# Računalna forenzika

## Lab 3

**1.)**

import os

import pandas as pd

import hashlib

import magic

import mimetypes

import time

dir\_path = 'E:'

file\_names = []

extensions = []

md5s = []

sha1s = []

sha256s = []

magic\_numbers = []

mag\_obj = magic.Magic(mime=True)

extension\_matches = []

creation\_times=[]

modification\_times=[]

access\_times=[]

for file in os.listdir(dir\_path):

# check if the file is a regular file (i.e., not a directory)

if os.path.isfile(os.path.join(dir\_path, file)):

# if so, add the file name to the list

(file\_name, extension) = os.path.splitext(file)

file\_names.append(file\_name)

extensions.append(extension)

with open(os.path.join(dir\_path, file), 'rb') as f:

data = f.read()

magic\_number = mag\_obj.from\_file(os.path.join(dir\_path, file))

magic\_numbers.append(magic\_number)

md5hash = hashlib.md5(data).hexdigest()

md5s.append(md5hash)

sha1hash = hashlib.sha1(data).hexdigest()

sha1s.append(sha1hash)

sha256hash = hashlib.sha256(data).hexdigest()

sha256s.append(sha256hash)

if extension.lower() == '':

extension\_matches.append(False)

elif mimetypes.guess\_type('test'+extension.lower())[0] in magic\_number.lower():

extension\_matches.append(True)

else:

extension\_matches.append(False)

creation\_time=os.path.getctime(os.path.join(dir\_path, file))

creation\_times.append(time.ctime(creation\_time))

modification\_time=os.path.getmtime(os.path.join(dir\_path, file))

modification\_times.append(time.ctime(modification\_time))

access\_time=os.path.getatime(os.path.join(dir\_path, file))

access\_times.append(time.ctime(access\_time))

df = pd.DataFrame({'file\_name': file\_names, 'extension': extensions,

'md5': md5s, 'sha1': sha1s, 'sha256': sha256s, 'magic\_numbers': magic\_numbers, 'extension\_matches': extension\_matches, 'creation\_times':creation\_times,'modification\_times':modification\_times, 'access\_times':access\_times})

print(df)

print(df['sha1'])

print(df['access\_times'])

print(df['modification\_times'])

**2.)**

import re

import datetime

log\_path='setupapi.dev2.log'

usb\_devices={

'device\_vendor\_id':[],

'device\_product\_id':[],

'device\_instance\_id':[],

'event\_time':[]

}

usb\_regex=r'^>>> \[Device Install.\*#(Disk&Ven\_[A-Za-z0-9]+)&(Prod\_([\w\s\S]+?))&(Rev\_([\w\s\S]+?))#([\w\s\S]+?)#.\*\]'

# Read the contents of the setupapi.dev.log file

with open(log\_path, "r") as log\_file:

# Store information about each USB device in a dictionary

for line in log\_file:

line\_text=next(log\_file)

matched\_object=re.search(usb\_regex,line)

if(matched\_object!=None):

event\_time=line\_text.split("start ")[1].strip()

usb\_devices["device\_vendor\_id"].append(matched\_object.groups()[0])

usb\_devices["device\_product\_id"].append(matched\_object.groups()[1])

usb\_devices["device\_instance\_id"].append(matched\_object.groups()[2])

usb\_devices["event\_time"].append(event\_time)

print(f"Printing \n {usb\_devices['device\_vendor\_id']} \n {usb\_devices['device\_product\_id']} \n {usb\_devices['device\_instance\_id']} \n {usb\_devices['event\_time']}")