CONTRAST NODE.JS/CircleCI INTEGRATION

# Introduction

This guide describes how to set up the Contrast as part of a CircleCI build pipeline to perform three functions:

* Download the Contrast agent in order to instrument your application during build pipeline tests
* Instrument your application during unit, integration or functional tests
* Verify if any vulnerabilities were located within the specific build and optionally fail the build

# Comments

CircleCI is different than TeamCity and Jenkins etc. again and many of the systems you might use. To some regard it is very low level and many things need to be done as a shell script. There are no plug-ins or similar. Basic idea of this document is to explain what kind of docker container you need, what else to install into it and how to use a (Python) script to make the build pass or fail. This document assumes Github to be the repo. And finally, it is worth mentioning, this is a SaaS solution and cannot be run locally. (Maybe there is an enterprise on-premise for commercial use)

# Steps

## Get yourself a CircleCI account and a Github account

You need to associate the accounts and then will see your Github repos to set up for projects in CircleCI.

## Follow project setup instruction

Basically, you have to create a directory called ‘circleci’ and create a config.yml inside it.

## Select the right container image

After deciding the language for the build process, you get a template with content that might reference a different node version than you intend to use, so adapt to something like this:

- image: circleci/node:10.15.3-browsers

Appending ‘-browsers’ to the node version enables you to run Selenium scripts later on to exercise an instrumented node.js app.

## Download Agent Dynamically

This step is optional as the agent can be included within the applications source code if required. To download the agent dynamically (recommended), create a new command line build step to download the agent (and config) using the script below:

#Download node-contrast.tgz in the workspace and set the instrumentation.

#

#Do not change below

#

authorization=<username>:<servicekey>

authorization=`echo $authorization |tr -d \\\n |base64`

curl --max-time 30 <team\_server\_url>/api/ng/<orgid>/agents/external/default/node -H Accept:text/yaml -H API-Key:<apikey> -H Authorization:$authorization -o contrast\_security.yaml

curl --max-time 30 <team server url>/api/ng/<orgid>/agents/default/node -H API-Key:<apikey> -H Authorization:$authorization -o node-contrast.tgz

actualsize=$(wc -c <"node-contrast.tgz")

if [ $actualsize -ge 2000000 ]; then

echo [Contrast] Node-contrast.tgz downloaded Successfully in the Workspace

else

echo [Contrast] Node-contrast.tgz not downloaded successfully, check keys and url

exit 1

fi

The parameters for the script above will need to be assembled according to your account settings.

## 4. Instrument Tests and run them

There are a number of methods to launch the Contrast agent but first it must be installed as a module. After you have run `npm install` you should run `npm install node-contrast.tgz --no-save `. This can also be done by modifying the postinstall script of your package.json

Once the module is installed you should ensure that you have modified your package.json with a script to use the node-contrast module when running tests, e.g.:

"scripts": {

"contrast": "node-contrast app.js",

"contrast-e2e": "node-contrast test/e2eTests.js",

"protractor": "npm run contrast-e2e",

To run your tests from CircleCI you have to make a change to a file, commit it per git and push it. The build will automatically kick off in CircleCI. The free plan does not have a UI button that you can click! In order to gather build-specific information you need to either set an env var in your container and therefore in your .circleci/config.yml file that will be fed into the node agent at startup or you can use a command line parameter when running node-contrast.

Watch out, you cannot simply copy one env into another in CircleCI:

….

- checkout

- run:

name: Update PATH and Define Environment Variable at Runtime

command: |

echo 'export APPLICATION\_\_VERSION=$CIRCLE\_BUILD\_NUM' >> $BASH\_ENV

source $BASH\_ENV

CIRCLE\_BUILD\_NUM is implicitly there and will populate APPLICATION\_\_VERSION which will then result in Build Number filtering in vulnerability view.

**The build number will be passed to the agent as an environment variable. All vulnerabilities found during this build will be tagged with this version so they can be queried in the next step or counting the vulnerabilities.**

If your tests are Selenium tests e.g., you might need to run the next command to be set up correctly before you can run your end-to-end test. Be aware how you could also pass CircleCI’s build number as a parameter:

- run: node node\_modules/protractor/bin/webdriver-manager update

- run: npm run contrast-e2e -- --application.version=$CIRCLE\_BUILD\_NUM

## 5. Check Vulnerability Count

Once your tests have run you can query the contrast API to see if any new vulnerabilities were found within the build. You can modify the highlighted sections of the python script below to accomplish this:

**The following pre-requisites need to be provided:**

- run: sudo apt-get install python-pip

- run: sudo pip install requests

Because the bulk of vulnerabilities sent from the agent to TeamServer may need some processing time, so it is good advise to wait a little so the script ‘checkVulns.py ‘ fetching data from TeamServer will be receiving all information.

- run: sleep 30

- run: python ~/repo/checkVulns.py

Because the example script below needs to import the ‘requests’ module, pip needs to be installed per sudo. See that script now:

#! /usr/bin/python

import base64

import datetime

import json

import requests

import urllib

import sys

# TODO: Set the severities you are interested in

SEVERITIES = 'CRITICAL,HIGH'

# TODO: Set your application name (as per your package.json)

url = '%env.url%api/ng/%env.orgid%/applications/name?'+urllib.urlencode({ 'filterText' : 'APPLICATION NAME', 'filterServers' : '%teamcity.agent.hostname%' })

headers = {

    'Accept': 'application/json',

    'API-Key': '%env.apikey%',

    'Authorization': base64.b64encode('%env.username%:%env.servicekey%')

}

print ('HTTP GET ' + url)

response = requests.get(url, headers = headers)

# Check the status of the request

if (not response.ok):

    response.raise\_for\_status()

    exit()

# Parse the JSON content

json\_data = json.loads(response.content)

# Get the application id for the app

# TODO: Error handling

APP\_ID = json\_data['applications'][0]['app\_id']

# Contrast Security API request to get vulnerabilities

url = '%env.url%api/ng/%env.orgid%/traces/'+APP\_ID+'/quick?'+urllib.urlencode({ 'severities' : SEVERITIES, 'filterText' : '%env.BUILD\_NUMBER%' })

print ('HTTP GET ' + url)

response = requests.get(url, headers = headers)

# Check the status of the request

if (not response.ok):

    response.raise\_for\_status()

    exit()

# Parse the JSON content

json\_data = json.loads(response.content)

vulns\_all = 0

vulns\_open = 0

for filter in json\_data['filters']:

    if (filter['name'] == 'All'):

        vulns\_all = filter['count']

    elif (filter['name'] == 'Open'):

        vulns\_open = filter['count']

print ('All vulnerabilities: ' + str(vulns\_all))

print ('Open vulnerabilities: ' + str(vulns\_open))

# TODO Set the threshold for the number of vulnerabilities (of given severities)

if (vulns\_open > 0):

    sys.exit(1)

In the script above if the number of high or critical vulnerabilities is greater than zero then the build will fail.

# About Contrast Security

Contrast Security provides technology that enables software applications to protect themselves against cyberattacks, heralding the new era of self-protecting software. Contrast's patented deep security instrumentation is the breakthrough technology that enables highly accurate assessment and always-on protection of an entire application portfolio, without disruptive scanning or expensive security experts.

Organizations who are practicing or adopting modern development methodologies (e.g., DevOps) are striving to innovate faster and deliver value quicker via their software offerings. Accelerating the delivery of value to customers is forcing companies to iterate their software more frequently, but what happens when that software needs to be audited or scanned for security and compliance? Answer: there is a halt to the organization’s ability to move fast!

Contrast Security instruments your applications from the inside by placing sensors throughout the applications. These sensors automatically discover application components, collect telemetry data and constantly observe transaction data flows to identify security vulnerabilities and block malicious attacks. This ubiquitous security monitoring does not need to be turned on and off like a scanner, it does not slow down your rate of software development and innovation, and it provides immediate insight into vulnerabilities and attacks, thus making software self-protecting.

# Appendix A - .circleci/config.yml

# Javascript Node CircleCI 2.0 configuration file

#

# Check https://circleci.com/docs/2.0/language-javascript/ for more details

#

version: 2

jobs:

build:

docker:

# specify the version you desire here

- image: circleci/node:10.15.3-browsers

# environment: Note: You cannot use a built-in environment variable to define another environment variable. Instead, you must use a run step to export the new environment variables using BASH\_ENV

# APPLICATION\_\_VERSION: CIRCLE\_BUILD\_NUM

# Specify service dependencies here if necessary

# CircleCI maintains a library of pre-built images

# documented at https://circleci.com/docs/2.0/circleci-images/

# - image: circleci/mongo:3.4.4

working\_directory: ~/repo

steps:

- checkout

- run:

name: Update PATH and Define Environment Variable at Runtime

command: |

echo 'export APPLICATION\_\_VERSION=$CIRCLE\_BUILD\_NUM' >> $BASH\_ENV

source $BASH\_ENV

# Download and cache dependencies

- restore\_cache:

keys:

- v1-dependencies-{{ checksum "package.json" }}

# fallback to using the latest cache if no exact match is found

- v1-dependencies-

- run: npm install

- run: npm install node-contrast-2.0.1.tgz --no-save

- save\_cache:

paths:

- node\_modules

key: v1-dependencies-{{ checksum "package.json" }}

# run tests!

- run: node node\_modules/protractor/bin/webdriver-manager update

- run: npm run contrast-e2e -- --application.version=$CIRCLE\_BUILD\_NUM

- run: sudo apt-get install python-pip

- run: sudo pip install requests

- run: sleep 30

- run: python ~/repo/checkVulns.py

# Appendix B - .circleci/config.yml