<https://etherpad.net/p/ArnoWillaert>

Correctly building and starting a container for this application:

Make a well-formed docker file such as:

Img

docker build -t sdtf/sdtf\_3 .

Takes the Dockerfile in the current directory (indicated by the “.”) and builds an image based on that. The –t stands for tag and is used to give the image a readable name. Formatted repository/name:tag, with repository and tag being optional

docker run -t -d --name ChromeyBoii sdtf/sdtf\_3 /bin/bash

Creates and starts a container using the docker image “sdtf/sdtf\_3”, with the entry point /bin/bash and the name “Chromeyboii”. –d stands for run detached and –t sets up a pseudo TTY link

docker exec -it ChromeyBoii bash

Used to execute commands. Here: enter a new bash shell in the container with the name Chromeyboii. –it is used to keep the shell open.

Short word about my sdtf image versions\

Sdtf/sdtf is the first testing ground before I merged with the master

Sdtf/sdtf\_2 is the container made after the merge, where I did most of the troubleshooting

Sdtf/sdtf\_3 is the image I made that installs sdtf and is able to successfully run the chrome webadmin

Sdtf/sdtf\_4 is the image that

# Week 2

## Feb 19

Verder gedaan aan Selenium docker.

Eerst docker-compose erbij geïnstalleerd

Dan met de volgende docker compose file gewerkt: (bron: https://github.com/SeleniumHQ/docker-selenium/wiki/Getting-Started-with-Docker-Compose)

selenium-hub:

image: selenium/hub

ports:

- 4444:4444

chrome:

image: selenium/node-chrome

links:

- selenium-hub:hub

volumes:

- /dev/shm:/dev/shm # Mitigates the Chromium issue described at https://code.google.com/p/chromium/issues/detail?id=519952

firefox:

image: selenium/node-firefox

environment:

HUB\_PORT\_4444\_TCP\_ADDR: hub

links:

- selenium-hub:hub

^ dat werkt, en het valt te scalen met:

docker-compose scale chrome=5

docker-compose scale firefox=5

Als ge die doet krijgt ge ne WARNING dat deze commands deprecated zijn, en dat ge docker-compose up met --scale flag moet gebruiken, maar dat hielp alles echt neig omzeep, dus wees daar voorzichtig mee.

Verder heb ik nu sdtf in een container gekregen, ~~het is nog niet installed wel, gewoon de repo zit in de container~~. Het is installed nu.

Om in de container te geraken gebruikt ge het commando

Docker exec -it zealous\_gates bash

Dan in de dir sdtf

Chmod +x install\_sdtf\_linux.sh

./install\_sdtf\_linux.sh

En dan om in de runtime te gaan :

source ~/sdtf/sdtf-activate

## Feb 20

Weekverslag gemaakt van vorige week.

Eens een poging doen om da op mijne windows te zetten.

Bij die commands voor selenium moet ge nen make build doen en als ge da in die docker toolbox doet, dan wilt da ni omda da geen bash scripts wilt runnen. En als ge da in een shell doet da da wel support (cygwin64) dan zegt het dat docker een unknown command is, dusja, ik geraak der nu ff nergens mee.

Dus da ga ik even achteruitschuiven op de prioriteitenlijst.

Dan heb ik bekeken of de sdtf container gebruik kon maken van de selenium grid, het antwoord is ja, maar de reis was wat anders.

(<https://github.com/SeleniumHQ/docker-selenium/wiki/Getting-Started-with-Docker-Compose>)

from selenium import webdriver

from selenium.webdriver.common.desired\_capabilities import DesiredCapabilities

chrome = webdriver.Remote(

command\_executor='http://localhost:4444/wd/hub',

desired\_capabilities=DesiredCapabilities.CHROME)

firefox = webdriver.Remote(

command\_executor='http://localhost:4444/wd/hub',

desired\_capabilities=DesiredCapabilities.FIREFOX)

chrome.get('https://www.google.com')

print(chrome.title)

firefox.get('https://www.google.com')

print(firefox.title)

chrome.quit()

firefox.quit()

We bewerken de bovenstaande code zodat ze enkel de test gaat doen in google chrome en de juiste ip gaan gebruiken.

IP van de container gevonden met :

$ docker inspect dockerselenium\_selenium-hub\_1 | grep IP

Dus dan maken we in de container met sdtf in het volgend script aan

from selenium import webdriver

from selenium.webdriver.common.desired\_capabilities import DesiredCapabilities

chrome = webdriver.Remote(

command\_executor='http://172.17.0.2:4444/wd/hub',

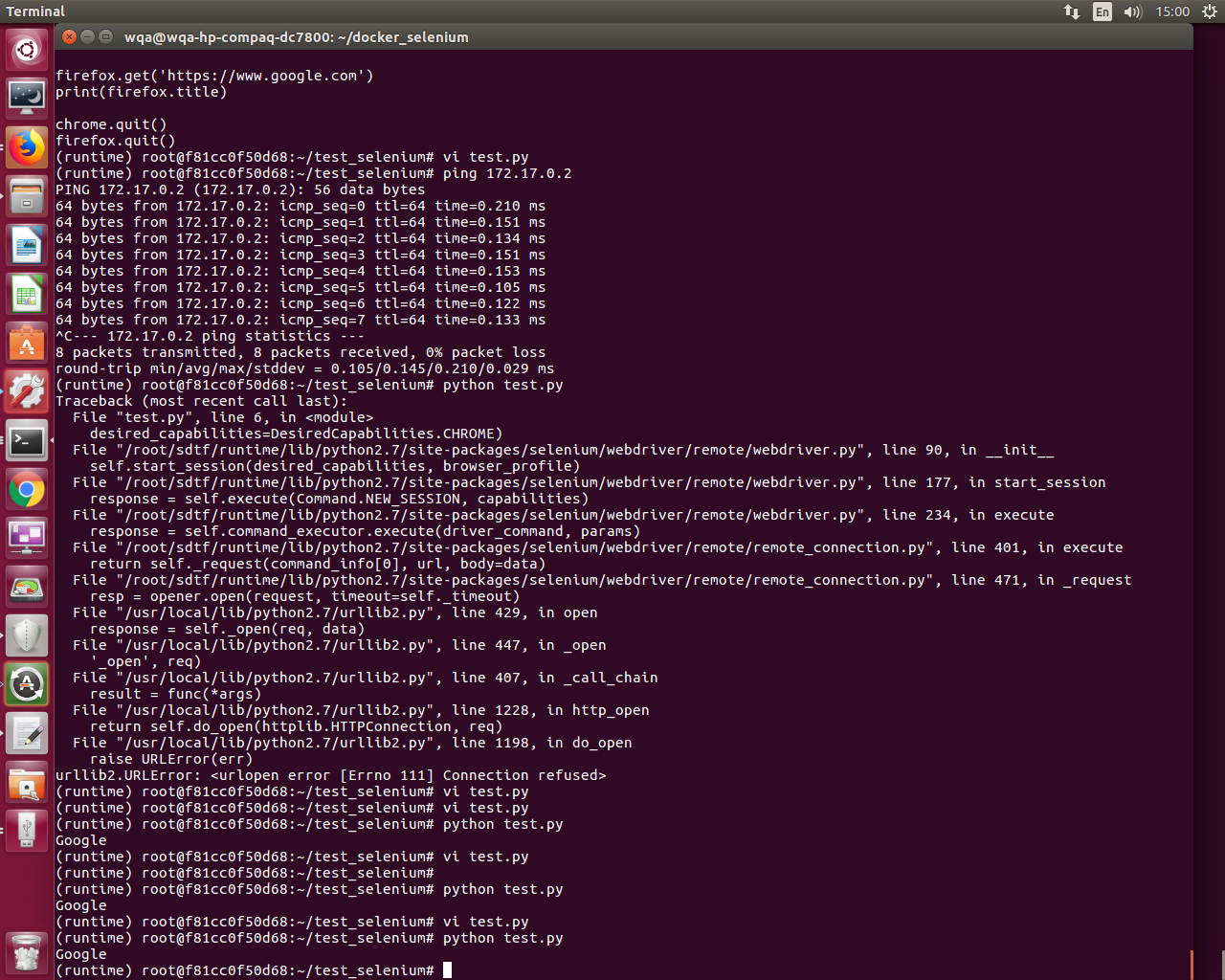
desired\_capabilities=DesiredCapabilities.CHROME)

chrome.get('https://www.google.com')

print(chrome.title)

chrome.quit()

Dit scriptje gaat in een chrome node van de selenium grid de webpagina [www.google.com](http://www.google.com) opvragen en dan de title printen in de console. We gaan eens kijken of da werkt.



Precies wel eh!

Hierna ga ik onderzoeken of ik de browsers automatisch een venster kunnen laten openen de moment dat de docker-compose up wordt gerunt.

firefox:

image: firefox

environment:

- DISPLAY=$DISPLAY

volumes:

- /tmp/.X11-unix:/tmp/.X11-unix

Macheert anders wel. Gewoon in die directory gaan staan en de command “docker-compose up” runen en firefox opent me een window.

(even sidetrack) Possible containers:

* Performance suite
* Radious

## Feb 21

Dan heb ik een poging gedaan om de docker-compose van selenium grid aan te passen zodat die x11 geforward word ten de environment variable gezet wordt voor display, maar dan werkt de docker-compose run niet meer.

Nader onderzoek toont aan dat dit komt omdat in de image gebruikt voor de nodes met de browsers genen x11 inzit.

Hoe lossen we dat op? Opties:

* Andere images builden die da wel hebben
* Zien da we x11 daarop daarop krijgen bij elke container die wordt gestart

Nevermind, da heeft da (de x11 eh) precies wel (ik deed dir in de tmp folder ipv ls-a). Maar waarom wilt da dan ni?

Na de docker-compose up is er in zowel firefox\_1 als chrome\_1 een error: “xvfb-run: error: Xvfb failed to start”

^mogelijk een probleem met environment variables

Ik wordt er echt ni wijzer uit, ik ga verder naar het volgende.

Mogelijke work-arounds: VNC gebruiken met debug images als nodes, of anders vanaf scratch images builden en zien of er rond die xvfb te werken valt.

Nu gaan ik ne compose file maken voor sdtf en selenium grid te lanceren tegelijkertijd. Om dit deftig te kunnen bereiken ga ik wel eerst een dockerfile aanmaken voor een sdtf image waar sdtf ineens op installed is, want diegene die ik tot nu toe heb gebruikt doet dat niet uit zichzelf.

FROM python:2.7

ADD python\_test.py /

ADD sdtf /root/sdtf

RUN apt-get update && apt-get install -y \

python-tk \

python-pip \

python-ldap \

&& rm -rf /var/lib/apt/lists/\*

RUN apt-get update --fix-missing

RUN pip install virtualenv

RUN chmod +x /root/sdtf/install\_sdtf\_linux.sh

RUN /root/sdtf/install\_sdtf\_linux.sh

CMD [ "/bin/bash" ]

Bovenstaande dockerfile gaat dit realiseren, MAAR ik krijg nog rood in de log. Het probleem zijnde dat sudo niet ondersteund wordt. Ik weet nu ni of dat een probleem gaat geven. Lijkt me niet want als ik met docker exec in de container gaat lijkt het te werken.

## Feb 22

sdtf:

image: sdtf/sdtf:

selenium-hub:

image: selenium/hub

ports:

- 4444:4444

chrome:

image: selenium/node-chrome

links:

- selenium-hub:hub

volumes:

- /dev/shm:/dev/shm # Mitigates the Chromium issue described at https://code.google.com/p/chromium/issues/detail?id=519952

firefox:

image: selenium/node-firefox

environment:

HUB\_PORT\_4444\_TCP\_ADDR: hub

links:

- selenium-hub:hub

Dit is eigenlijk gewoon de compose file die ik gebruik voor selenium met sdtf erboven toegevoegt.

Als ik deze compose file gebruik dan word de sdtf container inderdaad gestart, maar ze stopt ook meteen.

Dit probleem doet zich ook voor als ik enkel de image run, maar daar kan ik het oplossen door de flag –it mee te geven. -i houdt de interactive shell open en -t zorgt voor een pseudo tty (niet zeker wat da is, maar whatever, het macheerde).

Oplossing gevonden in deze thread (<https://github.com/docker/compose/issues/5016>).

Kheb enkel stdin: true moeten toevoegen en het werkte al, dus das cool.

Dus nu als ik docker-compose run doe in docker\_sdtf\_test, dan krijg ik 4 containers: de container met SDTF, een container met de selenium hub, een container met een selenium node met chrome in en ene met firefox.

Voor de volledigheid nog eens de compose file:

sdtf:

image: sdtf/sdtf

stdin\_open: true

selenium-hub:

image: selenium/hub

ports:

- 4444:4444

chrome:

image: selenium/node-chrome

links:

- selenium-hub:hub

volumes:

- /dev/shm:/dev/shm # Mitigates the Chromium issue described at https://code.google.com/p/chromium/issues/detail?id=519952

firefox:

image: selenium/node-firefox

environment:

HUB\_PORT\_4444\_TCP\_ADDR: hub

links:

- selenium-hub:hub

Wat onderzoek gedaan rond internet explorer en hoe dat in een node te steken. Eerst had ik wel goede moed, met die NodeBase files die ik vond in de selenium docker repo, maar ja, aangezien IE enkel op windows draait en containers dezelfde os gebruiken als hun host is dit redelijk onmogelijk op de moment.

## Feb 23

Eindelijk gefixt da ge in die selenium nodes kunt kijken

selenium-hub:

image: selenium/hub

ports:

- 4444:4444

chrome-debug:

image: selenium/node-chrome-debug

environment:

HUB\_PORT\_4444\_TCP\_ADDR: hub

links:

- selenium-hub:hub

ports:

- 5902:5900

volumes:

- /dev/shm:/dev/shm # Mitigates the Chromium issue described at https://code.google.com/p/chromium/issues/detail?id=519952

firefox-debug:

image: selenium/node-firefox-debug

environment:

HUB\_PORT\_4444\_TCP\_ADDR: hub

links:

- selenium-hub:hub

ports:

- 5901:5900

Let wel op dat het nodig is de nodes te starten met een debug image

# Week 3

## Feb 26

Worked on thesis proposal

Had the meeting between Phaedra, Chris and me.

Further finished the thesis proposal, started on the powerpoint.

Went to a class about presentation skills

## Feb 27

Further prepared for my presentation

Look into creating a Docker image from an existing container, so I don’t need to install SDTF each time I need a new container.

<https://docs.docker.com/engine/reference/commandline/commit/>

The Docker commit command is to be used to this purpose.

docker commit zealous\_gates sdtf/sdtf\_installed:latest

Zealous\_gates being a container with the sdtf installed on.

So the good news is that if this works, the image can be deployed on a machine that doesn’t have the files on hand to install SDTF

And it does work, so, cool.

Drawback: this means potentially putting multiple images on AWS, which is costly as price goes up with amount of images installed. The alternative is to get the install files in an AWS bucket and install them that way.

## Feb 28

Made a planning in an excel file kind of in the style of a SCRUM board, so that’s cool.

Initial feedback from Chris: maybe the thesis section is overrepresented. This could hurt my thesis. I should probably consult Phaedra about this.

Installing a Selenium node with ONLY IE on a windows machine:

java -jar selenium-server-standalone-3.9.1.jar -role node -nodeConfig nodeConfig.json

{

"capabilities":

[

{

"browserName": "internet explorer",

"platform": "WINDOWS",

"maxInstances": 1,

"seleniumProtocol": "WebDriver"

}

],

"maxSession": 5,

"port": 5555,

"register": true,

"registerCycle": 5000,

"hub": "http://10.132.224.165:4444",

"nodeStatusCheckTimeout": 5000,

"nodePolling": 5000,

"role": "node",

"unregisterIfStillDownAfter": 60000,

"downPollingLimit": 2,

"debug": false

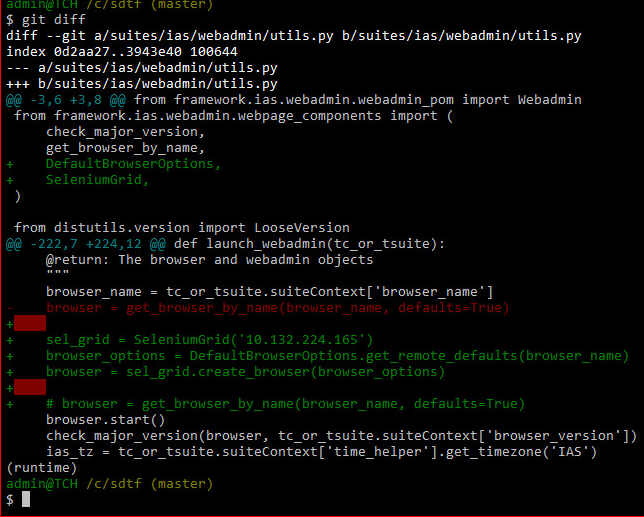
}

It works, yay

## March 1

Started out with some admin stuff, handed in weekly report number 2 on Toledo and the first promotor-business promotor report.

Next up: WebAdmin!



BUT when testing on IAS 3.15 it crashes after a bit due to problems out of the scope of this project. So further testing will be done on IAS 3.14

The Implementation of this on the SDTF container proved to be more difficult because the utils.py is different from that in the main branch.

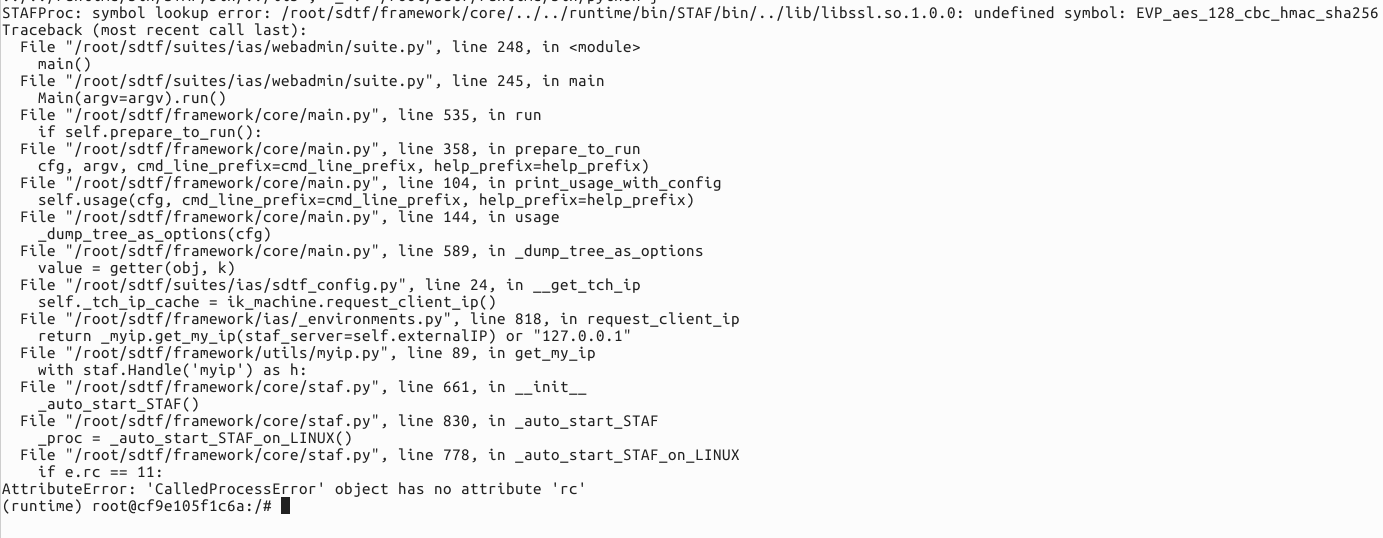
I talked to Jurgen about this issue and he proposed merging the master into the Linux port branch (the branch I am using on the SDTF container)

That part of the code is actually in the webamin\_suite.py, it got moved to utils for some work-around in more recent versions. So I actually need to change the code in webadmin\_suite.py

I made the changes and then ran

python ~/sdtf/suites/ias/webadmin/suite.py "3.14.0.4791" "UbuntuS-16.04-x64\_PGSQL\_SSM" ChromeWebadminTestSuite --projectName="Identikey Server" --testplanName="IAS 3.14.0 - Automated Webadmin" --virt.user="lm-auto-wemmel" --virt.password=$LMPW --traces.user="lm-user" --traces.password=Shared1234 --install.media="//10.132.0.242/wqa/QC-Projects/01-Identikey/3.15.0/builds/3.15.0.2262/ias-dev\_3.15.0.2262.iso" --deployed\_name="3.14.0.4791\_UbuntuS-16.04-x64\_PGSQL\_SSM"

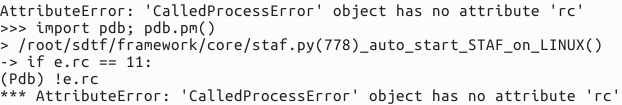
Returns this error



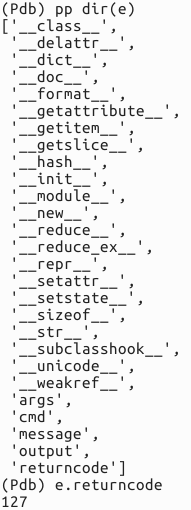
Probably related to STAF not launching automatically. Consult Koen next morning about this.

## March 2

So now we are root causing the issue I had yesterday

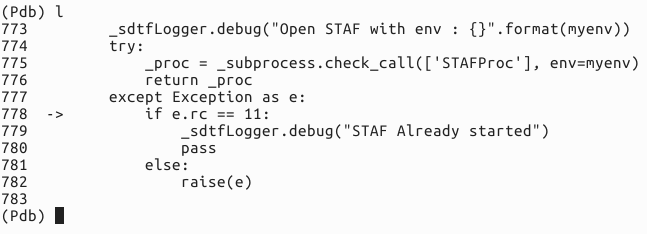


So here we find out that the object doesn’t in fact have an attribute rc



With the first command we find out what attribute the object does have, and we stumble upon an attribute returncode, which is probably what rc was intended to be.

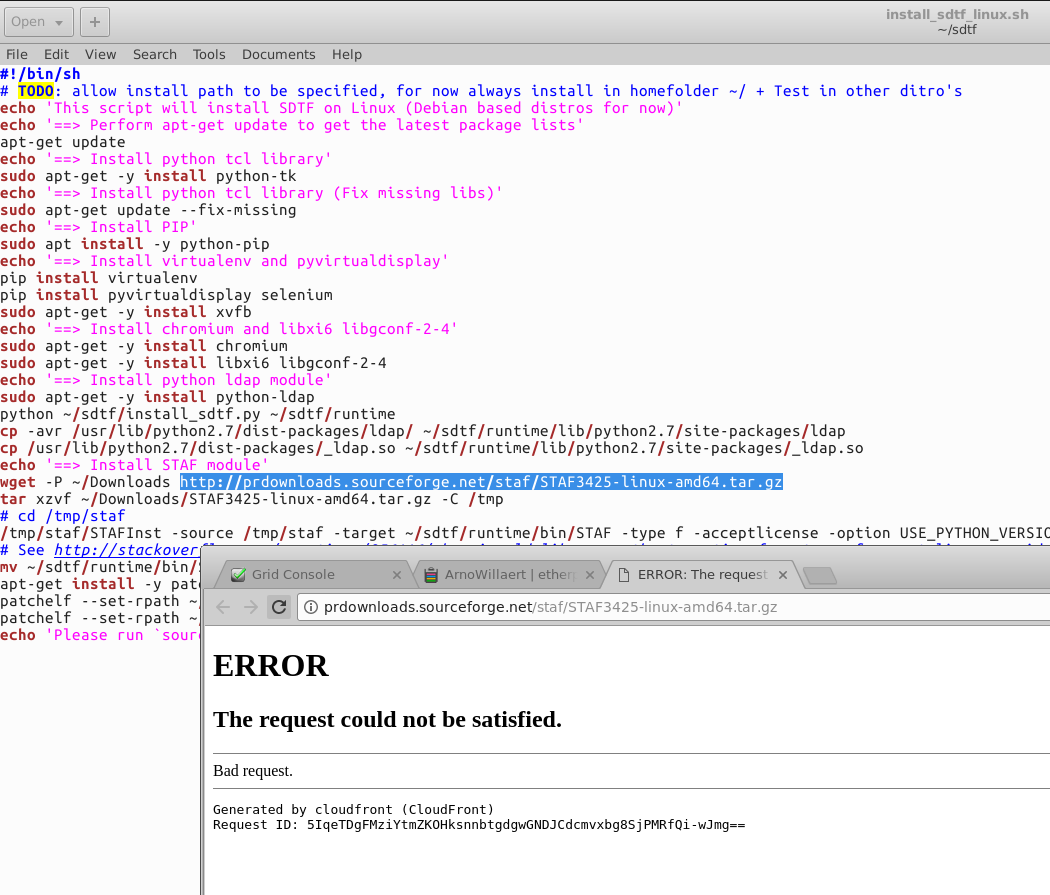
Then we see what is in the return code, and we find 127.



While it looks like the code is expecting an 11.

An error code 127 apparently has something to do with using double or single quotes as per this thread: <http://staf-users.narkive.com/XrEUgzXo/return-code-127>

Or just a problem with an unsuccessful STAF install



So, next point on the agenda: try to install STAF using a bin file in sdtf/resources/STAF3425-setup-linux-amd64-NoJVM.bin

It didn’t want to run, and gives the following message: “No Java virtual machine could be found from your PATH environment variable. You must install a VM prior to running this program”

So then did a quick java install with “apt-get install default-jdk”

Tried to install the bin file again, but it returned:

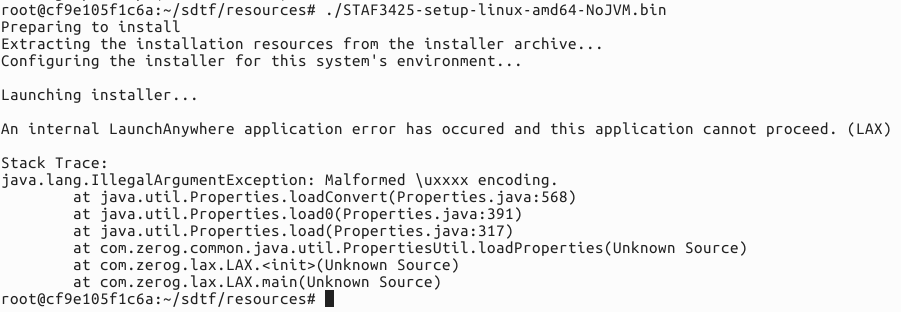
./STAF3425-setup-linux-amd64-NoJVM.bin: 1: ./STAF3425-setup-linux-amd64-NoJVM.bin: unzip: not found

Invalid unzip command found

Soo, that‘s not good.

Turns out that I need to install unzip ( apt-get install unzip )

So now I’m stuck with this:



Just kinda said fuck it to that file, it’s probably broken for some reason, idk, so I mounted the network drive with a decent tar file on it in my filesystem and copied it to my container using the following command:

docker cp /media/wqa/Software/STAF/STAF3420-linux-amd64.tar.gz cf9e105f1c6a:/root/

SO, TURNS OUT

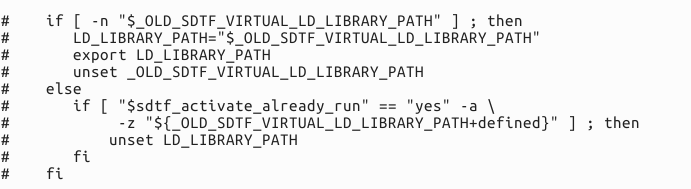
That STAF was probably installed correctly, BUT, LD\_LIBRARY\_PATH wasn’t being properly initialized in the sdtf activate script, because it was commented out in favor of putting it elsewhere.

So when uncommenting it, we can in fact run STAFProc in the virtual environment. So we’ll see Monday if the thing works.

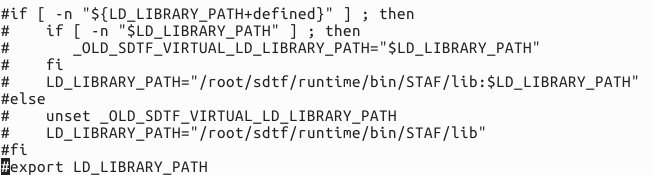
# Week 4

## March 5

In sdtf/activate-sdtf uncomment the following:

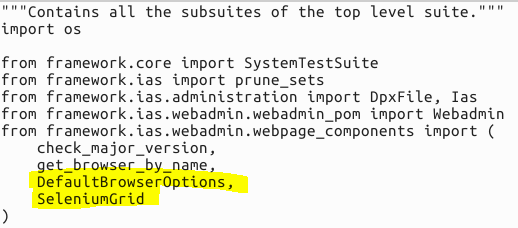
Lines 43 to 52: 

Lines 98 to 107:

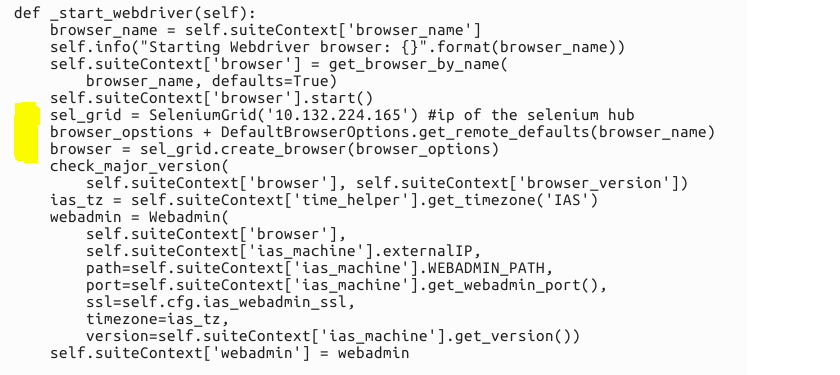


Then in the file webadmin\_suite, we need to make the following changes:

At the top of the file:



Then on lines 80 to 82:



(there is a mistake in here somewhere, fixed later on)

Framework/core/staf.py

Line 775: check\_call naar Popen

So that fixes the problem with staf

Dingen in de install script me sudo voor opnief gedaan

~/sdtf/runtime/bin chmod +x chromedriver

In short, the problem now is that webadmin suite tries to open chrome or firefox to register our IP with IAS as part of the SetUp. It can’t do this somehow though, due to problems with the webdrivers of the respective browsers.

## March 6

Further investigated the problems with the webdriver, a workaround is probably a better idea at this point.vi

Made a word file where I define a set of suites that I am planning to implement once I get it running.

## March 7

The problem with the webdriver concrete: as it stand now, the webadmin suite does some stuff locally in a browser, such as looking up the IAS version and creating users in webadmin.

My plan for a workaround is doing all this jazz on a selenium node instead of locally.

In webadmin\_suites.py, on line 72:

def \_start\_webdriver(self):

browser\_name = self.suiteContext['browser\_name']

self.info("Starting Webdriver browser: {}".format(browser\_name))

sel\_grid = SeleniumGrid('10.132.224.165') #ip of the selenium hub

browser\_options = DefaultBrowserOptions.get\_remote\_defaults(browser\_name)

browser = sel\_grid.create\_browser(browser\_options)

self.suiteContext['browser'] = get\_browser\_by\_name(

browser\_name, defaults=True)

self.suiteContext['browser'].start()

check\_major\_version(

self.suiteContext['browser'], self.suiteContext['browser\_version'])

ias\_tz = self.suiteContext['time\_helper'].get\_timezone('IAS')

webadmin = Webadmin(

self.suiteContext['browser'],

self.suiteContext['ias\_machine'].externalIP,

path=self.suiteContext['ias\_machine'].WEBADMIN\_PATH,

port=self.suiteContext['ias\_machine'].get\_webadmin\_port(),

ssl=self.cfg.ias\_webadmin\_ssl,

timezone=ias\_tz,

version=self.suiteContext['ias\_machine'].get\_version())

self.suiteContext['webadmin'] = webadmin

Changed the location of the selenium grid call by a few line, and also fixed some typo‘s

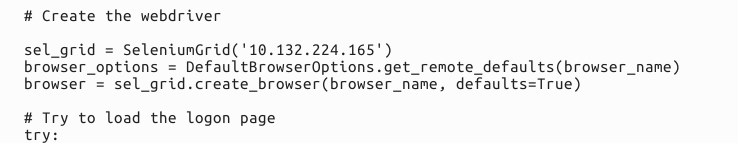
In the administration.py, I’m going to throw out some code related to windows and replace the linux part with code for selenium grid.

The goal here is to do the operations on selenium instead of locally.

So I’m deleting everything between # Create the webdriver and # Try and load the logon page (line 856 to 891)



To:



The comment “create webdriver” doesn’t really make sense anymore, but we’ll fix that eventually

But now something else broke:



This is probably fixed by importing the right things in \_environments.py

So I imported the following:

from framework.ias.webadmin.webpage\_components import (

check\_major\_version,

get\_browser\_by\_name,

DefaultBrowserOptions,

SeleniumGrid

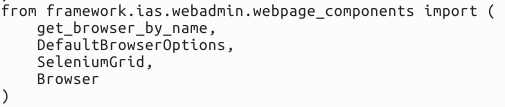
)

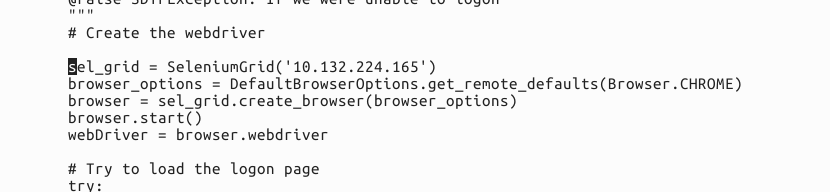
This is probably too much, but I’ll revise this once it works.

It didn’t help though

So apparently, this doesn’t need to be in \_environments.py, but in administration.py

Now we get another error, cause is probably that we don’t specify the browser version we want to open.





That fixed that error, but now it just gets stuck somewhere else.

## March 8

Job fair

## March 9

So I discovered today that due to a power outage the Linux machine running the docker containers stopped incorrectly causing me to basically lose all progress I had in my running containers.

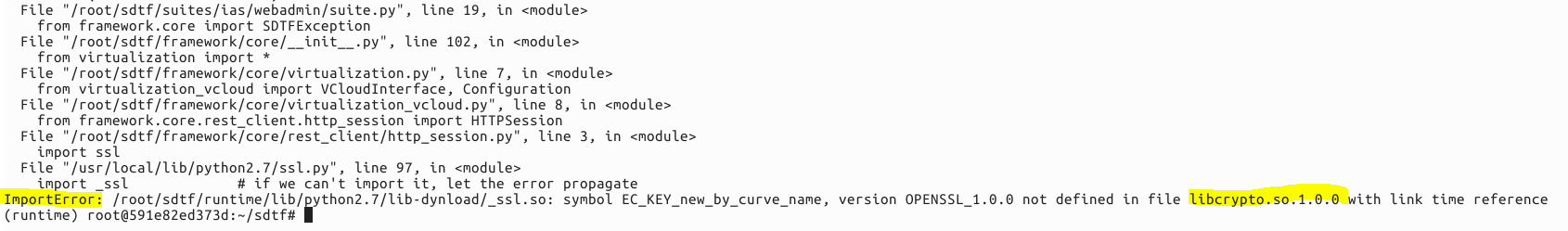
Luckily enough, I meticulously documented everything I have changed to date, so I can retrace my steps. Still, fuck.

# Week 5

## March 12

Proceeded on fixing

Ran into a new issue after retracing my steps:



Which is odd, because normally the libcrypto file should be fine.

Terug de lb\_library\_pat variable weggecomment, omdat die dinges waarschijnlijk ergens anders wordt geset en daarom een error geeft. ( in activate )

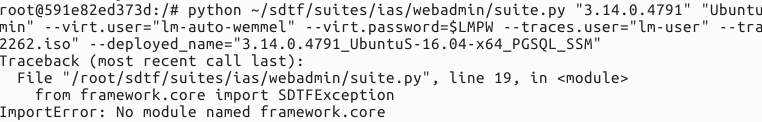
Further, the password is not set ($LMPW)

We put it in ~/.bash\_profile

export LMPW=U89yMN8R8bwgOKR1k6tz

Then it still doesn’t work, but apparently there is something broken atm, so there is that.

## March 13



Suddenly I get this error, instead of the password error yesterday, so let’s check that out.



These seem to be the offenders

NEVERMIND, I wasn’t in the runtime…

It still broke, but I probably fixed it by writing chrome in caps in administration.py (line 864)

So, I got it running eventually, but it gets stuck because I am working with outdated test code, and the IAS service.

The proposed solution for this is to pull the master branch into my branch to get the new test suite.

In preparation of this, I committed my changes to the branch to the repository.

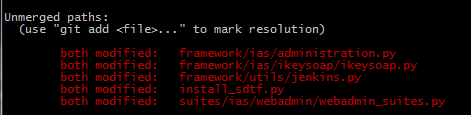
# March 14

Today we pull the master branch into my branch to “update” my code.

Merge with: git pull origin master

Then manually resolve merge conflicts with notepad++

Merge conflicts:



Important note on the conflict in webadmin\_suites.py: The start\_webdriver was moved to suites/ias/webadmin/utils.py and renamed launch\_webadmin.

That code now looks like this:

def launch\_webadmin(tc\_or\_tsuite):

"""Launch the webdriver and open the Webadmin page from the suite context.

Following values will be used from the C{tc\_or\_tsuite.suiteContext} variable:

- browser\_name

- browser\_version

- time\_helper

- ias\_machine

@return: The browser and webadmin objects

"""

browser\_name = tc\_or\_tsuite.suiteContext['browser\_name']

sel\_grid = SeleniumGrid('10.132.224.165') #ip of the selenium hub

browser\_options = DefaultBrowserOptions.get\_remote\_defaults(browser\_name)

browser = sel\_grid.create\_browser(browser\_options)

browser = get\_browser\_by\_name(browser\_name, defaults=True)

browser.start()

check\_major\_version(browser, tc\_or\_tsuite.suiteContext['browser\_version'])

ias\_tz = tc\_or\_tsuite.suiteContext['time\_helper'].get\_timezone('IAS')

webadmin = Webadmin(

browser,

tc\_or\_tsuite.suiteContext['ias\_machine'].externalIP,

path=tc\_or\_tsuite.suiteContext['ias\_machine'].WEBADMIN\_PATH,

port=tc\_or\_tsuite.suiteContext['ias\_machine'].get\_webadmin\_port(),

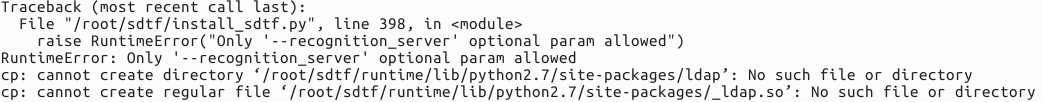
ssl=tc\_or\_tsuite.cfg.ias\_webadmin\_ssl,

timezone=ias\_tz,

version=tc\_or\_tsuite.suiteContext['ias\_machine'].get\_version())

return browser, webadmin

Next up: creating a container with the updated code in it, and hope that it runs



After the merge, there is an error somewhere in the install script

TO DO: Jenkins install failed, probably something wrong in the merge

Script asks to override git hooks, which requires manual actions, and that’s not good.

Jenkins install fails due to version mismatch

Modified install script

root@be5744c32fc8:~/sdtf# source ~/sdtf/sdtf-activate

(runtime) root@be5744c32fc8:~/sdtf# python ~/sdtf/suites/ias/webadmin/suite.py "3.14.0.4791" "UbuntuS-16.04-x64\_PGSQL\_SSM" ChromeWebadminTestSuite --projectName="Identikey Server" --testplanName="IAS 3.14.0 - Automated Webadmin" --virt.user="lm-auto-wemmel" --virt.password=$LMPW --traces.user="lm-user" --traces.password=Shared1234 --install.media="//10.132.0.242/wqa/QC-Projects/01-Identikey/3.15.0/builds/3.15.0.2262/ias-dev\_3.15.0.2262.iso" --deployed\_name="3.14.0.4791\_UbuntuS-16.04-x64\_PGSQL\_SSM"

Traceback (most recent call last):

File "/root/sdtf/suites/ias/webadmin/suite.py", line 23, in <module>

from framework.ias.\_environments import IdentikeyMachine

File "/root/sdtf/framework/ias/\_\_init\_\_.py", line 1, in <module>

from ias\_systemtest import IasSystemTestCase

File "/root/sdtf/framework/ias/ias\_systemtest.py", line 5, in <module>

from framework.ias.administration import get\_environment

File "/root/sdtf/framework/ias/administration.py", line 5, in <module>

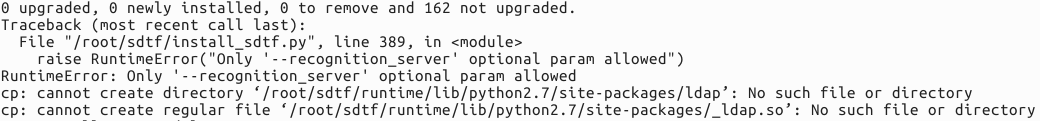
import ldap as \_ldap

ImportError: No module named ldap

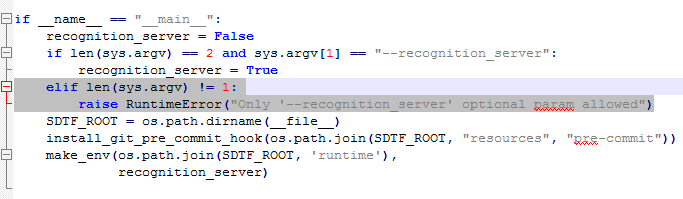
## March 15

In install script looked for the Jenkins mismatch and changed the older version for the newer

Problem of having an empty runtime dir persists

Relevant error during the install: 

Deleted these two lines in install\_sdtf.py:



So this solved that problem.

Now we’re back to running

python ~/sdtf/suites/ias/webadmin/suite.py "3.14.0.4791" "UbuntuS-16.04-x64\_PGSQL\_SSM" ChromeWebadminTestSuite --projectName="Identikey Server" --testplanName="IAS 3.14.0 - Automated Webadmin" --virt.user="lm-auto-wemmel" --virt.password=$LMPW --traces.user="lm-user" --traces.password=Shared1234 --install.media="//10.132.0.242/wqa/QC-Projects/01-Identikey/3.15.0/builds/3.15.0.2262/ias-dev\_3.15.0.2262.iso" --deployed\_name="3.14.0.4791\_UbuntuS-16.04-x64\_PGSQL\_SSM"

Doesn’t work, forgot to put the LMPW in the bash profile

vi ~/.bash\_profile

Add:

export LMPW=U89yMN8R8bwgOKR1k6tz

Save, then:

source ~/.bash\_profile

To load the variable

Now we get this soup



(Again?) Something with the libssl

To fix:

mv ~/sdtf/runtime/bin/STAF/lib/libcrypto.so.1.0.0\_bak ~/sdtf/runtime/bin/STAF/lib/libcrypto.so.1.0.0

TO DO: Test if I can’t just leave that out of the install script.

WEBADMIN RUNS, YAY

Issue now: webdriver\_move\_and\_resize\_window uses win32api on a certain point, which we can’t call because we’re on a Linux.

Temporary fix: moved it inside an if loop that we don’t really go into under normal circumstances. (So yeah, pretty temporary).

## March 16

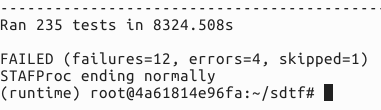
Issue: webdriver\_move\_cursor also relies on a windows specific attribute:

ctypes.windll.user32.SetCursorPos(cursor\_pos\_x, cursor\_pos\_y)

So yeah, that’s problematic.

The thing is, this operation is only necessary when running the test suite against an Internet Explorer.

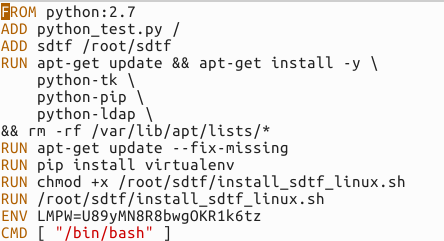
When this is commented, it can still be run against a chrome. I’m running the test suite as I am writing this, and it’s looking good.



YEAH BOII

# Week 6

## March 19



Added the command to run the install script

Added the environment variable that is a necessary credential for the command that executes the webadmin suite.

Made a docker image that is able to run the chrome suite out of the box

## March 20

Tried some different things with the Dockerfile and with docker exec to remove the need to go inside the machine to execute a suite but came upon some difficulties.

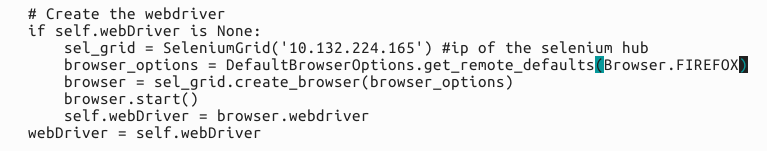
Is you ever run /bin/bash while in the sdtf-activate runtime, you exit the runtime and go into a new instance of a bash shell. This is problematic because I can’t really go and check if the machine is really in runtime without going into the machine; but to do that I need to do docker exec –it <containername> bash (or /bin/bash).

Docker attach is also deprecated apparently, so I can’t use that either.

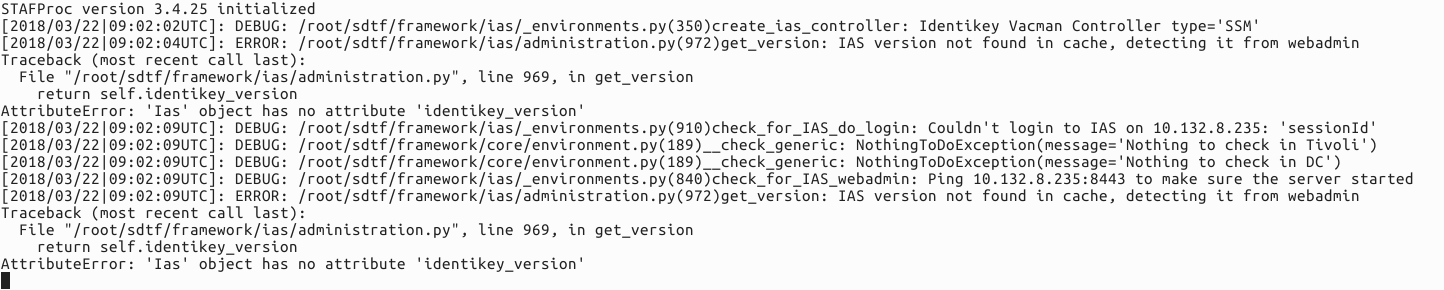
## March 21

Admin stuff

## March 22



^ sdtf/framework/ias/administration.py



Stalls on this

class FF58WebadminTestSuite(\_WebadminTestSuite):

tl\_mapping = ["WQA Webadmin", "Firefox ESR 58"]

browser\_name = Browser.FIREFOX

browser\_version = '58'