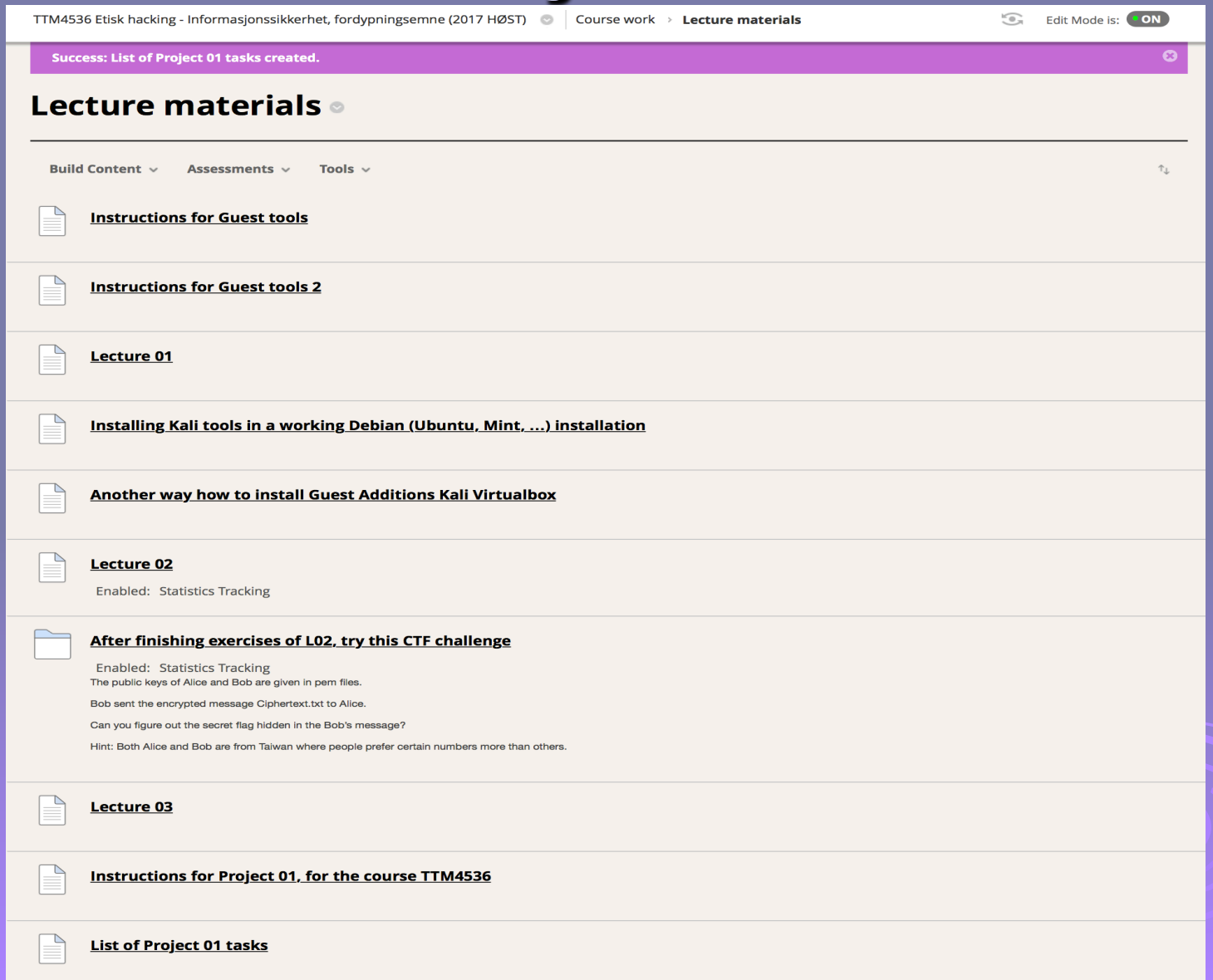
TTM4536 - Ethical Hacking - Information Security, Specialization Course

- Plan for 19 Sep 2017

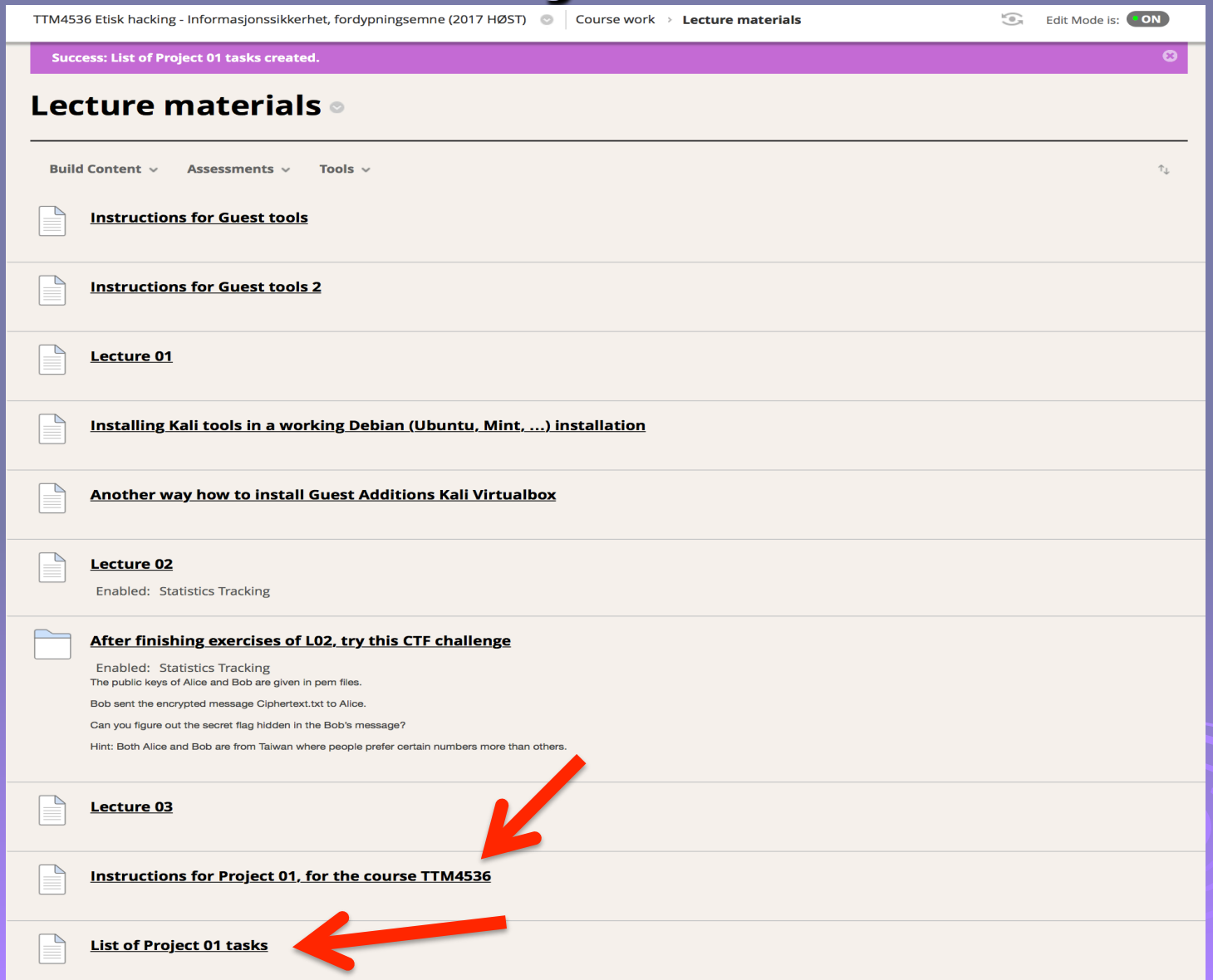
Part 1
Discussion about your Project 01

Part 2
Doing the exercises from Chapter 3

Discussion about your Project 01



Discussion about your Project 01



Discussion about your Project 01

Instructions for Project 01, for the course TTM4536

For the project you will work with the suite of python libraries called "angr". "angr" is a platform-agnostic binary analysis framework developed by the Computer Security Lab at UC Santa Barbara and their associated CTF team, Shellphish.

Steps:

1. In the file Project01.xlsx, find the title of the project that was allocated to you.
2. Visit the web site https://github.com/angr/angr
3. If "angr" is not installed on your machine follow the Installation instructions http://docs.angr.io/INSTALL.html
    a. If you are installing "angr" log the commands for its successful installation
4. Go to the link https://github.com/angr/angr-doc/tree/master/examples
5. Find the scripts and binary files for your project.
6. Prepare a document and presentation about your project.
    a. Pay attention to the EVALUATION CRITERIA as they are described in the table Project01.xlsx
7. Tentative date for both project deliveries is 12 November 2017

Discussion about your Project 01

For the project you will work with the suite of python libraries called “angr”. “angr” is a platform-agnostic binary analysis framework developed by the Computer Security Lab at UC Santa Barbara and their associated CTF team, Shellphish.

Discussion about your Project 01

What is angr?

angr is a multi-architecture binary analysis platform, with the capability to perform dynamic symbolic execution (like Mayhem, KLEE, etc.) and various static analyses on binaries. Several challenges must be overcome to do this. They are, roughly:

- Loading a binary into the analysis program.
- Translating a binary into an intermediate representation (IR).
- Translating that IR into a semantic representation (i.e., what it *does*, not just what it *is*).
- Performing the actual analysis. This could be:
  - A partial or full-program static analysis (i.e., dependency analysis, program slicing).
  - A symbolic exploration of the program's state space (i.e., "Can we execute it until we find an overflow?").
  - Some combination of the above (i.e., "Let's execute only program slices that lead to a memory write, to find an overflow.")

angr has components that meet all of these challenges. This book will explain how each one works, and how they can all be used to accomplish your evil goals.

Discussion about your Project 01

Steps:

1. In the file Project01.xlsx, find the title of the project that was allocated to you.

A list of scheduled topics for the Project 01

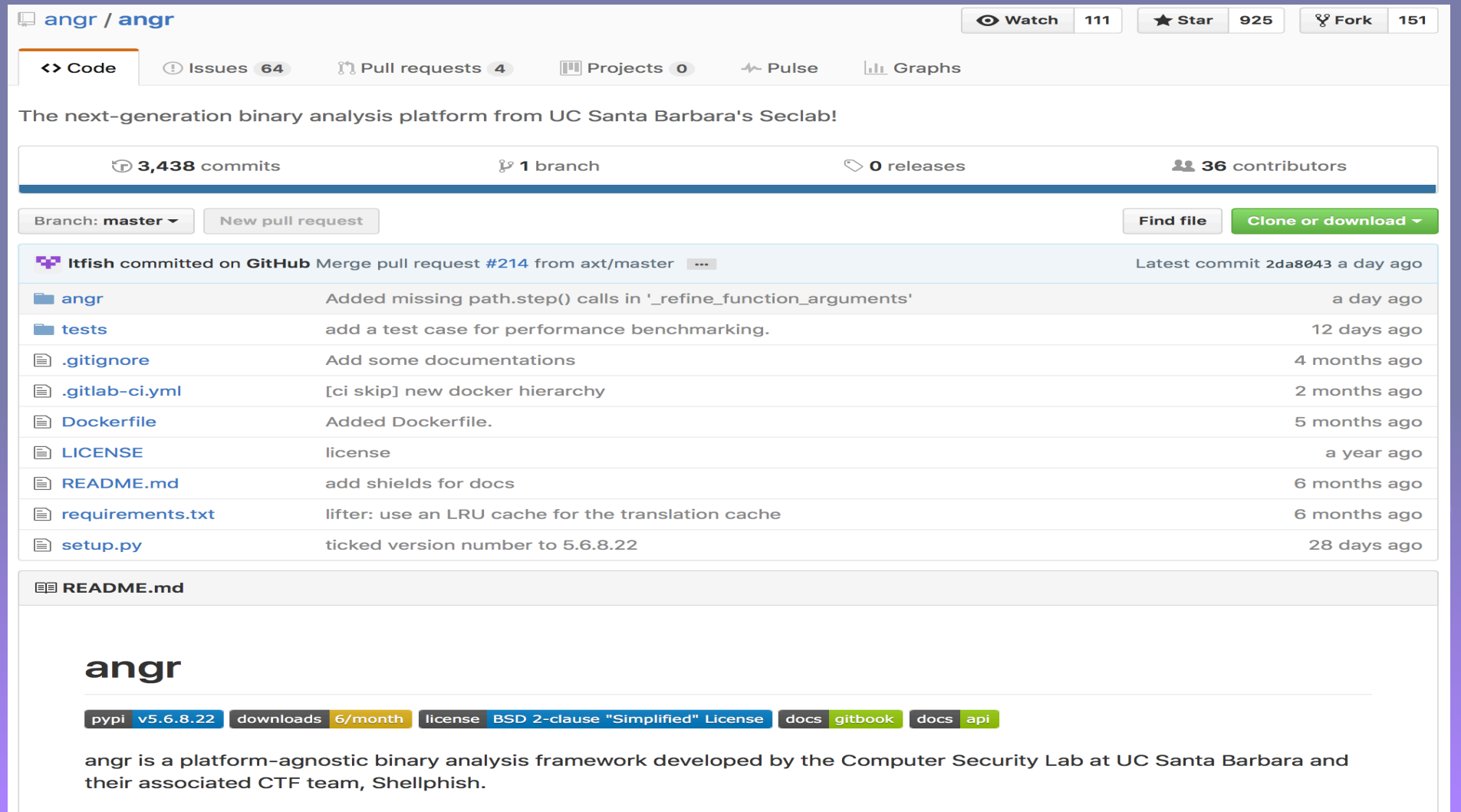
Group	Surename	Name	Project	EVALUATION CRITERIA: Description of the problem (max 3 pts)	Description of the "angr" tool (max 3 pts)	Decription of the idea(s) for solution (max 3 pts)	Description of the specific functions used in the solution (max 3 pts)	Practical demonstration how the solution works (max 3 pts)	Overall quality of the presentation (doc + slides) (max 5 pts)	Extra work (proposals how to extend the work, variants, ...) (max 3 pts)	Total
Project 01 01	Thomas	Skuggevik	0ctf_momo_3								0
	Erlend Solberg	Thorsrud									
Project 01 02	Marko	Savic	0ctf_trace								0
	Luis Tome	Pinto de Almeida Borges									
Project 01 03	Gaute Michel Maroni	Ferstad	9447_nobranch								0
	Luis Ricardo	RamÃ-rez Torres									
Project 01 04	Simon Markus Hoff	Olsen	CADET_00001								0
	Markus	Andersen									
Project 01 05	Martin Breivik	Smebye	CSCI-4968-MBE/challenges								0
	Odd Martin	Rognerud									
Project 01 06	Sigvart Midtsand	Hovland	ais3_crackme								0
	Brang Nu Bok	Tong									
Project 01 07	Ingve	SkÃ¥ra	android_arm_license_validation								0
	Anders	Liland									
Project 01 08	Emil Henry	Flakk	asisctffinals2015_fake								0
	Kevin	Kristiansen									
Project 01 09	Edwin Giancarlo	Vasquez Villano	asisctffinals2015_license								0
	Krishna	Shingala									
Project 01 10	Nina HoddÃ¸	BakÃ¥s	cmu_binary_bomb								0
	Joseph	Asamoah-Okyere									
Project 01 11	Ashmitha	Lokanath	codegate_2017-angrybird								0
	Amandeep Singh	Virdi									
Project 01 12	Michele	Zanchi	csaw_wyvern								0
	Sindre Elias	Solheim									
Project 01 13	Kazi Afroza	Sultana	defcamp_r100								0
	Thomas Michel	Conte									
Project 01 14	Per Oskar	Isdahl	defcamp_r200								0
	Magnus VitsÃ¸	BjÃ¸rnstad									
Project 01 15	Raghav	Khosla	defcon2016quals_baby-re_1								0
	Nina Tandrevold	Hermanrud									
Project 01 16	Edem	Avevor	defcon2017quals_crackme2000								0
	Matej	Spetko									
Project 01 17	Jesse	de Jonge	ekopartyctf2015_rev100								0
Project 01 18	Markus	Rauhut	ekopartyctf2016_rev250								0
	Martin	Kjellevand									
Project 01 19	Javier	Felipe Blanco	ekopartyctf2016_sokohashv2								0
	Nicolas Georges Romain	Roels									
Project 01 20	Rafael	LonskÃ¼	fauxware								0
	Matej	Mnoucek									
Project 01 21	Trygve	SÃ¸rensen	flareon2015_10								0
	Marius Oscar	Moe									
Project 01 22	Gianfranco	Fantappie	flareon2015_5								0
	Sebastiano	Cucinotta									
Project 01 23	Sindre	Trandum	google2016_unbreakable_0								0
	Kristoffer Alvern	Andersen									
Project 01 24	Aishwarya	Sreekanth	google2016_unbreakable_1								0
	Prasant Kumar	Sukumar									
Project 01 25	Peder Lind	Mangset	grub								0
	Anders	Nese									
Project 01 26	Kristine Larsen	Strand	hackcon2016_angry-reverser								0
	Ingvild Hovdelien	Midthaug									
Project 01 27	Erlend Moen	Hov	insomnihack_aeg								0
	Andreas BergstrÃ¸m	Aarnseth									
Project 01 28	Karen Birgitte Ljones	FjÃ¦restad	layer7_onlyone								0
	Kristin	Aaby									
Project 01 29	Vilde Elise SamnÃ¸y	Amundsen	mma_simplehash								0
	Maja Worren	LegernÃ¦s									
Project 01 30	Jakob Svennevik	Notland	secconquals2016_ropsynth								0
	Ken Siva	Lie									
Project 01 31	Ruben Schmidt	MÃ¤llberg	secuinside2016mbrainfuzz								0
	Valdemar Edvard Sandal	Rolfsen									
Project 01 32	HÃ¥vard	Kittelsen	securityfest_fairlight								0
	Martin BÃ¸rÃ¸	Nilsen									
Project 01 33	Eivind Skjelmo	Kristiansen	sharif7/rev50								0
	Anders Been	Wilhelmsen									
Project 01 34	Hans Kristian	StrÃ¸mme	simple_heap_overflow								0
	Henrik	Willett									

Discussion about your Project 01

2. Visit the web site https://github.com/angr/angr

Discussion about your Project 01

2. Visit the web site https://github.com/angr/angr



angr / angr

Watch 111 | Star 925 | Fork 151

<> Code | Issues 64 | Pull requests 4 | Projects 0 | Pulse | Graphs

The next-generation binary analysis platform from UC Santa Barbara's Seclab!

3,438 commits | 1 branch | 0 releases | 36 contributors

Branch: master | New pull request | Find file | Clone or download

Itfish committed on GitHub Merge pull request #214 from axt/master ... Latest commit 2da8043 a day ago

angr	Added missing path.step() calls in '_refine_function_arguments'	a day ago
tests	add a test case for performance benchmarking.	12 days ago
.gitignore	Add some documentations	4 months ago
.gitlab-ci.yml	[ci skip] new docker hierarchy	2 months ago
Dockerfile	Added Dockerfile.	5 months ago
LICENSE	license	a year ago
README.md	add shields for docs	6 months ago
requirements.txt	lifter: use an LRU cache for the translation cache	6 months ago
setup.py	ticked version number to 5.6.8.22	28 days ago

README.md

angr

pypi v5.6.8.22 | downloads 6/month | license BSD 2-clause "Simplified" License | docs gitbook | docs api

angr is a platform-agnostic binary analysis framework developed by the Computer Security Lab at UC Santa Barbara and their associated CTF team, Shellphish.

Discussion about your Project 01

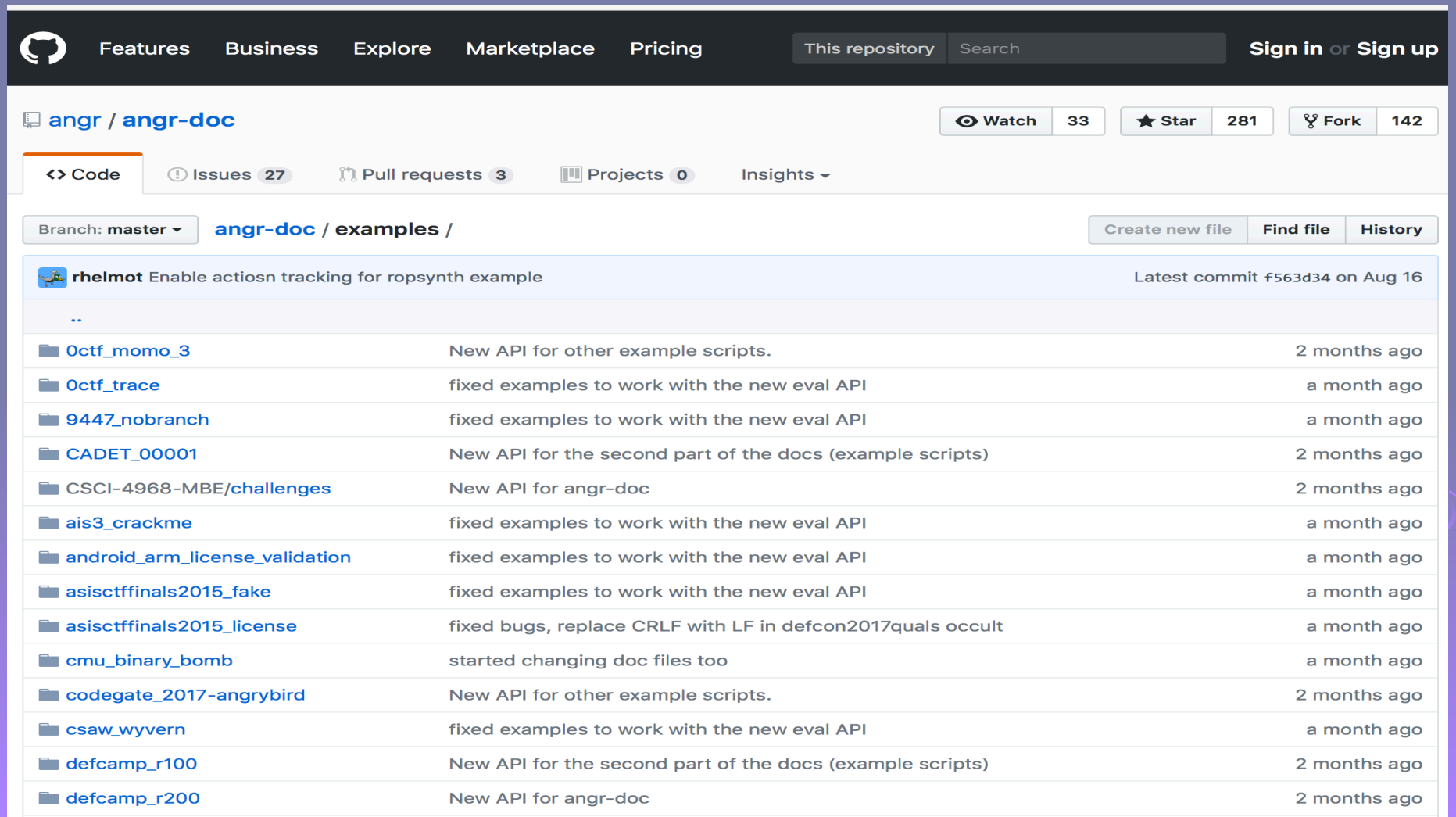
3. If “angr” is not installed on your machine follow the Installation instructions http://docs.angr.io/INSTALL.html
    a. If you are installing “angr” log the commands for its successful installation

Discussion about your Project 01

4. Go to the link https://github.com/angr/angr-doc/tree/master/examples

Discussion about your Project 01

4. Go to the link https://github.com/angr/angr-doc/tree/master/examples

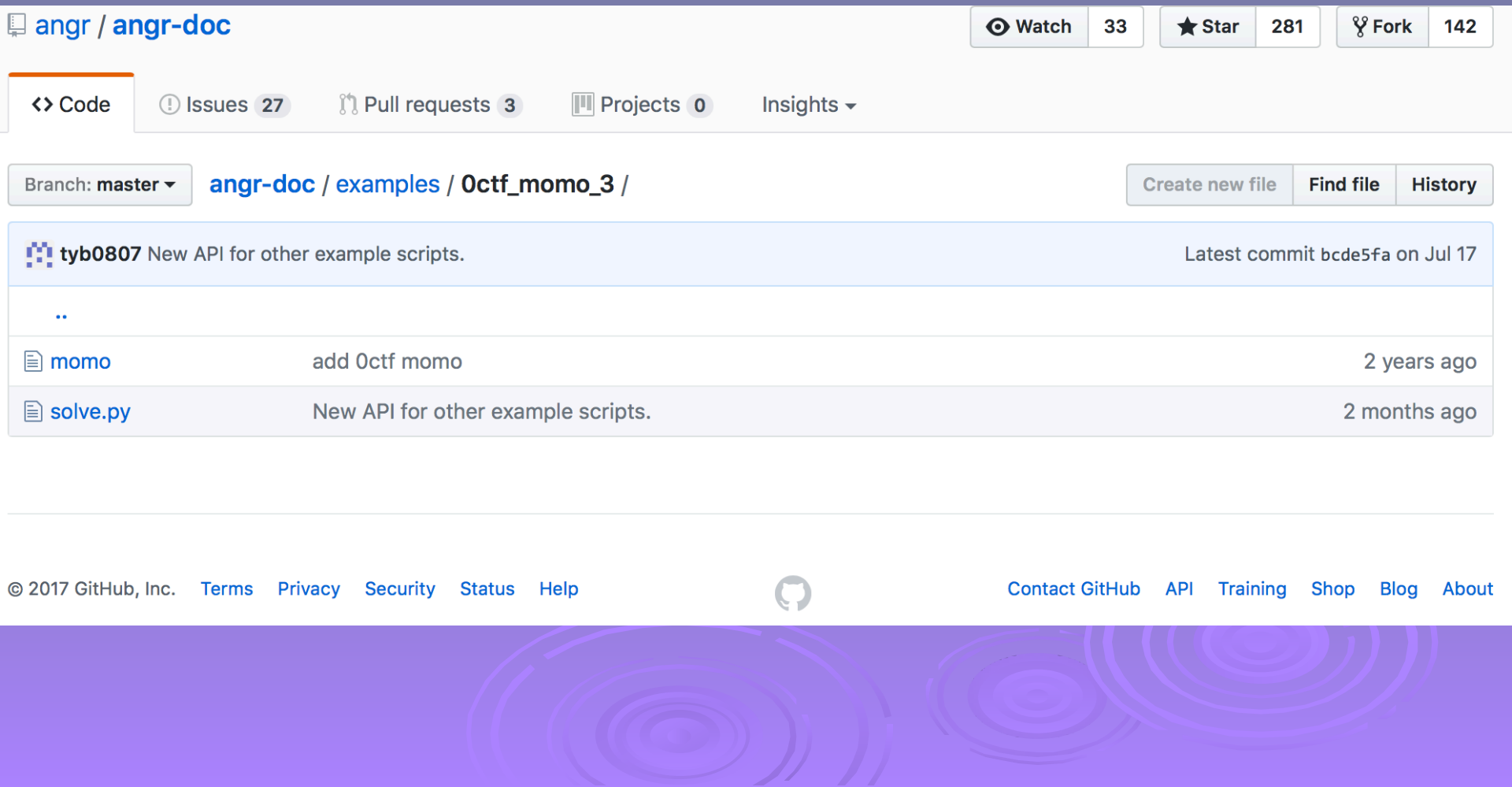


Discussion about your Project 01

5. Find the scripts and binary files for your project.

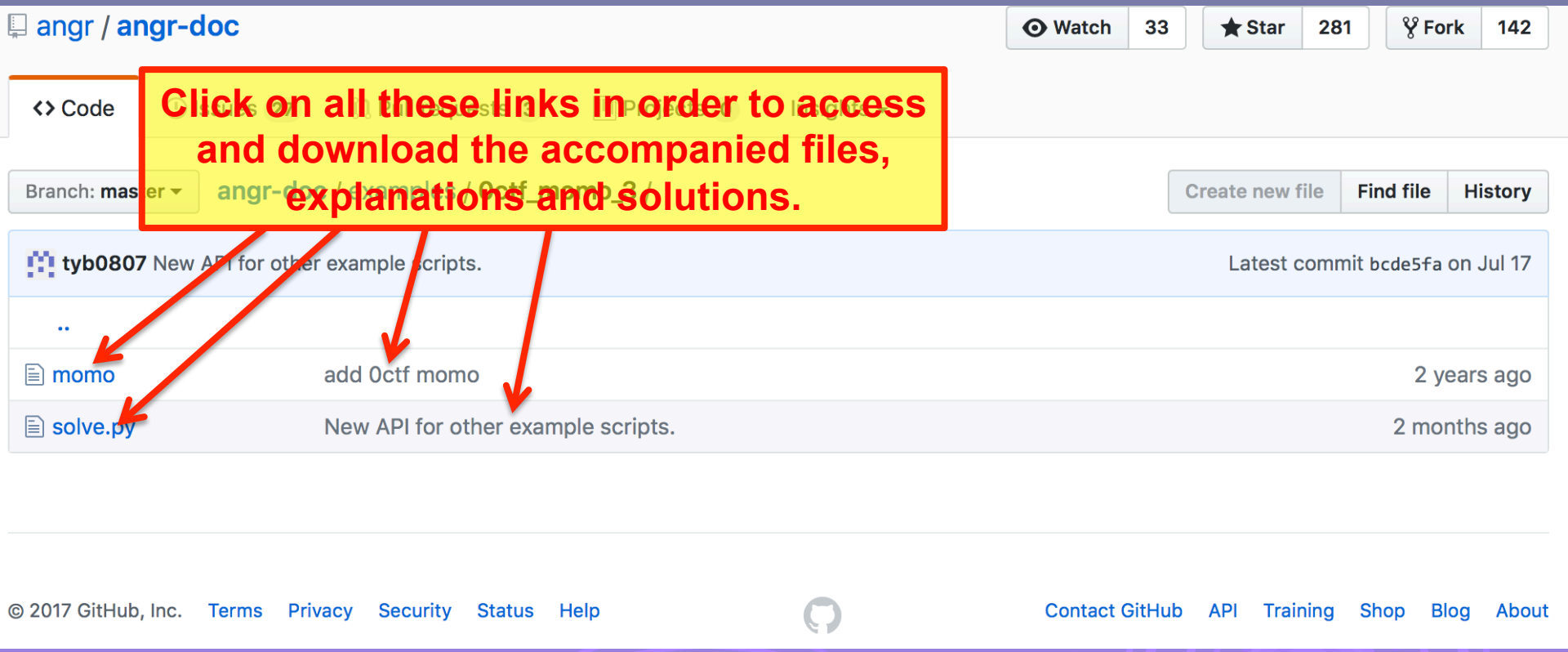
Discussion about your Project 01

5. Find the scripts and binary files for your project.



Discussion about your Project 01

5. Find the scripts and binary files for your project.



Discussion about your Project 01

6. Prepare a document and presentation about your project.
    a. Pay attention to the EVALUATION CRITERIA as they are described in the table Project01.xlsx

Discussion about your Project 01

6. Prepare a document and presentation about your project.
    a. Pay attention to the EVALUATION CRITERIA as they are described in the table Project01.xlsx

E	F	G	H	I	J	K	L
			EVALUATION CRITERIA				
Description of the problem (max 3 pts)	Description of the "angr" tool (max 3 pts)	Decription of the idea(s) for solution (max 3 pts)	Description of the specific functions used in the solution (max 3 pts)	Practical demonstration how the solution works (max 3 pts)	Overall quality of the presentation (doc + slides) (max 5 pts)	Extra work (proposals how to extend the work, variants, ...) (max 3 pts)	Total
							0

Discussion about your Project 01

7. Tentative date for both project deliveries is 12 November 2017

1. All groups will have to deliver their projects in a zip file: GroupXXProjectYY.zip in Blackboard.
2. Evaluation of your projects will be finished on 19 Nov 2017
3. Due to large number of students, there will be no oral presentations of all projects.
4. But the best evaluated projects will be posted on Blackboard

Installing and teting angr

(if these instructions do not work for your machine, you will have to search internet forums to figure out how to install angr properly)

- angr should be **installed and used inside** a so called "python virtual environment"
- So, first you should install virtualenv with

```
sudo apt-get install virtualenv
```

- Then you can use pip to install virtualenvwrapper with

```
pip install --user virtualenvwrapper
```

Installing and teting angr

(if these instructions do not work for your machine, you will have to search internet forums to figure out how to install angr properly)

- Then export the WORKON_HOME variable which contains the directory in which our virtual environments are to be stored with the following instructions

```
export WORKON_HOME=~/.virtualenvs
mkdir $WORKON_HOME
echo "export WORKON_HOME=$WORKON_HOME" >> ~/.bashrc
echo "source ~/.local/bin/virtualenvwrapper.sh" >> ~/.bashrc
echo "export PIP_VIRTUALENV_BASE=$WORKON_HOME" >> ~/.bashrc
source ~/.bashrc
```

Installing and teting angr

(if these instructions do not work for your machine, you will have to search internet forums to figure out how to install angr properly)

- Test if it works with the following instructions

```
mkvirtualenv -p python2.7 test
```

- You will see that the environment will be set up, and your prompt now includes the name of your active environment in parentheses. Then run the following command

```
python -c "import sys; print sys.path"
```

- you should see a lot of /home/user/.virtualenv/... because it now doesn't use your system site-packages.
- You can deactivate your environment by running

```
deactivate
```

Installing and teting angr

(if these instructions do not work for your machine, you will have to search internet forums to figure out how to install angr properly)

- Now, let us again enter the virtual environment with

```
workon test
```

- **Inside the virtual environment you install** angr with the following command

```
pip install angr
```

- This will take some time.

Installing and teting angr

(if these instructions do not work for your machine, you will have to search internet forums to figure out how to install angr properly)

- Now, go again to the Examples folder of angr and download three files for the example **Fauxware**

angr examples

To help you get started with angr, we've created several examples. These mostly stem from CTF problems solved with angr by Shellphish. Enjoy!

Introduction example - Fauxware

This is a basic script that explains how to use angr to symbolically execute a program and produce concrete input satisfying certain conditions.

Binary, source, and script are found here.

Installing and teting angr

(if these instructions do not work for your machine, you will have to search internet forums to figure out how to install angr properly)

- Now, go again to the Examples folder of angr and download three files for the example **Fauxware**

angr examples

To help you get started with angr, we've created several examples. These mostly stem from CTF problems solved with angr by Shellphish. Enjoy!

Introduction example - Fauxware

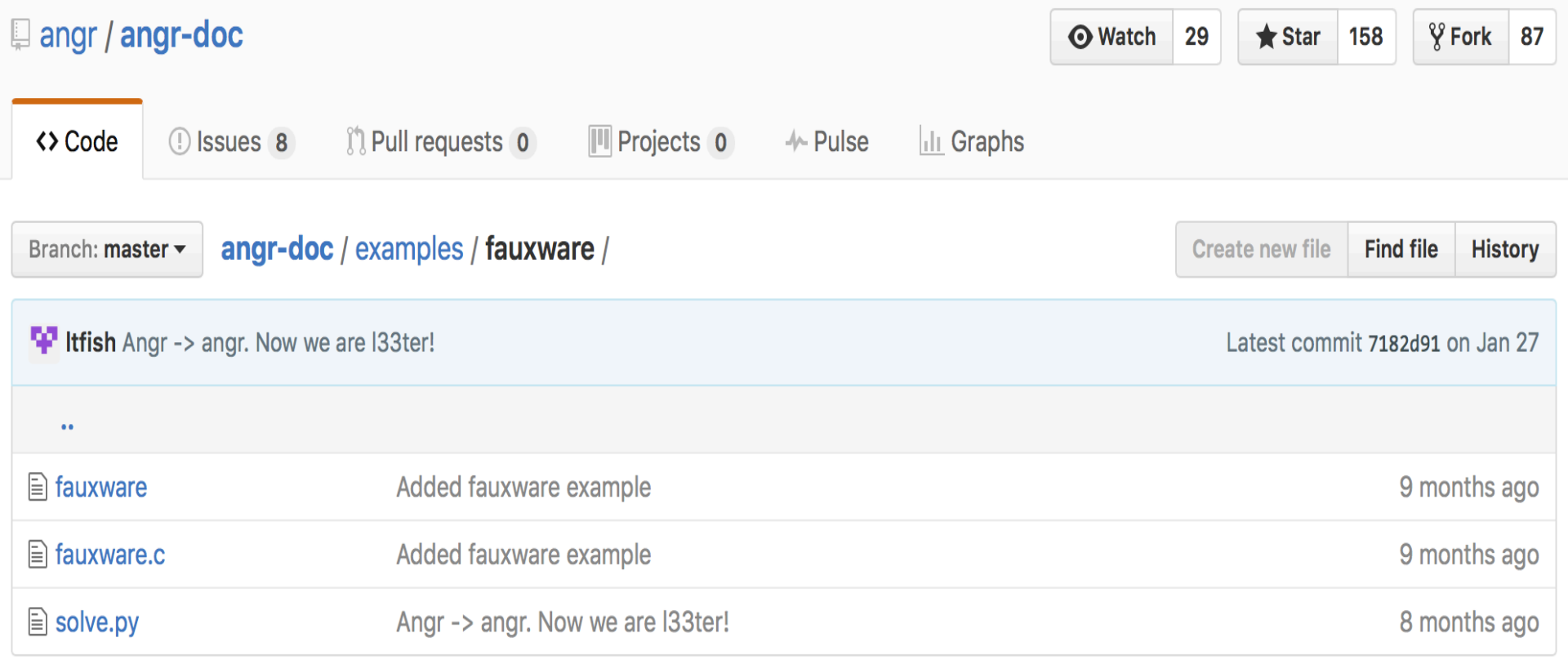
This is a basic script that explains how to use angr to symbolically execute a program and produce concrete input satisfying certain conditions.

Binary, source, and script are found here.

Installing and teting angr

(if these instructions do not work for your machine, you will have to search internet forums to figure out how to install angr properly)

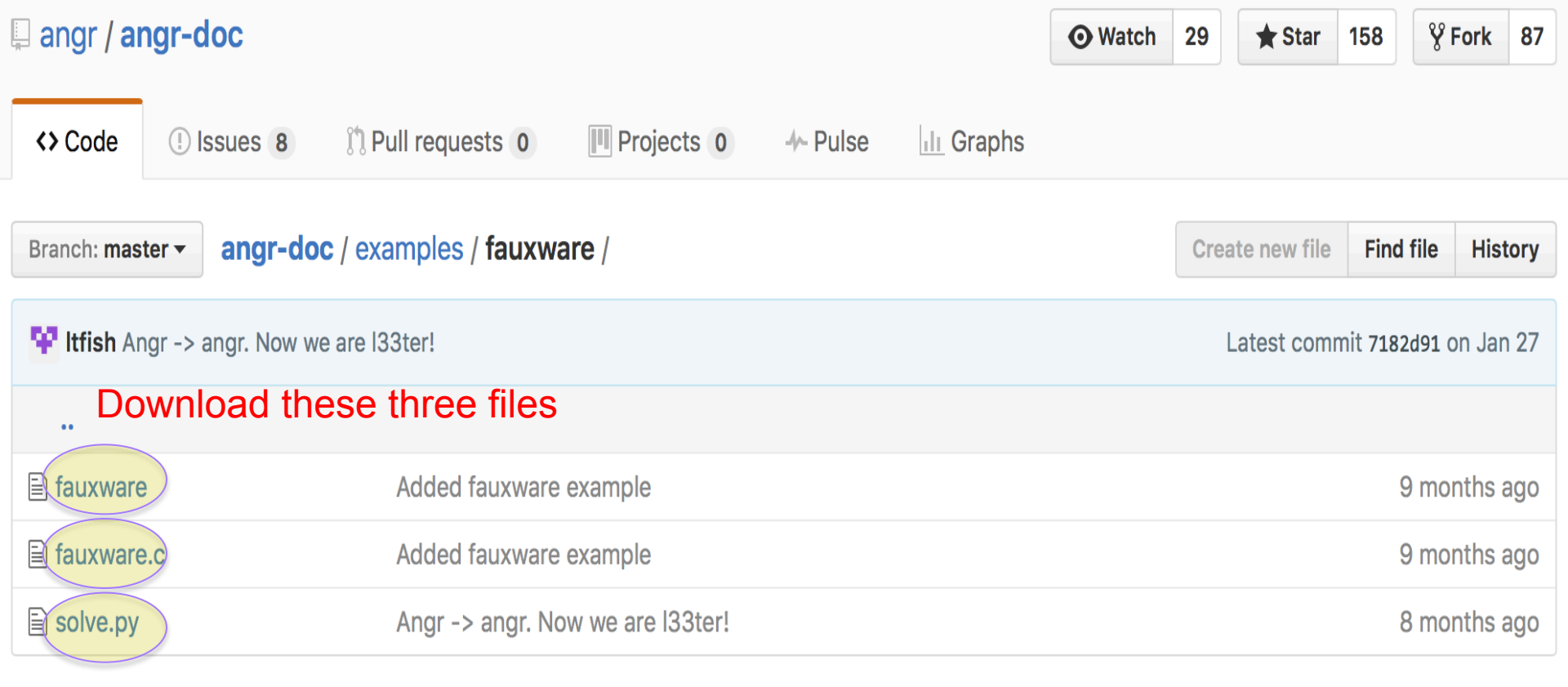
- Now, go again to the Examples folder of angr and download three files for the example **Fauxware**



Installing and teting angr

(if these instructions do not work for your machine, you will have to search internet forums to figure out how to install angr properly)

- Now, go again to the Examples folder of angr and download three files for the example **Fauxware**



Installing and teting angr

(if these instructions do not work for your machine, you will have to search internet forums to figure out how to install angr properly)

- Still in the workon environment (test) make solve.py executable

```
chmod 755 solve.py
```

- Start

```
./solve.py
```

- you should see a response like

```
/root/.virtualenvs/test/local/lib/python2.7/site-packages/pyvex/block.py:75: UserWarning: implicit cast from 'char *' to a different pointer type: will be forbidden in the future (check that the types are as you expect; use an explicit ffi.cast() if they are correct)  1)SOSNEAKY
```

Doing the exercises from Chapter 3

- Use the files from the folder Chapter3
  - sniffer.py
  - sniffer_ip_header_decode.py
  - sniffer_with_icmp.py
  - scanner.py

Doing the exercises from Chapter 3

- In PyCharm open a new project Ch03
- Add an empty python file sniffer.py (delete the author command in the beginning of the file)
- Open the file sniffer.py from Chapter 3 and copy-paste its content to the file sniffer.py of the Ch03 project

Doing the exercises from Chapter 3

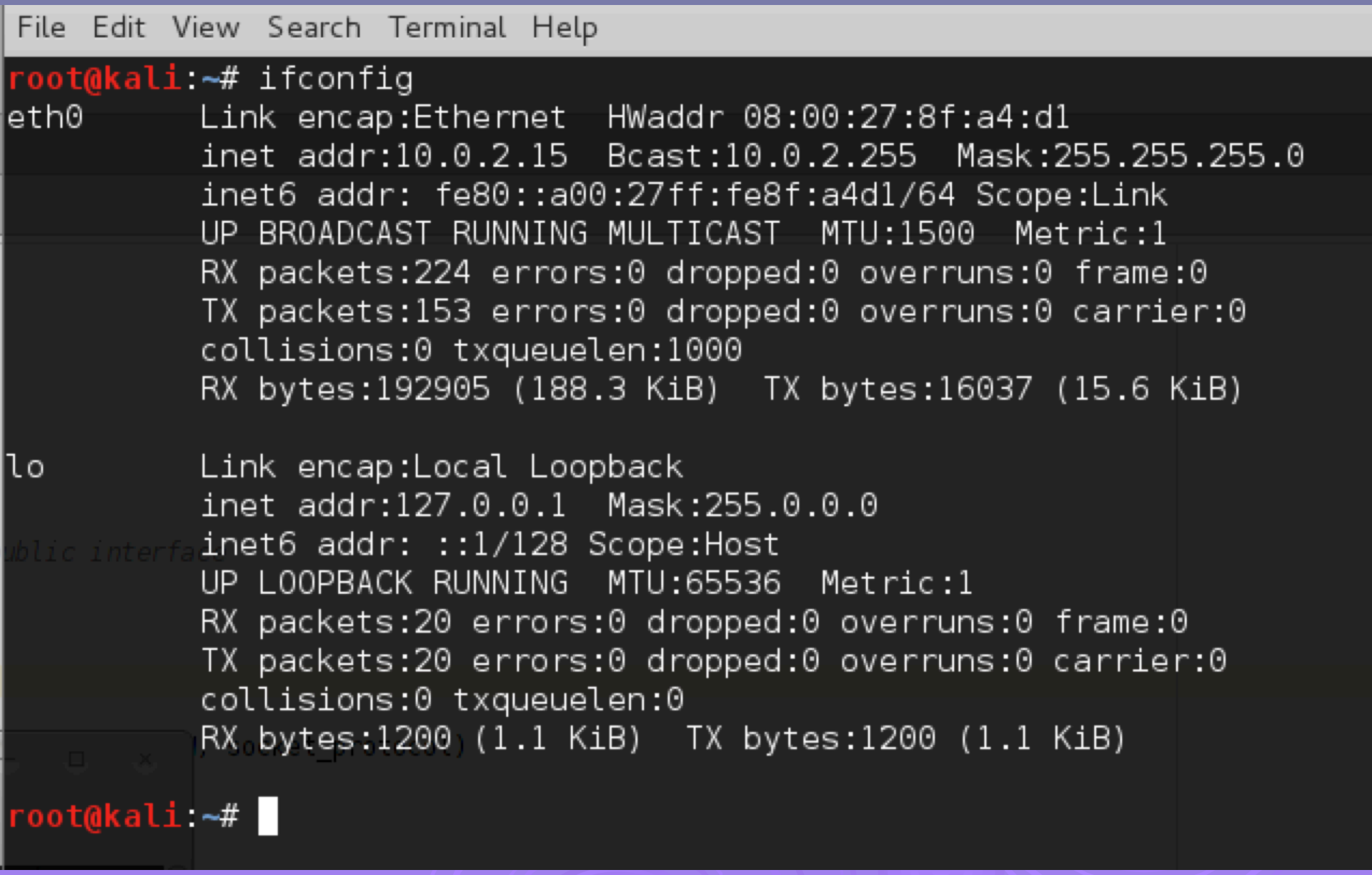
- For our installed Kali machines you should insert the following line AS THE FIRST LINE at sniffer.py

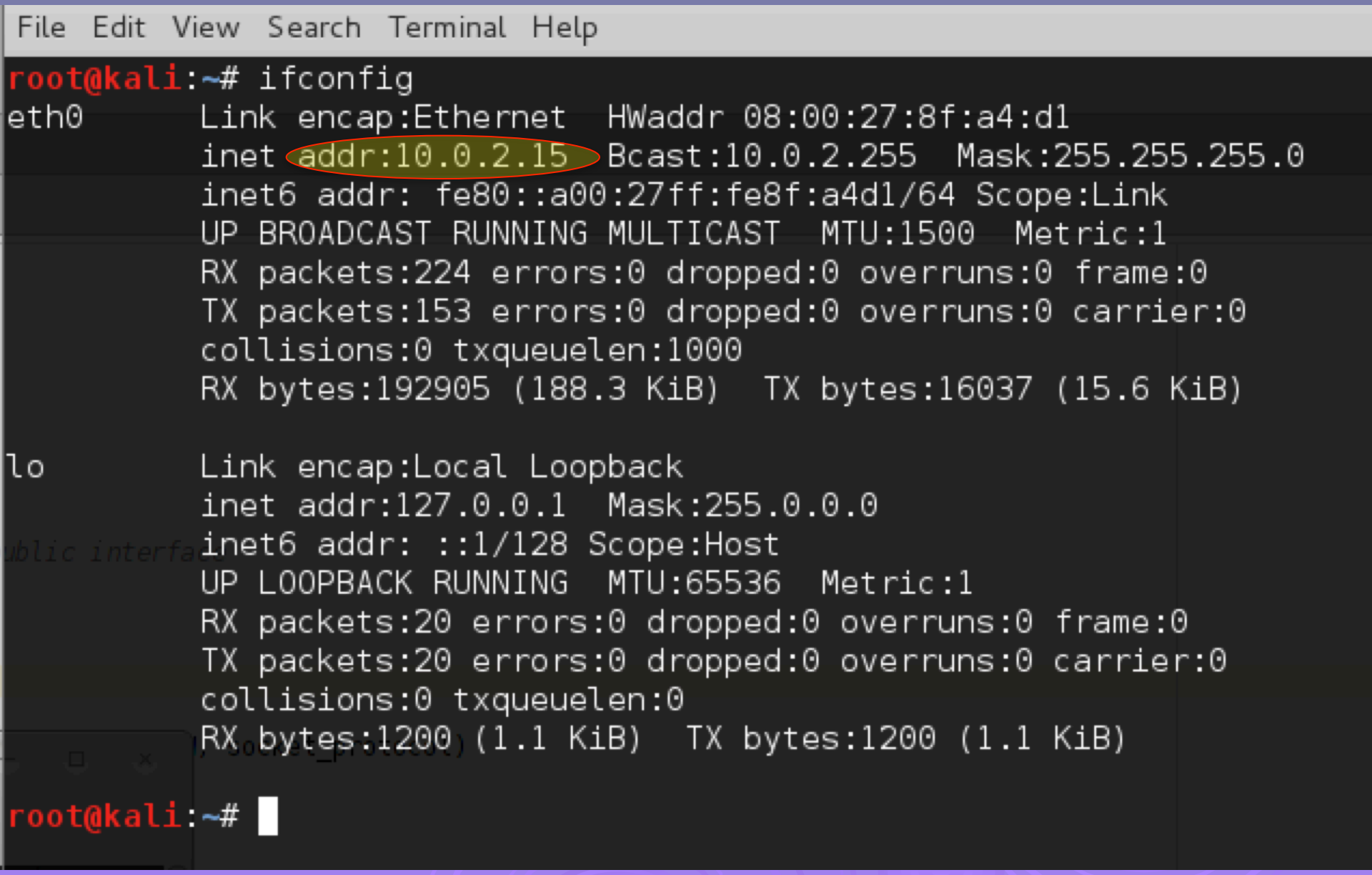
```
#!/usr/bin/python2.7
```

Doing the exercises from Chapter 3

- In one terminal find out what is the IP address of your machine with

```
# ifconfig
```


File Edit View Search Terminal Help
root@kali:~# ifconfig
eth0 Link encap:Ethernet HWaddr 08:00:27:8f:a4:d1
inet addr:10.0.2.15 Bcast:10.0.2.255 Mask:255.255.255.0
inet6 addr: fe80::a00:27ff:fe8f:a4d1/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:224 errors:0 dropped:0 overruns:0 frame:0
TX packets:153 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:192905 (188.3 KiB) TX bytes:16037 (15.6 KiB)
lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:65536 Metric:1
RX packets:20 errors:0 dropped:0 overruns:0 frame:0
TX packets:20 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:1200 (1.1 KiB) TX bytes:1200 (1.1 KiB)
root@kali:~#


File Edit View Search Terminal Help
root@kali:~# ifconfig
eth0 Link encap:Ethernet HWaddr 08:00:27:8f:a4:d1
inet addr:10.0.2.15 Bcast:10.0.2.255 Mask:255.255.255.0
inet6 addr: fe80::a00:27ff:fe8f:a4d1/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:224 errors:0 dropped:0 overruns:0 frame:0
TX packets:153 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:192905 (188.3 KiB) TX bytes:16037 (15.6 KiB)
lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:65536 Metric:1
RX packets:20 errors:0 dropped:0 overruns:0 frame:0
TX packets:20 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:1200 (1.1 KiB) TX bytes:1200 (1.1 KiB)
root@kali:~#

Doing the exercises from Chapter 3

- Change the line

```
# host to listen on
host = "192.168.0.196”
```

- With the address of your machine

```
# host to listen on
host = ”10.0.2.15”
```

Doing the exercises from Chapter 3

- Save the file and open two terminal windows
- Go to folder /PycharmProjects/Ch03
- Make the sniffer.py executable with:

```
# chmod 755 sniffer.py
```

- Start sniffer.py first

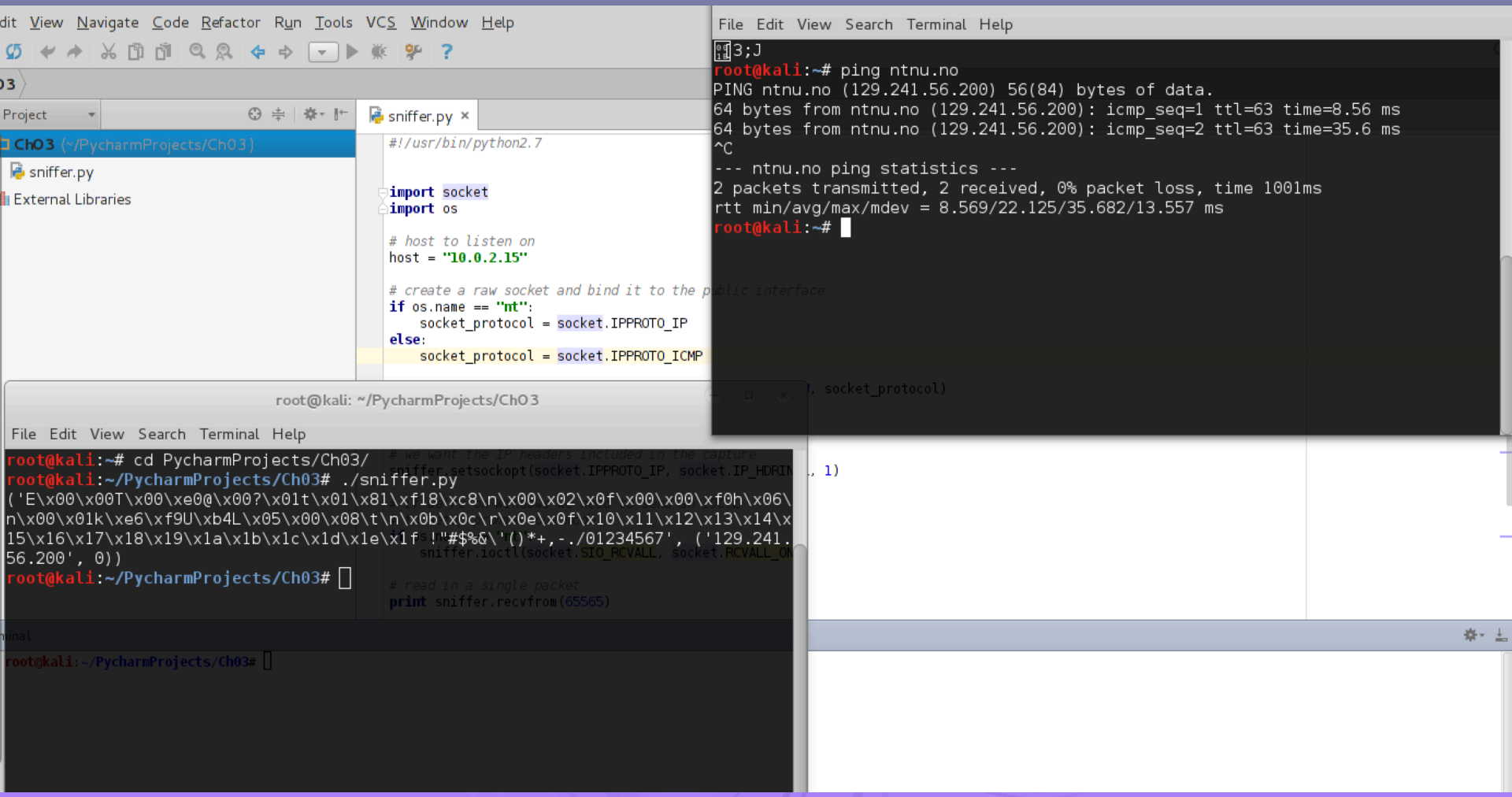
```
# ./sniffer.py
```

- Then in the other terminal ping ntnu.no

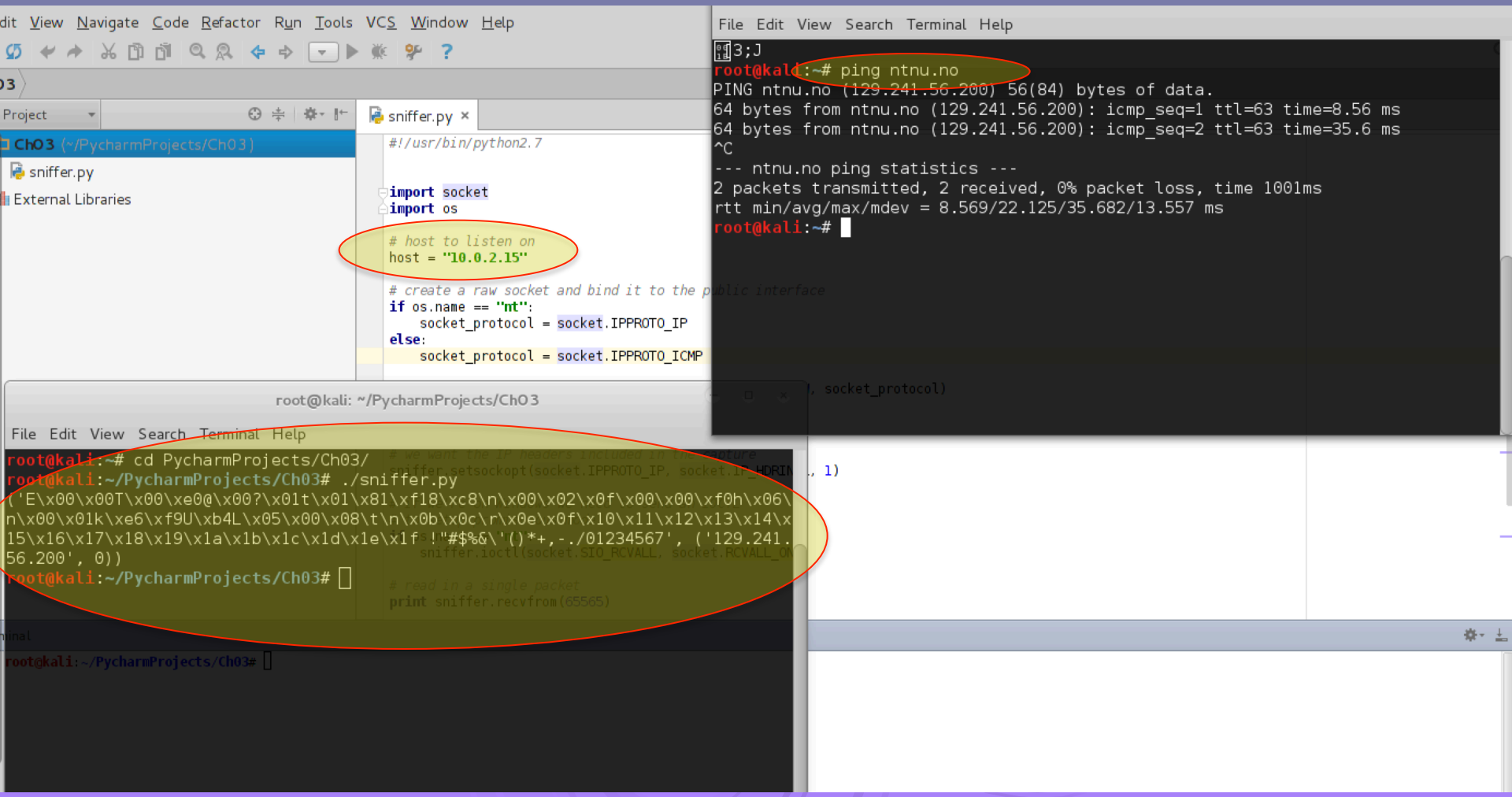
```
# ping ntnu.no
```

- Press Ctrl C to stop pinging

Doing the exercises from Chapter 3



Doing the exercises from Chapter 3



Doing the exercises from Chapter 3

- Repeat similar actions as you did with sniffer.py for the following Python programs in this order
  - sniffer_ip_header_decode.py
  - sniffer_with_icmp.py
  - scanner.py (NOTE: try to run this script on a real machine, not on a virtual box. In virtual box there will not be many other active IP addresses)
- Follow the instructions from the textbook

Doing the exercises from Chapter 3

- Repeat similar actions as you did with sniffer.py for the following Python programs in this order

If you run the scripts in Windows, make sure that you start cmd.exe as Administrator

machine, not on a virtual box. In virtual box there will not be many other active IP addresses)

- Follow the instructions from the textbook

If you get an error about a small buffer size in scanner.py

- With scanner.py it might happen that you will get error messages that the buffer size of 20 is too small (it should be 32).
- This can happen if you run the script on 64-bit operating system.
- In that case do the following:

If you get an error about a small buffer size in scanner.py

- In class IP(Structure): change

```
("src", c_ulong),
("dst", c_ulong)
```

- to this:

```
("src", c_uint32),
("dst", c_uint32)
```

-

If you get an error about a small buffer size in scanner.py

- Also change these two lines

```
self.src_address = socket.inet_ntoa(struct.pack("<L",self.src))
self.dst_address = socket.inet_ntoa(struct.pack("<L",self.dst))
```

to this:

```
self.src_address = socket.inet_ntoa(struct.pack(”@I",self.src))
self.dst_address = socket.inet_ntoa(struct.pack(”@I",self.dst))
```

Doing the exercises from Chapter 3

- Extra work:
- Learn about the socket library and its functions
- In your Linux box try to play with

  import socket as socket_mod
- and find out how to use

```
socket = socket_mod.socket(socket_mod.AF_PACKET, socket_mod.SOCK_RAW, socket_mod.IPPROTO_IP)
```

After finishing exercises of L04, try this CTF challenge

Lecture 04

After finishing exercises of L04, try this CTF challenge

Without knowing the secret key, can you decrypt the file **AttackAtTheDawn.txt.enc** and retrieve the vital information about the flag inside that file?

After finishing exercises of L04, try this CTF challenge

Build Content Assessments Tools

An Excel file was encrypted with the following command: openssl enc -aes-128-cbc -in Workbook.xlsx -out Workbook.xlsx.enc -K xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx -iv 0123456789abcdef0123456789abcdef

This file was encrypted with the same key in ofb mode. The IV for this encrypton was picked up from the encrypted Excel file.