin...	6600	28 minutes a...	OPENED
☐		MED	X-Frame-Options Header ...	(443/tcp) http	faraday-vmpipelin...	34.225.224.1...	<p>X-Frame-Options header is not included in the HTTP response to protect against 'Clickjackin...	6601	28 minutes a...	OPENED
☐		MED	X-Frame-Options Header ...	(443/tcp) http	faraday-vmpipelin...	34.225.224.1...	<p>X-Frame-Options header is not included in the HTTP response to protect against 'Clickjackin...	6602	28 minutes a...	OPENED
☐		MED	X-Frame-Options Header ...	(443/tcp) http	faraday-vmpipelin...	34.225.224.1...	<p>X-Frame-Options header is not included in the HTTP response to protect against 'Clickjackin...	6603	28 minutes a...	OPENED
☐		MED	X-Frame-Options Header ...	(443/tcp) http	faraday-vmpipelin...	34.225.224.1...	<p>X-Frame-Options header is not included in the HTTP response to protect against 'Clickjackin...	6604	28 minutes a...	OPENED
☐		MED	X-Frame-Options Header ...	(443/tcp) http	faraday-vmpipelin...	34.225.224.1...	<p>X-Frame-Options header is not included in the HTTP response to protect against 'Clickjackin...	6605	28 minutes a...	OPENED
☐		MED	Vulnerable JS Library	(443/tcp) http	faraday-vmpipelin...	34.225.224.1...	<p>The identified library jquery, version 1.12.4 is vulnerable.</p> Reference: <p>https://github.c...	6606	28 minutes a...	OPENED
☐		MED	Vulnerable JS Library	(443/tcp) http	faraday-vmpipelin...	34.225.224.1...	<p>The identified library jquery, version 1.12.4 is vulnerable.</p> Reference: <p>https://github.c...	6607	28 minutes a...	OPENED

1 / 1 100 items per page

1 - 100 of 100 items

Conclusions

As you can see, this implementation was pretty easy. In addition you can see that the steps to set up the environment were pretty similar to **Jenkins Pipelines** and **Github Actions.**

Useful links

App Vuln Management: Integrating Faraday
in the software development process

Docker Faraday Report Uploader
Example repository (branch: travis)
Faraday plugin list
OWASP Zap official site
Bandit official site
Vulnerable example app