ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

**NEUMANN JÁNOS**
**INFORMATIKAI KAR**


# SZAKDOLGOZAT


| | | |
|---|---|---|
| **OE-NIK** | Hallgató neve: | **Alföldi Péter** |
| **2022** | Hallgató törzskönyvi száma: | **T/008048/FI12904/N** |

Óbudai Egyetem
Neumann János Informatikai Kar
Biomatika és Alkalmazott Mesterséges Intelligencia Intézet

# SZAKDOLGOZAT FELADATLAP

Hallgató neve: **Alföldi Péter**
Törzskönyvi száma: T/008048/FI12904/N
Neptun kódja: [redacted]

A dolgozat címe:

**Zombihálózatok elleni védelem**
**Protection against zombie networks**

Intézményi konzulens: Dr. Póser Valéria
Külső konzulens:

Beadási határidő: 2022. május 15.

A záróvizsga tárgyai: Informatikai rendszerek üzemeltetésének biztonsága
IT hálózatbiztonság

**A feladat**

Napjainkban megszokott, hogy a vállalatok és az átlagos felhasználók is folyamatosan használható internet kapcsolattal rendelkeznek. Az informatikai eszközök fejlődésével és az internet csatlakozáshoz szükséges szolgáltatások könnyebb és gyorsabb elérhetőségével egyre több és több eszköz kapcsolódik a világhálóra, napról-napra növelve ezzel a kártékony megoldások célterületeit.
A szakdolgozat célja megvizsgálni, hogy a kártékony informatikai megoldások egyik fajtája, a zombihálózat mit jelent és milyen veszélyeket hordoz magában, hogyan válhat egy eszköz a zombihálózat egyik elemévé, és hogyan történhet a védekezés ellene.
A hallgató feladata bemutatni a zombihálózatok kártékonyságát, kialakulásukat és működésüket, valamint az azonosításuk lehetőségeit és az ellenük lehetséges védekezés módszereit. Feladat továbbá egy lehetséges védekezési megoldás kidolgozása, a szükséges alkalmazás megtervezése, implementálása és tesztelése.

**A dolgozatnak tartalmaznia kell**:

- a zombihálózat fogalmának részletes bemutatását, veszélyeit, és működésének lényegét,
- a zombihálózat elleni védekezés lehetőségeinek bemutatását,
- a tesztkörnyezet kialakításának szempontjait és felépítését,
- az elképzelt védekezési forma, az elkészülő alkalmazás specifikációját, működési elvét,
- az elkészülő alkalmazás védekezési mechanizmusának bemutatását, illetve a védekezési mechanizmus kiiktatásának lehetőségét,
- továbbfejlesztési lehetőségeket.

ÓBUDAI EGYETEM * Neumann János Informatikai Kar * 1. Budapest

Ph.

.....
Dr. Eigner György
mb. intézetigazgató

A szakdolgozat elévülésének határideje: **2024. május 15.**
(OE TVSz 55.§ szerint)

A dolgozatot beadásra alkalmasnak tartom:

.....
külső konzulens

.....
intézményi konzulens

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

# HALLGATÓI NYILATKOZAT

Alulírott hallgató kijelentem, hogy a szakdolgozat / diplomamunka saját munkám eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült szakdolgozatomban / diplomamunkámban található eredményeket az egyetem és a feladatot kiíró intézmény saját céljára térítés nélkül felhasználhatja.

Budapest, 2022. ÁPRILIS 30.

Alsóki Ádám

.....

hallgató aláírása

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

# KONZULTÁCIÓS NAPLÓ

Hallgató neve: Alföldi Péter

Neptun Kód: [redacted]

Tagozat: Levelező

Telefon:

Levelezési cím (pl.: lakcím): [redacted]

Projektmunka címe magyarul:

Zombihálózatok elleni védelem

Projektmunka címe angolul:

Protection against zombie networks

Intézményi konzulens: Dr. Póser Valéria

Külső konzulens:

Kérjük, hogy az adatokat nyomtatott nagy betűkkel írja!

| Alk. | Dátum | Tartalom | Aláírás |
|---|---|---|---|
| 1. | 2022.03.04. | Előző beszámoló értékelése alapján tartalom módosítás egyeztetése. | |
| 2. | 2022.05.05. | Tartalmi és formai áttekintés, egyeztetés. | |
| 3. | 2022.05.08. | Javasolt módosítások elvégzése, azok bemutatása. | |
| 4. | 2022.05.09. | Újbóli egyeztetés, javasolt módosítások elvégzése, javítása. | |

A Konzultációs naplót összesen 4 alkalommal, az egyes konzultációk alkalmával kell láttamoztatni bármelyik konzulenssel.

A hallgató a Projektlabor 1. / 2. / 3. / Záródolgozati projekt[1] tantárgy követelményét teljesítette, beszámolóra/védésre[2] bocsátható. A javasolt érdemjegy: jeles (5)

.....

Intézményi konzulens

Budapes, 2020. 05. 12.

---

[1] A megfelelő bekarikázandó/aláhúzandó!

[2] A megfelelő aláhúzandó!

KÖSZÖNETNYILVÁNÍTÁS

Ezúton szeretnék köszönetet mondani az oktatásban részt vevő tanáraimnak, ügyintézőknek és szervezőknek, hogy munkájuk eredményeként összefogott tudást szerezhettem, külön megköszönve Árgyelán Mária igazgatási ügyintéző képzés alatti folyamatos támogatását.

Köszönöm Albininé Budavári Edinának és Sziklay Péternek az információbiztonság területén eltöltött éveket, a barátságot, a közös munkákat, amelyek végzése közben tanulhattam tőlük.

Külön köszönöm Hambalkó Balázsnak és Enikőnek a több éve tartó barátságot és a közös munkát. Köszönet Balázsnak az inspiráló megoldásokért, amelyekkel rávezetett, mit jelent az IT Security, hol van a "white hat" és a "black hat" határa, valamint azért a szemléletmódért, ami security-hez szükséges. [www.balasec.com]

Végül köszönöm családomnak a tanuláshoz nyújtott megértést és támogatást.

# Tartalomjegyzék

# 1. BEVEZETÉS

Szakdolgozatomban arra keresek választ, hogy a zombihálózat - mint kártékony informatikai fenyegetés - ellen milyen módon védekezhetünk. Lehetséges-e egyedi, saját módon védekezni ellene. Ha lehetséges, akkor milyen megoldással valósítható meg a szükséges védelem.

Először bemutatom a zombihálózatok világát, mi jellemzi őket, milyen veszélyt hordoznak magukban.

Következő témaként a védekezési lehetőségek bemutatása érdekében az informatika területeiről (informatikai biztonság, információbiztonság, informatikai üzemeltetés) gyűjtöm össze a lényeges információkat és elemzem azok tárgyhoz tartozó vonatkozásait az egyedi védekezés megvalósíthatóságának keresése céljából.

A saját logikájú védekezés lehetőségeként bemutatok egyfajta mechanizmust, amely alkalmas a zombitevékenység megakadályozására, valamint megismerhető a megvalósításhoz vezető út: teszt- és programkörnyezet, alkalmazás fejlesztés.

Bemutatok egy lehetőséget is, amellyel a kitalált és megvalósított védekezés megkerülhető, egyfajta "hekkelést", amelyet használva elérhetném a kártékony kódom futását leállítás esetén is.

Végül a megépített alkalmazás fejlesztésének további irányait is keresem: milyen elvárt működéshez milyen fejlesztések szükségesek.

# 2. PROBLÉMAFELVETÉS

Napjainkban megszokott, hogy a vállalatok és az otthoni felhasználók is folyamatosan elérhető és használható internetkapcsolattal rendelkeznek, a szervezeti kommunikációt biztosító levelező szerverektől egészen a kezünkben tartott okostelefonokig. A technika rohamos fejlődésével és a piaci felvásárlóerő megcélzásával újabb és újabb eszközbe kerül beépítésre valamilyen informatikai hálózati csatoló felület. Ilyenek például az intelligens otthonok kellékei : programozható robotporszívó és robotfűnyíró.

Az eszközök fejlődésével és az internet csatlakozáshoz szükséges szolgáltatások könnyebb és gyorsabb elérhetőségével egyre több és több eszköz kapcsolódik a világhálóra, naprólnapra növelve ezzel a kártékony megoldások célterületeit.

Ezek az otthoni okos eszközök nyitott kapui lehetnek akár rosszindulatú tevékenységeknek is. Tudtunkon kívül adatokat küldenek az irányító felhőjükbe, kommunikálnak más eszközökkel, jobb esetben azért, mert a működésükhoz szükséges ("önvezető" - újabb generációs, vezetést támogató autók), nem feltételezve szándékos adatgyűjtést a gyártó vállalat, iparág vagy nemzet részéről.

Jól megszervezett és kivitelezett, célzott támadás esetén valószínű észre sem vesszük az előlünk elrejtett támadási formát. Elég néhány elnézett kattintás a képernyőn látható böngészőben, vagy egy nem hivatalos forrásból szerzett alkalmazás, sőt még a Windows operációs rendszerek fiókfelügyeletét is felül lehet bírálni a rendszergazda jogosultsági szintű felhasználói profillal a gyorsabb, könnyebb működés érdekében. Ez esetben a leggyengébb láncszem manuális közreműködését már csak a naprakész végpont- és határvédelmi rendszerek elemei (vírusvédelem, tűzfal) tudják ellensúlyozni. Amennyiben pedig sikeresen lefut az ember által engedélyezett kártékony kód, akkor az eszköz egy zombihálózat, avagy botnet tagjává is válhat.

Az otthoni felhasználású eszközök informatikai védelme természetesen ugyanolyan fontos, de amíg egy otthoni számítógépet megfertőződés esetén újra lehet telepíteni és maximum a saját, biztonsági másolattal nem rendelkező adatok (fényképek, levelezések) vesznek el, addig egy vállalatnál egészen más dimenziókban kell gondolkodni, nem megengedhető a tartományi számítógép megfertőződése. Például, ha a vállalat domain név regisztrációjakor megadott publikus és fix IP cím botnet tevékenység miatt a szolgáltató által tiltásra kerül, megszűnik a szervezet Internet irányú elérhetősége, kommunikációja a tiltás idejére. Védekezési technikaként a megemlített példában szereplő probléma gondos

informatikai tervezéssel megelőzhető úgy, hogy legalább két IP címmel rendelkezzen a vállalat, és a domain regisztráció alkalmával a mutató rekordok esetén az MX rekordhoz a második számú IP címet rendeljük, amelyet a belső levelező szerverek használnak. Így legrosszabb esetben a vállalat levelezési rendszerének kommunikációja kerül felfüggesztésre, a webes forgalom (számlázás, publikált szolgáltatások) elérhető marad. Mindezek alapja a kockázatelemzés, amellyel a tervezéskor el lehet dönteni, hogy a költségek vagy a rendelkezésre állás fontosabb a vállalat számára.

# 3. ZOMBIHÁLÓZAT

A zombihálózat, vagy botnet, megfertőzött informatikai eszközök (számítógép, mobil eszköz, router, tűzfal) hálózatát jelenti. Minden olyan informatikai eszköz fertőzötté válhat, amely irányítás alá szervezhető. [1] Ez a megfertőzött hálózat lokáció szempontjából belső hálózat szintű és világszerte kiterjedt is lehet. Ez adja a zombihálózat lényegét: a megfertőzött eszközök elosztott erőforrásokként használhatók különböző rosszindulatú tevékenységek végzésére valamilyen irányítóközpont által. [2]

Rosszindulatú tevékenységek:

- kéretlen levelek tömeges küldése:
  Spam áradat küldése adatszerzés, csalás, dezinformálás végrehajtása céljából.
- adathalász és kémprogramok terjesztése:
  Felhasználói adatok (például bankkártya adatok) megszerzése, felhasználói viselkedés megismerésére (például meglátogatott honlapok), számítógépekből információk kinyerése, ezzel segítve támadási kampányokat, vagy Interneten keresztüli felhasználói megszemélyesítéseket.
- szolgáltatások leállítására szolgáló támadások:
  A zombigépek valódi kérésekkel (például HTTP) árasztják el a megcélzott szolgáltatást, így próbálva belassítani, megbénítani annak működését (DoS, DDoS támadás).

és egyéb károkozásra alkalmas műveletek végrehajtása.
A 3.1. táblázatból összegezve megismerhető a zombihálózat működése és annak tulajdonságai.

| Tulajdonság | Leírás |
|---|---|
| támadási mód | DoS, DDoS, spam, backdoor, stb. |
| vezérlő | központosított, láncolt, hibrid, véletlenszerű |
| kommunikáció | IP, IRC, HTTP, stb. |

3.1. táblázat. Zombi tevékenység jellemzése (forrás: a szerző)

Az első zombihálózatok, botnetek az IRC hálózatok terjedésével alakultak ki. Az IRC, az Internet Relay Chat, egy szerver-kliens alapú csevegő protokoll, kezdetben ez a protokoll volt az alapja a zombi kód működésének. [3]

Feltérképezett sérülékenységet kihasználva a felhasználók - afféle viccként - egymás számítógépei felet tudták átvenni az irányítást. A profik az utasítások átírásával saját fejlesztésű szkripteket tudtak az IRC hálózatba küldeni, megalapozva ezzel a rosszindulatú támadó és az automatizált tevékenységű kártékony robot tervezett és célzott munkáját, a zombihálózatot.

Zombihálózat kialakulhat:

- Kampány segítségével a zombivá alakító kártékony kód terjesztése távoli, más rendszerekhez, akár más zombi gépek segítségével.
- Hiszékenységeket, hibákat és sérülékenységeket (emberi, technikai) kihasználva a zombikód lefut az áldozat számítógépen, fertőzötté válik.

A zombihálózat létrehozói:

- Mély IT tudással rendelkező szakemberek, akik számára a támadás megvalósítása a cél. Felhasználják őket:
- Idegen hatalmak, katonai-, civil- és bűnszervezetek.

A fertőződés, a "zombisodás" ismertető jelei:

- Jól megtervezett kártevő esetén, amely nem okoz emberi szemmel észrevehető viselkedést, telepített intelligencia nélkül, szinte sehogy.
- Telepített intelligenciával (host alapú behatolás felismerő rendszer - host IDS, network alapú IDS, logelemző, végpontvédelem) kapcsolatok elemzése révén központi tudásbázissal rendelkező technológiákkal nagy hatékonysággal tetten érhető a zombi tevékenység.

A zombigép működése:

- A települt robot-bot-malware ezután kódolásától függően végrehajt vagy várakozik a vezérlésre, melyet egy úgynevezett Command&Control (C&C) szervertől, irányítótól kap a backdoor-ként megnyitott csatornán keresztül. A zombivá vált számítógép akár 25000 darab kéretlen levelet is kiküldhet úgy, hogy a számítógép kezelője nem is tud róla. [4]
- A C&C csatornán a vezérlőtől érkező utasítás értelmében a zombi végrehajt, majd

ismét utasításra vár.

A szakdolgozatban bemutatásra kerülő saját fejlesztésű, lokálisan futó alkalmazás ezt a nem megengedett kifelé irányuló csatorna nyitást ellenőrzi kitalált környezetben.

A zombigép életciklusa:

- Addig tart, amíg a kártékony kód rejtve tud maradni és törlésre nem kerül. Ekkor megszűnik a botnet tagság.
- Addig tart, amíg a C&C csatornán keresztül a vezérlő elérhető. Kiiktatott vezérlő szerver esetén a zombi a programkódjától függően viselkedik (várakozik, megfigyel, adatot gyűjt), viszont amint a C&C szerver újra elérhetővé válik és utasítást ad, "a zombi feléled".

A technika fejlődésével a zombi kód is intelligensebbé válik, képes átalakulni - IP-t és portot váltani, képes lehet felismerni a védekezésre használt csapdákat és technológiákat (például honeypot).

A zombivezérlőkről pár szóban.
Olyan személyek, csoportok üzemeltetésében lévő szerverek és alkalmazások, melyek többféle hálózati topológiájú (központosított, láncolt, egymással kommunikáló, hibrid, véletlenszerű) környezetben szerver-kliens alapú kommunikációt használva (például az IRC protokoll) működnek, kártékony tevékenységek végrehajtását vezérelve. Ezek a rendszerek az informatikai világ "sötét oldalán" bérbe vehetőek, megvásárolhatóak. [5]

# 4. VÉDEKEZÉSI LEHETŐSÉGEK

A szakdolgozat bevezető részében a zombihálózat, mint kártékony informatikai fenyegetés elleni védekezés lehetőségei kerültek kérdéskörbe.
A szakdolgozat ezen szekciója talán a legfontosabb része a teljes egésznek, mert rávilágít az informatikai biztonság fontosságára és arra az óriási, folyamatosan változó területre, amelyre a biztonsági szakembereknek figyelniük kell.
Az informatika világában - üzemeltetés, tervezés, biztonság - eltöltött két évtizednyi tapasztalat birtokában keresem a legjobb választ a védekezési lehetőségek megadására.

Mi a biztonság? Mit jelent az informatikai rendszer biztonsága? Mi ellen kell védeni az informatikai rendszert? Hogyan lehet védeni az informatikai rendszert? Hogyan kerüljük el az áldozattá válást? Mi a teendő egy incidens esetén és az után?
Csak néhány kérdés és gondolat, amelyeket több szempontból, sokféle értelmezésben kell elemezni ahhoz, hogy megválaszolhatóak legyenek.

A felsorolt kérdéseken túl a védekezési lehetőségek témáját értelmezhetjük továbbá:

- megelőzés - eseménykezelés,
- felhasználói - üzemeltetői,
- otthoni - vállalati,
- adminisztratív - technikai

szempontok alapján is, de sorban említhetőek még az olyan lehetőségek, mint működési elvek alapján, vagy éppen hozzáférhetőség alapján besorolható védekezési formák. Ez utóbbira jó példa a zárt hálózat.

## 4.1. Zárt hálózat

Talán a legjobb védekezési mód - amely miatt a dolgozatban külön alfejezetet is kapott - az elkülönített, külvilágtól elszigetelt informatikai rendszer, a zárt hálózat, amelybe vagy kézi adatbevitelen keresztül, vagy adatdióda használatával történik az információ beemelése, azaz befelé kerülhet adat, kifelé nem.
Adatszivárgást csak szándékos emberi tevékenység okozhat.
Az adatdióda adatok átemelésére szolgáló eszköz, alkalmazása nagy mennyiségű adat automatizált zárt hálózatba pumpálása esetén szükséges. Ez esetben a zárthoz képest gyengébben védett hálózat és a zárt hálózat között OSI-modell szerint Layer 1, fizikai össze-

köttetés szinten kapcsolat áll fenn. Ekkor az adatdióda működési elve adja a biztonságot. (Például: három darab média konverter, az őket meghatározott módon összekötő optikai patch kábelek, valamint UDP (user datagram protokol) csomagok küldésére fejlesztett szerver-kliens alkalmazás páros használatával megoldható az adatok digitális formában zárt hálózatba juttatása.[1])

A zárt hálózatokat egy gondolat erejéig megszakítva "fehér kalap"-ról "fekete kalap"-ra váltva: alkalmazhatjuk-e a 'VLAN hopping' elnevezésű technikát adatdióda felhasználásával történő hekkelésre? Rombolásra fejlesztett kártevőt juttatva be ezzel a zárt hálózatba károkozási céllal.
VLAN hopping technika használatával a támadó károkozó kódot, adatot próbál küldeni egy VLAN (virtual local area network) hálózatból a másik VLAN hálózatba. Kicsavart módon tekintve a támadási technikát, az rá is húzható az adatdióda működési mechanizmusára.

A zárt hálózat jó védekezési lehetőség, kialakítása viszont költséges, mérlegelni szükséges, hogy megéri-e bevezetni. Amennyiben jogszabályi kötelezettség áll fenn, kötelező a használata. Ha egy kockázatértékelés maximális veszélyszintet állapít meg (például rejtjelzett hálózat végpontjai), akkor csak a zárt hálózat marad, mint a biztonság egyik alappillére.

## 4.2. Megelőzési és kezelési lehetőségek

A következő alfejezetekben továbbfejtésre kerülnek a védekezési lehetőségek, amelyek sok esetben átfedésbe kerülnek egymással. Ez is mutatja azt a tényt, hogy az informatikai biztonság egy sokténye zős, összekeveredő "massza".
Leggyengébb láncszem minden esetben van, de nem mindegy, hogy a leggyengébb elem védtelen és támadható, vagy szakszerű tervezés következtében védett eszközök mögött található, amely elfoglalt pozíció el is lehetetlenítheti a támadás végrehajtását. Ilyenre lehet példa a következő, egyszerűsített hálózati szegmens: LAN - tűzfal - tűzfal - szerver. Ebben az esetben az első tűzfal megtörése után még mindig ott a második tűzfal, amely blokkolni tudja a támadást, szerencsés esetben pedig a tűzfalak naplóiból elemzés során rá lehet találni a támadás nyomaira, így megteremtve a lehetőséget a védekezésre.

[1] A szerző saját fejlesztése alapján.

Tervezés, mint megelőzési forma. Jól szervezett és átgondolt tervezéssel megelőzhetjük azokat támadásokat, amelyekre készülünk, amelyekről azt gondoljuk, hogy alkalmazni fogják a technológiánk ellen.
Minden olyan tervezett lépés, átgondolt eseménylánc, megvalósított kivitelezés, amelyek előre felkészüléssel biztonságot növelő célokat szolgálnak, támadás megelőzési eljárásként alkalmazhatóak.

Incidens előzhető meg:

- tudatosítással, képzéssel:
  Általános, illetve szakrendszerhez kötődő eseti és rendszeres oktatás, figyelem felhívás, tájékoztatás. Célja a biztonságtudatos módon viselkedő humán erőforrás megteremtése.
  Ha a felhasználó a tudatosító képzések hatására felismer egy helyzetet - nem küldi tovább a gyanús e-mailt vállalaton belül - és szól az informatikai, biztonsági területnek, akkor már elérte a célját a biztonságtudatos munkavégzésre "nevelés".

- működési szabályok megalkotásával:
  Biztonsági szabályzatok, eljárásrendek megalkotása: informatikai biztonsági szabályzat, cselekvési terv, eszköz és technológiák használatának szabályozása, valamint további kényszerítő intézkedések és azok betartatása.

- külső szervezetek figyelmeztetéseivel:
  Szakszolgálatok, informatikai biztonságra szakosodott intézmények, nemzetközi szervezetek figyelmeztetéseinek követésével, érintettség esetén a sérülékenység javítás implementálásával.

- informatikai rendszer karbantartásával:
  A felügyelt informatikai rendszer folyamatos ellenőrzésével, "up2date" állapot használatával, avulások követésével, egyszóval folyamatos felügyelettel.

- szankiconálással:
  Az informatikai rendszerhez kapcsolódó felhasználók figyelmeztetése a munkafolyamat kezdetekor, illegális tevékenység végzésének megtiltásával. Szándékos károkozás megállapítása esetén, helyzet függvényében történő válaszlépések alkalma-

zása (fegyelmi, feljelentés, más minőségű reagálás.)

Incidens kezelhető:

- Üzletmenet folytonossági terv információbiztonsági eseményre vonatkozó lépései alapján, például Disaster Recovery Plan (DRP) - informatikai katasztrófaterv, Business Continuity Planning (BCP) - üzletmenet folytonossági tervezés. (DRP minden esetben elindul, BCP csak akkor, ha a kiesés ideje meghaladja a meghatározott maximális kiesés idejét.)
- Információbiztonsági esemény kivizsgálásával: naplók és adatok vizsgálata, károk becslése, helyreállítás, szükség szerint az érintett terület újratervezése.
- Jelentésköteles esemény, vagy - súlyosabb esetben - bűncselekmény azonosítása esetén továbbítás az illetékes hatóság felé.

A fentebb leírtak alapján látható, hogy az otthoni tevékenységek nem kerültek említésre a megelőzés témakörében. Otthon mindenki a maga ura:
Ki használhatja a WiFi-t?
Mindenki, aki ismeri a jelszóval védett vezeték nélküli hálózat jelszavát. Abban az esetben, ha jelszó használat sincs, vagy könnyű a jelszó, az nyitott kapu a rosszindulatú tevékenységre készülő ember számára.

Kik használhatják a számítógépet?
Amennyiben nincs legalább jelszóval védett belépés az otthoni számítógépre, a válasz tulajdonképpen: mindenki. Rosszindulatú tevékenység végrehajtása szempontjából ideálisak ezek az eszközök.

Default beállításokkal működik a szolgáltatótól kapott router?
Amennyiben igen, internet irányból az érintett router ismert hiányosságai kihasználva károkozás végrehajtására felhasználható fel az otthoni előfizetés.

Vírusvédelem telepített és naprakész? [6]
A teljes funkcionalitással működő, megvásárolt végpontvédelmek a sérülékenységek bizonyos fajtáinak és többféle kártékony megoldásnak a kivédésére alkalmasak.

Csak néhány gondolat, amelyekből azonosítható, hogy az áldozattá válás megelőzésének az otthonokban is fontos szerepe lenne.

Szervezeti körülmények között pedig nagy felelősség hárul az informatikáért, biztonsági szakterületért felelő szakemberekre, valamint a szervezet vezetőjére, akik együttes munkája szükséges a támadásokra felkészült informatikai háttér biztosításához.

## 4.3. Humán lehetőségek

Miért válik egy informatikai rendszer sérülékennyé? Fontos kérdés!

- Hiányosan tervezett, kivitelezett és karbantartott rendszer. - Emberi tényező miatt.
- Hanyag viselkedés következtében. - Emberi tényező miatt.
- Nulladik napi támadás következtében. - Végső soron ez is emberi tényező, de ez esetben inkább a támadó kerül a fókuszba.

A felsoroltak alapján látható, hogy az emberi tényező mindig jelen van az áldozattá válás folyamatában. Valahol mindig ott az ember, akár úgy is, mint a leggyengébb láncszem.
A támadó pedig rendelkezik azzal a tudással, hogy a leggyengébb láncszemet különböző technikák használatával megtalálja: social engineering, adathalászat, "elhagyott" fertőzött eszköz használata.

Mit tehet az ember, hogy ne váljon áldozattá, ne rajta keresztül használja ki a támadó a további sérülékenységet?
Válasz: Biztonságtudatos viselkedés elsajátítása. - Tréningek, oktatások tartalmának megjegyzése, sérülékenység kihasználására irányuló helyzet felismerése a munkahelyi környezetben és az élet más területein:

- Adathalászat (online, e-mail vagy telefonon keresztül) elkerülése.
- Felhasználói és személyes adatok védelme.
- Megtévesztő, nem megszokott tevékenységek és események felismerése.
- Nem megbízható eszközök, hálózatok használatának elkerülése.
- Gyanús folyamatok megszakítása.

Mit tehet az üzemeltető és a biztonsági szakember, hogy a felügyelt rendszer ne váljon áldozattá?
Válasz:

- Szabályozott működési környezet létrehozása és a szabályok betartatása.
- Felügyelt rendszer ismerete, annak folyamatos ellenőrzése, szükség esetén újratervezése.

– Képzés, fejlődés, oktatás.
– Trendek, kampányok követése.
– Technikai fejlődés követése, implementálása.

A fentebb leírtak alapján - a megelőzéshez hasonlóan - az emberi tényező hatása is fontos szerepet játszik a támadások elleni védekezésben. Fontos a prevenció, a szabályozott üzletmenet. A megfelelően biztonságos működéshez szabályozott és előre ismerhető építőelemek szükségesek. Ha ezek rendelkezésre állnak, akkor a humán erőforrás - ha akar - tud alkalmazkodni környezetéhez és viselkedésével nem lesz hátráltatója az informatikai rendszer működésének, azaz biztonságtudatosan fog viselkedni.

## 4.4. Technikai lehetőségek

Technikai védekezés - az emberi tényező hibáinak javítására, utólagos felderítésre. A rendszerben rendelkezésre álló összes eszköz tudásának felhasználásával téve biztonságosabbá annak működését, lényegében tervezett hardening végrehajtása.

Csökkenthetjük a biztonsági események számát:

– fizikai biztonság létrehozásával:
  Beléptető, megfigyelő és riasztó rendszerek beüzemelése a beléptetés, felügyelet és helyiségek védelme érdekében.

– hálózat szegmentálással:
  Az informatikai rendszer hálózatának szétválasztása, működési sajátosságok alapján szegmensekre bontása, például a demilitarized zone (DMZ) - háttérhálózat, VLAN - virtuális LAN.

– csomópontok ellenőrzésével:
  Azonosítani kell azokat a hálózati pontokat, amelyeknél meg lehet fogni a támadást, ezekre több erőforrás allokálás (főleg anyagi) szükséges. A szegmentált részek kapcsolódási pontjaiban üzemelő eszközök - tűzfal, router, switch - beállítása: hozzáférés kontroll, naplózás, biztonsági mentés, sérülékenység ellenőrzés.

– biztonsági mentés használatával:
  Bekövetkezett esemény (szándékos támadás, véletlen törlés, konfiguráció hiba) után offline tárolóból a rendszerműködés visszaállíthatóságának biztosítása.

– monitorozás, naplózás beállításával:
  Rendszerelemek által rögzített események ellenőrzése céljából, valós időben illetve utólagos ellenőrzés keretében végrehajtva. Például az Intrusion Decection System (IDS) - illetéktelen hálózati behatolást jelző rendszer, az Intrusion Prevention System (IPS) - behatolás megelőző rendszer, logelemző- és adatszivárgás ellenőrző rendszerek.

– végpontvédelem használatával:
  Központosított vírusvédelem, kártékony kód elleni védelem és egyéb alkalmazások telepítése.

– hozzáférés menedzsment kialakításával:
  Központi szabályozás segítségével befolyásolni a kártékony események kialakulásának lehetőségét, például cserélhető eszköz használatát tiltó policy, mobil eszköz működését szabályozó menedzsment.

## 4.5. Helyi lehetőségek

További kategória a helyszín alapú védekezési módszerek elve. Keveredik benne a humánum és a technológia, esetei:

– Munkahelyi tiszta asztal politika (biztonságtudatosság).
– Otthoni munkavégzés, távmunka (biztonságtudatosság).

Közös jellemző:

– Idegen, illetéktelen harmadik fél számára nem szabad a munkavégzésre kapott eszközökhöz hozzáférést engedni. Szerviz tevékenység is csak felügyelet mellett végezhető.
– Ismeretlen adathordozó használata, tartalmának megtekintése nem engedélyezett.
– Nem ellenőrizhető alkalmazások, ismeretlen vezeték nélküli hálózatok használata nem engedélyezett.
– Csak a feladatvégzéshez szükséges alkalmazások használata engedélyezett, hozzá nem kötődő tevékenység végzése nem engedélyezett.

A fejezet fontossága abban rejlik, hogy rámutat a biztonságtudatos viselkedés fontosságára. Nem elég a technológiai védelem, meg kell akadályozni a használatban lévő eszköz

idegen kezekbe kerülését, így gátolva egy incidens kialakulásának a lehetőségét.

## 4.6. Legjobb módszer

A legoptimálisabb védekezési módot nehéz definiálni, mert 100 százalékos védelem nincs. Befolyásoló tényezők lehetnek:

- Pénzügyi - költségvetési helyzet.
- Technikai felszereltség, ami függ az anyagi kondícióktól.
- Helyszíni adottságok.
- Szaktudás!

A legjobb módszer a védekezésre az előző fejezetek összességének alkalmazása: felmérés, kockázatelemzés, tervezés, kialakítás, képzés, karbantartás, követés, felülvizsgálat - majd az idő előrehaladtával, vagy incidens esetén az előbbi lépések folyamatos megismétlése. A zombivá válás talán legbiztosabb megelőző módszere a biztonságtudatos felhasználói viselkedés.
A módszerek összegzése következtében a legoptimálisabb védekezési megoldásnak egy olyan alkalmazás kifejlesztését látom, amely a zombitevékenység célzott megszüntetésére képes.

# 5. TESZTKÖRNYEZET

A bevezetőben megfogalmazott védekezési témakör vizsgálata szerint egyedileg fejlesztett alkalmazással kezelhető a zombitevékenység végrehajtása.
A fejlesztendő alkalmazás kitalált környezetben, kitalált tesztkörnyezetben kerül felhasználásra. Ennek oka, hogy a valósággal ellentétben - ahol a környezeti tényezők specifikálják a programot - a szakdolgozat készítés adta szabadságot felhasználva, a környezet adaptálható a programhoz: a védekezési mechanizmus megvalósításához és működésének ellenőrzéséhez csak egy autonóm rendszerre van szükség, amely lehetőséget biztosít a megtervezett védekezési mechanizmus alkalmazhatóságának bemutatására is.

## 5.1. Szervezeti környezet

Fizikai megközelítést bemutató módszert használva:

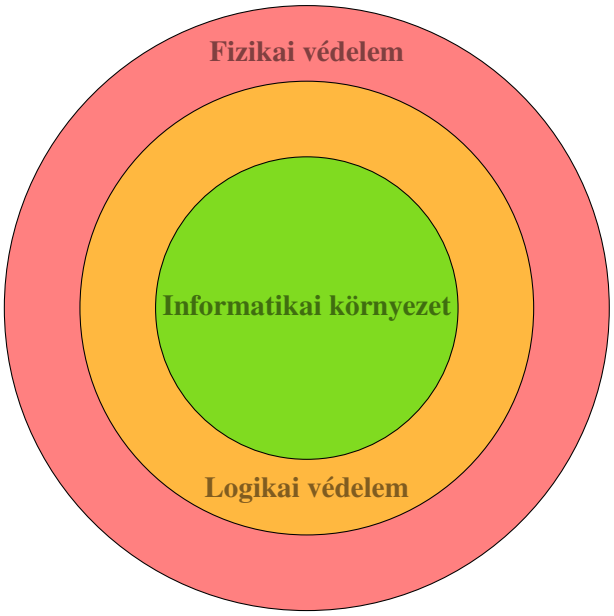
- 0. zóna - külső határ: az elhelyzést biztosító épület téglafallal, zárható bejárattal és vagyonvédelmet biztosító berendezésekkel rendelkezik.
- 1. zóna - belső héj: adminisztratív biztonsági tevékenységet ellátó, portaszolgálat jellegű recepció napközben.
- 2. zóna - munkaterület: a humán és technikai erőforrások tartózkodására és elhelyezésére rendelkezésre álló kiszolgáló és szociális helyiségek.

Logikai védelem szerinti bemutatás:

- Működést szabályozó rendelkezések:
  - Informatikai Biztonsági Szabályzat
  - Szervezeti Működési Szabályzat
  - Üzemeltetési Szabályzat: például jelszó menedzsment, változáskezelés.
  - Katasztrófa-elhárítási Terv
- Kockázatkezelés: a bizalmasssság - sértetlenség - rendelkezésre állás hatásainak elemzése a folyamatokban.
- Tudatosítás: például a bekövetkező események kezelési módjainak, munkaeszközök használatának oktatása.

  Az 5.1. ábra a fejlesztendő alkalmazás védelmi környezetét elméletben ábrázolja, elindulva a fizikai környezetből, átlépve a logikai védelmi megoldásokhoz és az informatikai környezethez, ahol a rosszindulatú fertőzés megvalósulhat.

  A valóságban egy fizikai fenyegetés már a fizikai védelem területén nagy valószínű-




5.1. ábra. Védelmi környezet (forrás: a szerző)

séggel megállítható, a támadó nem juthat el az informatikai környezetig. Kódsoros fenyegetés logikai védelmi intézkedésekkel megelőzhető, meggátolva a károkozó kód bejutását az informatikai környezetbe, amelyben további megelőző és védekező technológiák állhatnak még rendelkezésre.

## 5.2. Informatikai környezet

A kitalált informatikai infrastruktúra technikai felépítése a következő elemekből áll:

- Határvédelmi rendszer: redundás működési elvű tűzfal, mely a külső lábán az ISP-vel, az internet szolgáltatóval tart kapcsolatot, a szabályok alapján átengedhető forgalmat be- és kiforgatva látja el a belső hálózat és a DMZ forgalmának szabályozását.
- Végpontvédelmi rendszer: az infrastruktúrában üzemelő szerverek, munkaállomások vírus- és hálózati védelmét ellátó szerver és szoftverközpont, valamint a hozzá kapcsolódó végponti telepítésű szoftverek.
- Demilitarizált zóna: a csökkentett védelmű hálózati szegmensben üzemelnek a Internet felé kapcsolatot tartó rendszerek: levelező rendszer, webes publikációt végző szerverek és a hozzájuk kapcsolódó adatbázis szerverek. A belső hálózatból történő kommunikáció (adat, menedzsment) csak tűzfalon keresztül történik.
- Strukturált hálózat: fizikailag (végponti kábelek, horizontális rendezők, gerinc ká-

belek) és logikailag (VLAN-ok, alhálózati szegmensek) is elválasztott hálózati kialakítás áll rendelkezésre.
– A hálózatban IDS és IPS, DLP - adatvesztés elleni védelem és SIEM - biztonsági információ és eseménykezelő rendszerek nem állnak rendelkezésre.

Az informatikai infrastruktúra logikai felépítése a következő elemekből áll:

– Microsoft Windows alapú tartomány: tartományvezérlők a belső halózatban, illetve írásvédett tartományvezérlő a DMZ-ben.
– Microsoft Active Directory: címtárszolgáltatás a tartomány tagjainak, komponenseinek központosított menedzselése (authentukáció, authorizáció, SSO - egyszeri bejelentkezési módszer, LDAP - központosított felügyeleti rendszer, policy-k, felhasználó menedzsment) rendelkezésre áll.
– Microsoft Windows UAC -felhasználói hozzáférési kontroll kikapcsolva.

Az elméletben felvázolt informatikai környezet biztosítja a strukturált belső hálózatot, belső szerverszintű szolgáltatásokat, automatizált frissítési rendszert, a felhasználói szintű korlátozásokat, a felügyeleti rendszert és az ezeket működtető technikai személyzetet.

## 5.3. Tesztelési követelmények

Programfejlesztés során szükséges egy rendelkezésre álló tesztkörnyezet is, amelyben az éles környezet szimulálása lehetséges.
Minden fejlesztés közbeni kód változás működését a tesztkörnyezetben szükséges ellenőrizni. Éles rendszerbe a már minden szempontból bevizsgált program és működési környezete kerülhet.
A szakdolgozatban fejlesztésre kerülő alkalmazás működési elvéből adódóan nincs szükség komoly hátterű tesztrendszerre, virtuális és fizikai gépekre. Jellegéből adódóan lokális működést végez az alkalmazás, ezért akár a fejlesztést kiszolgáló számítógép maga is alkalmas a tesztelésre.

A szakdolgozati alkalmazás teszteléséhez szükséges feltételek:

– Működő hálózati kapcsolat - TCP stack eléréséhez.
– Működő operációs rendszer - Windows folyamatok használatához.
– Jogosultsági szint - rendszergazda szintű tevékenységhez.

# 6. ALKALMAZÁS

A bevezetés kérdéskörének utolsó gondolata kerül megválaszolásra ebben a fejezetben. Védekezési oldalról helytálló egy olyan alkalmazás fejlesztése, amely saját logikával védekezik a zombitevékenység ellen. Az elképzelt informatikai környezet és egyszerűen felépíthető tesztkörnyezet rendelkezésre áll, következhet a megfelelően tervezett alkalmazás létrehozása.

## 6.1. Programtervezés

Az alkalmazás létrehozása előre definiált kontroll módszer alkalmazásával történt, amely segíteni tudja a fejlesztés hatékonyságát:

- Kérdés: Mi a feladat? Milyen alkalmazást kell készíteni?
  Válasz: Olyan program készítése, amely el tudja dönteni a kifelé induló TCP vagy UDP kapcsolódás jogosságát.
  Indoklás: A zombikód IP hálózaton keresztül próbál kommunikálni a vezérlő szerverével, amelyet IP alapokon TCP vagy UDP kommunikációval érhet el.

- Kérdés: Mi a működési elv, milyen modell alapján működjön az alkalmazás?
  Válasz: Az ISO/OSI modell hálózati és szállítási rétegét elérő, csomagok vizsgálatára alkalmas eljárások és logikai függvények alkalmazása.
  Indoklás: Az ellenőrizendő adatcsomagok a szállítási rétegben értelmezendőek, ahhoz a réteghez kell utat találnia az alkalmazásnak.

- Kérdés: Milyen algoritmusokra van szükség?
  Válasz: IP stack-ből adatok ciklikus olvasása, feltételes elágazások készítése, utasítás végrehajtó folyamatok.
  Indoklás: A tervezésre kerülő program elemeinek különböző részfeladatok elvégzését kell végrehajtaniuk, a bemeneti és kimeneti információknak egymásra kell épülniük a működés érdekében.

- Kérdés: Milyen szoftver és hardver környezet áll rendelkezésre?
  Válasz: Szabványos x86-64 utasításkészlet-architektúra, Microsoft Windows 10 operációs rendszer.
  Indoklás: A szakdolgozatban fejlesztésre kerülő alkalmazás tesztelési és futtatási

környezete meghatározza a fejlesztés kereteit.

– Kérdés: Milyen kódolással készüljön az alkalmazás?
  Válasz: A szabványos PC alapú működési környezet nem kívánja meg speciális eszközökre (Raspberry Pi, Arduino) fejlesztett nyelvek és fejlesztő környezet használatát, viszont a többrétű felhasználhatóság (Linux, macOS, Windows) érdekében mindenképpen egy rugalmas, többplatformos, magas szintű programozási nyelv használata szükséges.
  Indoklás: Az általam ismert és használt programozási nyelvek közül a feladat elkészítésére a Python és a C++ nyelvek megfelelőek.

Fejlesztések esetében igaz, hogy a programozó tudása dönti el a választott programnyelv típusát, így a szakdolgozati alkalmazás az egyszerűség és könnyű fejleszthetőség miatt Python nyelven íródott, annak minden előnyével és hátrányával együtt.
Előny: Python környezet folyamatos fejlődése, nagyszámú fejlesztői közösség, szabad forráskódú összetevők.
Hátrány: a futtatható fájlba fordított alkalmazás a Pythonban használt csomagok importálása miatt a néhány kilobájt méretű forrás állományból több megabájt méretűre is hízhat.

## 6.2. Fejlesztés

A fenti kontroll kérdések tisztázása után következhet a program fejlesztésének bemutatása:

– a programírás:
  A kiválasztott nyelven történő fejlesztéshez a Visual Studio Code nyílt forráskódú kódszerkesztő és a Python-hoz szükséges kiegészítők és csomagok telepítése szükséges.
  A 7.2. ábra alapján, amely a működési elvet mutatja, részenként fel kell építeni az alkalmazás kódjait, például:
    • hálózati kommunikáció figyelés, kimenő csomag keresés (Python előfeltételek: psutil, socket modulok)

      *for IP_folyamat in psutil.process_iter(['pid', 'name']):*
      *Folyamat_neve[IP_folyamat.info['pid']] = IP_folyamat.info['name']*

*for Kapcsolodas in psutil.net_connections(kind='tcp4'):*
*IPcim_helyi = (Kapcsolodas.IPcim_helyi)*

*if IPcim_tavoli != "" and IPcim_tavoli.find ("127.0.0.1"):*

- kiválasztott program, process bezárása

*program_bezar = psutil.Process(Kapcsolodas.pid)*
*program_bezar.terminate()*

Néhány szóban a Python-ról és a modulokról. A python interpretált nyelv, azaz a használatával fejlesztett programok önálló működésre, futásra alkalmatlanok. Fordításukhoz, futtatásukhoz szükséges egy interpreter, értelmező, amely a leírt utasítás sorokat a gazda rendszer utasításkészletének megfelelő gépi kódú kódsorokká alakítja át és rögtön futtatja is azokat.
E programozási nyelv érdekessége, hogy benne minden elem objektum. Csak dinamikus típusokat tartalmaz, tehát típusok fordítási időben keletkeznek, nem kell előre definiálni azokat.
Az előbbiekben bemutatott kódsorok elkészítését úgynevezett Python modulok használata teszi lehetővé. A szakdolgozati program elvárt működéséhet a psutil és a socket modulok használatára volt szükség. Amennyiben ablakos felületű GUI szükséges, akkor rendelkezésre állnak további modulok, mint a wxPython vagy a Tkinter.
A modulok olyan könyvtárak, melyek többplatformos környezethez illeszkedő, előre elkészített függvényeket, kódsorokat tartalmaznak, így például egy process azonosító kinyeréséhez nem szükséges mély kernel vagy driver ismeret, elég a közösség által már rendelkezésre bocsájtott, szabadon használható függvények felhasználása. Ugyanez a helyzet egy GUI létrehozása esetében is: modul behívás, előre definiált eljárások használata. Így néhány programsor lehetővé teszi, hogy periféria használatával, lényegében kattintásra átméreteződjön a zombitevékenység ellen fejleszett alkalmazás ablaka.
A modulok félesége és számossága nagyszámú, köszönhetően a folyamatosan fejlődő Python világnak.
A modulokban található tudás felhasználásához a világhálón találhatunk mindig naprakész információkat [7]

– fejlesztés közbeni tesztelés, hibakeresés, javítás és kommentek rögzítése:
  Az elkészült folyamatok (lekérdezések, ciklusok, stb.) eredményeinek futtatás közbeni, akár "breakpoint"-ok felhasználásával történő ellenőrzése, elősegítve ezzel a hibakeresés folyamatát, például egy végtelen ciklusba eső folyamat javítása miatt.
  A kommentek használata pár szóban szintén erősen javasolt, mert így bármikor emberi nyelven lekövethető, hogy a kódsorok halmaza mi célt szolgál.

– fejlesztés befejezése utáni tesztelés, hibakeresés, javítás és kommentek rögzítése:
  A fejlesztés utolsó fázisainak egyike, ekkor már rendelkezésre áll egy működő alkalmazás példány. Ekkor szintén célszerű teszteli a bemutatható állapotot, feljegyezni az esetleges hibákat, például: egy beviteli mező inputjai megfelelnek-e az input feltételeknek.
  Amennyiben rendelkezésre áll a felfedezett hibák javítására fordítható idő, mindenképpen törekedni kell azok javítására, elkerülve ezzel az éles használat közbeni gondokat.

– optimalizálhatóság keresése:
  A tesztelt és jól működő alkalmazás kódját célszerű újra átfésülni, találhatóak -e benne felesleges kódsorok - egy globális változó, amire nincs szükség -, esetleg rosszindulatú támadásnak kedvező fejlesztés alatt használt cleartext jelszavak és biztonsági adatok. Illetve fontos szempont, hogy lehet-e valahol a kódsorok között egyszerűsíteni a folyamatokat.

– dokumentáció létrehozása:
  Elsősorban a felhasználó számára fontos a működési leírás, különösen ha felhasználói interakcióra is szükség van használat közben. Lehet beépített 'Help' menü, vagy szövegszerű nyomtatott forma.
  Továbbá szükséges a fejlesztői dokumentáció is, hiszen ha a későbbiekben további fejlesztés szükséges, vagy auditálásra kerül sor, a forráskód mellett a hozzá tartozó magyarázó leírás is fontos eleme a tovább haladásnak.

– szükség esetén szoftverkövetés:
  A megváltozott felhasználói igények lekövetése, szükséges hozzá az újbóli programtervezés, amelynek alapul szolgálnak a korábban készített dokumentációk.

Programtervezés közben nem árt figyelni, milyen problémákat okozhat az alkalmazás az őt futtató környezetben. Fejlesztés és tesztelés közben ki is kell tekinteni a fejlesztő környezetből, történik-e nem kívánt működésbeli hatás például a kiszolgáló operációs rendszerben. Lehet, hogy a program az elvárások szerint működik, "teszi a dolgát", de ezzel lehetséges, hogy instabillá teszi az alatta futó operációs rendszert vagy más alkalmazást.

Következzen néhány élő példa, milyen nem várt események adódtak az alkalmazás építése közben:

- Végtelen ciklus.
  A rosszul definiált elágazási feltétel végtelen ciklusba kényszerítette az alkalmazást. Az operációs rendszer próbálta a rendelkezésre álló processzor magok számítási szálait, kapacitását átadni, de így is másodpercek alatt DOS támadás áldozatává vált a fejlesztő PC. Ezt a hibát a kód módosításával lehet kiküszöbölni, mert a Python interpreter korlátok közé szorítható, - csak 1 CPU magot használjon - de ettől a kód még hibás marad.

- Operációs rendszer saját folyamatainak bezárása.
  Az épülő alkalmazás egyes verzióiban a saját logikája, rendszergazdai jogosultság birtokában, minden kifelé kapcsolódást kezdeményező process-t, folyamatot leállított, ezzel kikapcsolva a Windows operációs rendszer saját működéshez szükséges folyamatait, részeges vagy teljes fagyást idézve elő (ez a process típusának és annak függőségeinek tekintetében változott).

- Egy alkalmazás több process-t nyit (például egy webböngésző).
  Webböngésző használatakor, több különböző böngésző oldal és honlap megnyitásakor az épülő alkalmazás az első távoli IP elfogásakor bezárta az azt indító, nyitó process-t, azaz a webböngészőt. A "kill process" után az alkalmazásom tervezetten, programkódja szerint tovább haladt, és a már megnyitott, következő távoli honlap IP címébe belekapaszkodott. Viszont ez esetben már nem volt a felfedezett IP-hez köthető process, alkalmazás, amit bezárhat, ezért hibára futott és leállt.
- Rosszul definiált feltételrendszer.
  Rossz logikai utasítások használatával nem, vagy hibásan keletkezett eredmény érték.

– Fejlesztő környezet frissítése.
  A fejlesztő környezetben bekövetkező változás nem várt problémát generált: Python beépülő kiterjesztés verzióváltása után a fejlesztett kód nem működött.

A fejlesztési folyamat végére az "A" függelékben található kódsorokból álló alkalmazás elkészült, képes az elvárt működésre, azaz a nem megengedett kifelé irányuló kapcsolódások leállítására. A továbbiakban lássuk a működés leírását és a védekező működés megkerülési lehetőségét is.

# 7. MŰKÖDÉS ÉS MEGKERÜLÉS

Ebben a fejezetben megismerhető, hogyan védekezik a fejlesztett alkalmazás a zombi tevékenység ellen, illetve milyen kártékony támadási módszer állhat rendelkezésre a felhasznált védekezés kiiktatására.

## 7.1. Az alkalmazás védekezési mechanizmusa

A tervezett védekezési módszer a következő alapvetésekre épül:

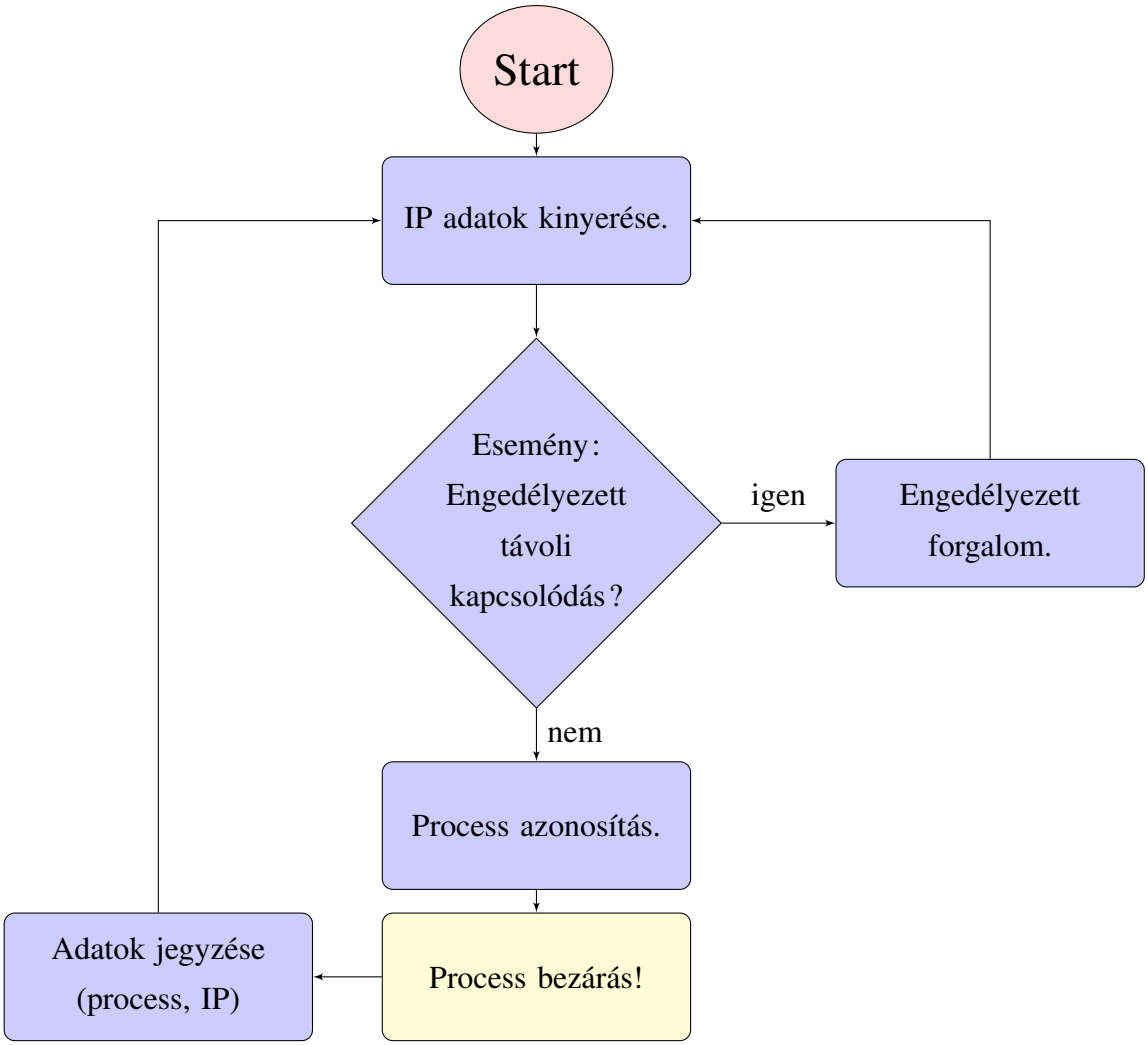
- A telepített Microsoft Windows operációs rendszerű munkaállomások frissítései WSUS használatával, belső hálózaton keresztül történnek meg. A munkavégzéshez szükséges programokat a belső hálózaton elérhető szerverek szolgálják ki.
- A vállalat által biztosított elektronikus levelezés megengedett, az Internet használat korlátozások mellett engedélyezett.
- Magáncélú böngészésre különálló számítógépeket biztosít a munkáltató.

### 7.1.1. A védekezési módszer leírása

Az alapvetésekből kiderül, hogy a munkaállomások nem köthetőek alapvetően zárt hálózatba, ahová csak adatdiódákon keresztül lehetséges az adatbevitel. A felhasználók számossága miatt a tűzfal kilyukasztása, szabályainak növelése sem megoldás, azaz, hogy minden felhasználó külön tűzfalszabályt kapjon, amelyek megengedik az Interneten barangolást. Hálózati oldalról is nehézkes lenne a kifelé irányuló kommunikáció konrtollja, mert nem minden munkaállomást lehet megfosztani a Internet felé biztosított átjárótól, így a szegmentálás sem megoldás.

Az alkalmazás minden előzőleg felsorolt problémát logikai úton old meg a 7.2. ábrán is látható módon:

- Az alkalmazás figyeli és leolvassa a hálózati kommunikációhoz köthető adatfolyamot.
- Az adatfolyamból kiszedett IP cím alapján logikai ellenőrzés történik:
  - Az IP a belső forgalomnak megfelelő?
  - Az IP a megengedett forgalomnak tekinthető?
- Megfelelőség esetén az alkalmazás tovább keres és hallgatózik.
- Nem engedélyezett alkalmazás által nyitott távoli IP esetén a kezdeményező process azonosításra kerül.




7.2. ábra. Védekezési elv folyamatábrája. (forrás: a szerző)

– A beazonosított process bezárásra és bezárásra kerül.

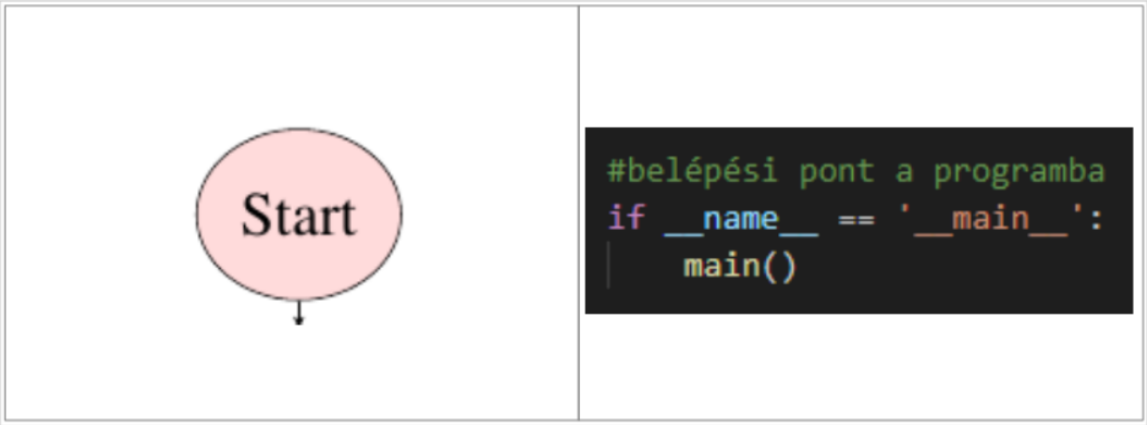
### 7.1.2. A védekezési módszer kifejtése

Hogyan feleltethető meg a működési mechanizmus a leprogramozott kóddal?
A működési elv teljes egésze négy nagyobb részre bontható:

1. indulás
2. hallgatózás
3. döntés
4. beavatkozás

Indulás.

A Python terminológia szerint az interpreter sorban egymás után, az első sortól haladva hajtja végre az utasításokat, ezért a program belépési, indulási pontja előre definiált kódsor alapján azonosítható, az abban megadott metódus használatával kezdődik a program futása, amelyet a 7.3. ábra mutat be.



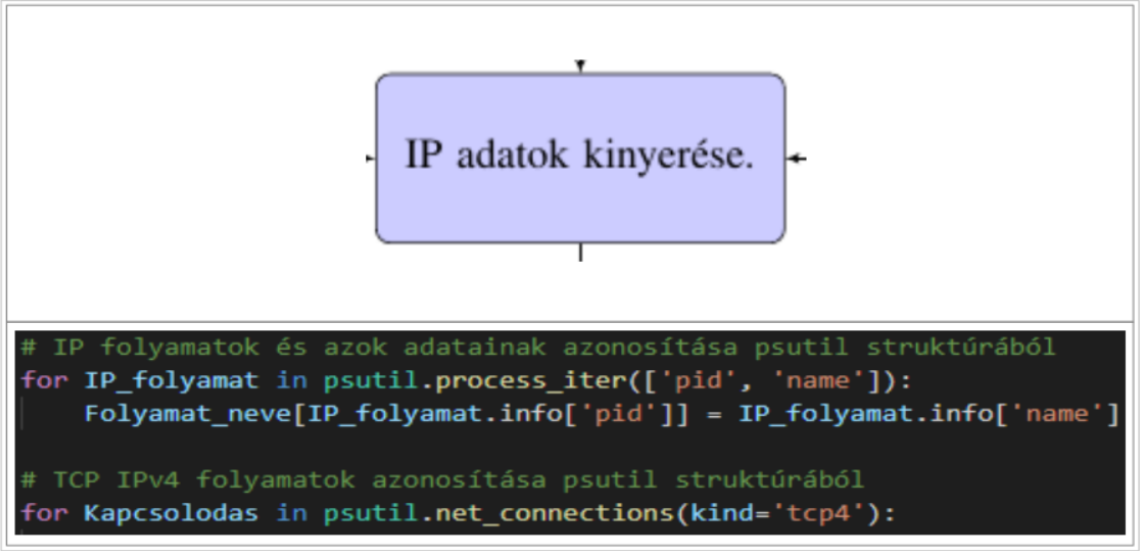

7.3. ábra. Program indulás (forrás: a szerző)

A szakdolgozati verzióban előfeltétel az adminisztratív jogosultsággal támogatott program indítás. Kilépési ponttal, program bezárási lehetőséggel a működési elvből adódóan nem rendelkezik az alkalmazás. A fejlesztett alkalmazás indulhat üzemezett feladatként vagy az operációs rendszer indulása után.
A fejlesztés adta lehetőségek használatával ez a korlát is megszüntethető a rendszer nevében futtatott Windows szolgáltatásra ültetéssel.

Hallgatózás.
A hallgatózás, mint folyamat a TCP/IP protocol stack, a protokollkészletbe tartozó, működéshez szükséges protokollok vizsgálatát jelenti. A 7.4. ábra segítségével látható, hogy milyen Python kódok használatával végezhető el a hálózati forgalomból történő kiolvasás.
Ellenőrzésre kerül az:

- az IP - internet protokoll -, amely nem megbízható csomag szolgálatatás a forrás- és a célgép között.
- a TCP - transmission control protocol -, amely megbízható, kétirányú árvitelt biztosít a bájt csomagoknak kapcsolat kiépítés után.

A szakdolgozati program verzióban nem kerül vizsgálatra a szintén az IP-re épülő, nem megbízható UDP - user datagram protocol - használatával kimenő adatcsomagok vizsgá-



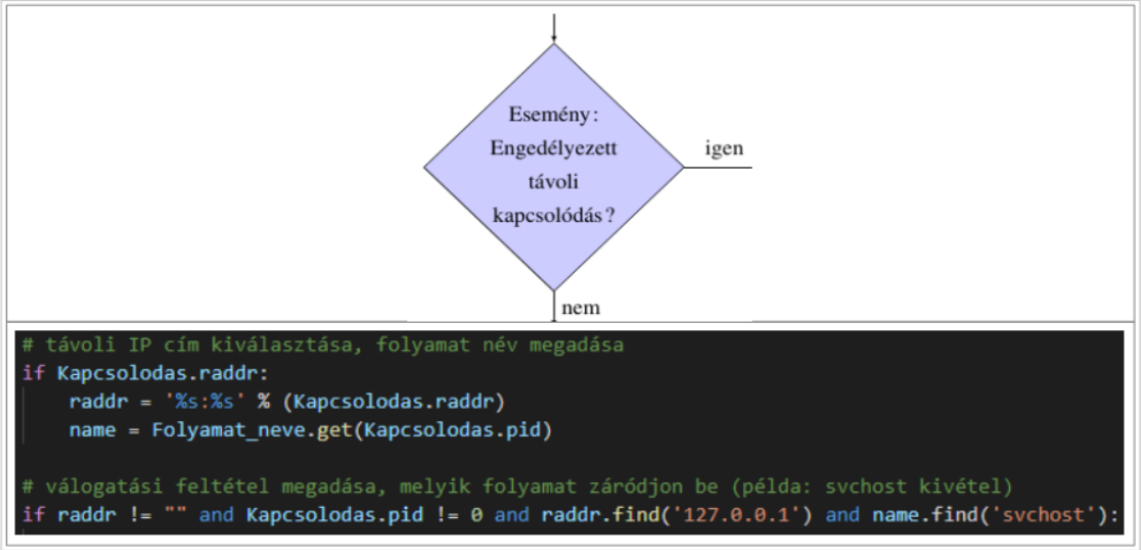

7.4. ábra. Forgalom kiolvasás (forrás: a szerző)

lata.
A hallgatózás lekódolt menete:

I. A psutil csomag ’process.iter’ függvényének használatával az adott kiolvasási idő alatt jelenlévő (futó, bezárt) folyamatok azonosítója (pid) és folyamat neve (name) megfogható.
II. A psutil csomag ’net.connections’ függvényének használatával a kiolvasás ideje alatt jelenlévő (futó, bezárt) folyamatok foghatóak meg, amelyek TCP alapúak.

A 4.3. ábrán látható, hogy "tcp4" alapú vizsgálat történik. Kódsor változtatással nemcsak az IPv4 címeket, hanem IPv6 címeket is lehet kezelni, illetve az UDP vizsgálatát is be lehet hozni a programba.

Döntés.
A döntés folyamata maga a működési mechanizmus. A 7.5. ábra mutatja a logikát, amely használatával a fejlesztett alkalmazás eldönti, mi történjen a kifelé irányuló folyamattal.
A program e szakasza a legfontosabb része az egésznek. Logika dönti el, mi legyen az előzőekben begyűjtött adatokkal. Ebben a szakaszban van igazán jelentősége az előzetes tervezésnek. Jelen esetben "bedrótozott" logika dönti el, melyik alkalmazással mi történjen.
A döntéshez szükség van egy következő változóra, ami a távoli IP cím átadására szolgál: ez a psutil csomag ’remote address’ (raddr) mezője. Kiolvasáskor ez a mező érték a távoli




7.5. ábra. Forgalom minősítés (forrás: a szerző)

IP cím.

A ’remote address’ elnevezés megtévesztő, mert azt az IP-t takarja, amelyik felé a kapcsolódás megtörtént. Ez lehet valóban LAN-on kívüli cím, de lehet saját cím is, mint a 127.0.0.1 IP cím. A 127.0.0.1 a ’localhost’ vagy ’loopback’ cím operációs rendszerrel ellátott számítógépekre értelmezve saját címet jelent, külső címmel történő kommunikációja nem engedélyezett. Speciális IP cím.

Az ’raddr’ mezőnek van egy párja, a ’laddr’ mező. Ez tartalmazza a lokális címet, ahonnan indul a kapcsolódás.

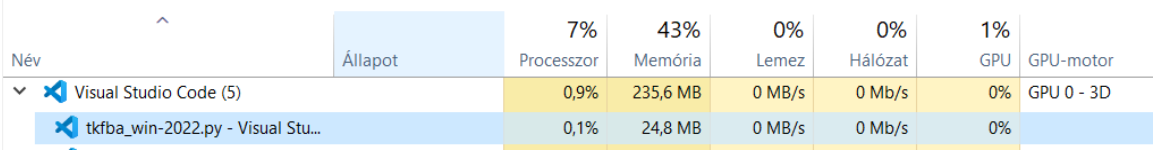
Program működés szempontjából ellenőrizni kell tehát a távoli IP cím megfelelőségét, ki kell szűrni a valóban távolra mutató címeket a működési logika szempontjából lényegtelen címek közül, mint például a 127.0.0.1 cím.

Szűrés közben további feltétel beállítása szükséges, mert előfordulhat olyan gyermek folyamat, amelyhez nincs folyamat azonosító rendelve.

Érdemes további szűrőfeltétel alkalmazása is, amellyel az operációs rendszer saját folyamatait kerüli ki az alkalmazás. A szakdolgozati alkalmazásban ez a Microsoft Windows ’svchost.exe’ rendszerfolyamata, amelyet nem fog meg az alkalmazás. (Az svchost.exe, mint rendszerkomponens veszélytelen, viszont léteznek vírusok, trójaiak, malware-ek, amelyek álcázhatják magukat ezzel a névvel.)

Összegezve a működést: rendelkezésre áll a kibányászott valódi távoli IP cím, szintén rendelkezésre állnak a válogatási feltételek, így megalkotható a védekezési logika.

A szakdolgozati alkalmazás esetén egyszerű logikai kifejezések láncolatával kifejezhető az elvárt eredmény:
ha (a távoli IP nem üres) és (folyamat azonosító is van) és (a távoli IP nem a localhost) és (svchost is hamis), akkor jöhet az elméleti ábra NEM ága, azaz a [4.] Beavatkozás és az újra ellenőrzés.
Bármely feltétel hamis állapota esetén az elméleti ábra IGEN ága érvényesül, az alkalmazás a következő kiolvasott sorra lép, azon a soron is lefut az ellenőrzés. Ezek az ellenőrzések addig tartanak, amíg az utolsó IP stack-ből kiolvasott sor elfogy. Ekkor a [2.] Hallgatózás funkció kezdődik újra.
A szakdolgozati alkalmazás esetében öt másodperc időtartam telik el két adatkinyerés között. Ez az időzítő ciklus rövidíthető akár egy másodpercre is, vagy ki is iktatható, mert az alkalmazás futása nem okoz erőforrás pazarlást (7.6. ábra):




7.6. ábra. Fejlesztett alkalmazás erőforrás használata (forrás: a szerző)

A 7.6. ábra adataiból kiderül, hogy minimális erőforrás felhasználás történik az alkalmazás működése közben, a kijelzett és lefoglalt 24-25 MB RAM a program futásának teljes szükséglete.

Beavatkozás

Ebben az állapotban a programkód már beazonosította és leválogatta a bezárni kívánt folyamat jellemzőit, következik a programozott leállítás (7.7. ábra). Az ábrán látható, hogy a psutil csomag 'Process' függvényének használatával változó értékébe kerül a bezárni kívánt folyamat azonosítója, majd egy utasítás használatával az úgynevezett 'kill process' eljárásra átadásra kerül az azonosító az operációs rendszer számára, amely végül megszünteti a folyamat, alkalmazás működését.
Az alkalmazások operációs rendszer általi bezárására több lehetőség is létezik, különböző eljárásokkal:

- terminate() - utasítás hatására befejeződik, lezáródik a folyamat.
- kill() - utasítás azonnal véget vet a kijelölt folyamatnak.

A 'terminate()' utasítás lefutása megakadályozható például egy fájl nyitva tartással, amelyet az alkalmazás nem tud lezárni, így függőben marad, a 'kill()' utasítás viszont azonnali


7.7. ábra. Alkalmazás bezárás (forrás: a szerző)

kilövést eredményez, nem számít, milyen adat- és memória töredékek maradnak hátra. A ’kill process’ azonnali és biztos módszer.
Illetve mégsem! Mert a ’kill’ is megkerülhető hekkeléssel.
A szoftverfejlesztés világában szinte minden lehetséges, ha ismerjük a szoftver és hardver környezetet. Amennyiben nem ismert a célkörnyezet, de szükséges annak ismerete, akkor fejleszthető olyan szkript, amely rendszerhívásokkal ellenőrzi a környezet, majd az eredményt lejegyezi, esetleg eljuttatja azokat a fejlesztőnek. Akár egy zombihálózat esetében, ahol a kártékony kód kommunikál a szerverével.

## 7.2. Védekezési mechanizmus megkerülése

A szakdolgozatban fejlesztett alkalmazás egy viszonylag egyszerű módszerrel igyekszik megakadályozni a nem kívánatos, internet felé irányuló, távoli hálózatba történő kommunikációt, szimulációként bemutatva a zombitevékenység elleni célzott védekezés egyik lehetséges aspektusát.
A tervezett működés egy programozott 'kill process' eljárásra épül, amely művelet kifejtésre és magyarázatra kerül egy feltételes gondolatmenettel együtt, amely leírja, hogyan kerülhető meg bezárási folyamat, így biztosítva például egy rosszindulatú kód futását, azaz a zombigép kártékony kódjának folyamatos működését.
A "feltételes" jelzőt fontos kiemelni a téma folytatása előtt, mert csupán gondolatmenetről van szó, afféle elméleti - hogyan hekkelmén - kifejtésről, nem cél kód analizáló eszközökön (például IDA Pro) keresztüli próba és bizonyítás.

Egy Microsoft Windows környezetben futó folyamat bezárásáról általában a PsTerminateProcess elnevezésű folyamat gondoskodik. (Persze több lehetőség is adott, például az ExitProcess elnevezésű folyamat használata.) [8] A bezárásra kijelölt process bitmap-jén (_KTHREAD tábla) ellenőrzésre kerül (KeRequestTerminationThread függvény) az úgynevezett APCQueueable flag. [9] Ha a flag értéke 1, akkor a process bezárható lesz, tehát logikai 1 értékre kell állítani.
Ezután a bezárásra szánt process azonosítót be kell szúrni az APC queue-ba, a Microsoft Windows rendszer aszinkron eljáráshívó algoritmusába, ami által tulajdonképpen 'halál listára' kerül az alkalmazás, amit ezután le is állít a rendszer. [10]

Amennyiben rosszindulatú tevékenység végrehajtása a cél a megtámadott számítógépen, azaz
# egy célzott támadás után,
# felhasználói hiba miatt,
# rendelkezésre áll már egy 'reverse shell' nyitotta 'backdoor',
akkor egy pár kódsoros script, driver segítségével a kártékony kód process-éhez tartozó APCQueueable flag értékét 0 értékre kellene állítani, így az nem kerül APC Queue-ba, ezáltal az operációs rendszer nem állítja le a kártékony kódot. Ezzel a technikával lehetne biztosítani a zombi tevékenység zavartalan működését.
A világhálón több 'APC Injection' támadás leírás is megtalálható kódsorral együtt. [11]
A gondolatmenetben nem kerültek vizsgálatra az internet szolgáltató védekezési lehetősé-

gei, az operációs rendszer védelmét ellátó végpontvédelmi rendszerek, hálózati biztonsági elemek, logelemző rendszerek hatása, amelyek egyesével vagy összességükben akadályt képeznek egy valós támadás megvalósításában.

# 8. TOVÁBBFEJLESZTÉSI LEHETŐSÉGEK

Általános felépítésű hálózatok esetében, mint például: a munkaállomás kapcsolódhat Internet irányba és LAN irányba is, a támadások felismerése és kivédése nehézkes folyamat különálló technológiák külön-külön alkalmazásával. Egy rendszer általában egy tevékenység kezelésére készül. A vírusvédelem vírus definíciós adatbázisból ellenőrzi és hasonlítja össze az ellenőrzött kódot. A tűzfal whitelist - blacklist és szabályok alapján ismeri fel, hogy engedélyezhető a forgalom, vagy sem. Mindkét példa esetén előre definiált, központi tudástárak alapján dolgoznak a rendszerek. Ami a tudástárban nem szerepel, az azonosítás esetén vagy emberi döntés elé kerül, vagy automatikusan feljegyzésre és például karanténba kerül. Jó eredmény akkor érhető el, ha a védelemre szánt megoldásokat közös, tanítható intelligencia követi, a felhasználó biztonságtudatos viselkedése mellett.
Ám lehet az üzemeltetett rendszer mindenre felkészült, elég egy jól kitalált rootkit, amely elrejtőzik a rendszerfájlok között, lehetővé téve a támadó számára a távoli belépést, és védtelenné válik a rendszer.

A szakdolgozatban bemutatott működési elvű alkalmazás nem rendelkezik az előbb említett magas fokú, tanulásra képes intelligenciával. [12] Saját tudásbázis, megadott szabály alapján működik. Az igen-nem / mehet-nem mehet szintre egyszerűsített működés:

- Előny és felhasználható - minden olyan zártabb jellegű hálózat esetében (pl.: röntgengépet vagy gyártósort támogató munkaállomás és hálózatuk), ahol minden kifelé irányuló, működéstől eltérő adatforgalom adatszivárgást feltételezhet.
  Az alkalmazás platform független, környezetre adaptálva (rendszer környezet, szabad/tiltott kommunikációs irány) automatikus működést végez, tűzfal megoldás hiánya és jelenléte esetén is.
- Hátrány - otthoni környezetben történő felhasználás esetén, hiszen az otthoni számítógépeken minden "is" megtalálható, mindenre használva vannak.
- Hátrány - olyan nyitott hálózatokban ahol autonóm módon működnek az eszközök, nincs szabályozott környezet, minden eszköz mindenre használva van, például egy kisvállalkozás pár gépet számláló belső hálózata.

Mit jelent az előny és hátrány említése?
Amennyiben egy munkahelyen megengedett a munkavégzésre használt számítógépeken az internethasználat - nincs hálózaton kívüli, elkülönített gép erre a célra -, akkor magasabb, tanulásra képes intelligenciával rendelkező technológiára van szükség, amely kapcsolatok elemzésével ellenőrzi a rendszert.

Egy honlap megnyitásakor tucatnyi távoli címmel épülhet fel kapcsolat, ezek elérését és figyelését is kezelnie kell egyfajta intelligenciának.
A legtöbb honlap HTTPS biztonsági protokoll használatával kommunikál. Ha a kártékony zombi kód TCP443 porton índít kapcsolódást a vezérlője felé, egyszerűen ki a tudja játszani a csomagszűrő tűzfalat.
Az előny és hátrány fogalmak ez esetben nem minősítésként szolgálnak, hanem rávilágítnak az informatikai biztonság szerteágazó világára. Arra a világra, amelyben egy új zero-day felboríthatja a biztonsági szakember addigi munkáját, és akár ismételt kockázatelemzéssel a kockázattal arányos módú újratervezés kivitelezését követelve meg tőle válaszként az új sérülékenységre.

A fejlesztett alkalmazás egy céleszköz, egy problémakörre egy szabályrendszerben adott megoldás. Amennyiben a környezet egyik tényezője változik, változik: a szabályrendszer, felborul az egyenlet, és más megoldásra lehet szükség. Ilyen esetben adatmódosítással, végső esetben fejlesztői beavatkozással adaptálható a megváltozott környezethez az alkalmazás új verziója.

A záró gondolatsorok irányt is mutatnak a szakdolgozati alkalmazás további életútjának. Néhány lehetséges fejlesztési lehetőség:

- "White list", "black list" indexelt adatbázis létrehozása az engedélyezett/tiltott IP-k számára.
  Adatbázis használatával dinamikusabbá válhat az alkalmazás kezelése, nem szükséges minden egyes változáshoz új verziót kiadni, a rendszerért felelős személy is módosíthat adatot. [13]
- IPv6 címek ellenőrzése.
  Engedélyezett és tiltott forgalmak ellenőrzése kód bővítéssel IPv6 formátumú kommunikációra is beállítható.
- Felhasználóbarát interfész fejlesztése.
  Pop-up, vagy egyéb figyelmeztető üzenet megjelenítése például blokkolt folyamat adataival.
- Eseménynapló rögzítése.
  Központosított naplóelemzés számára eseménynapló vezetése a működési eseményekről.
- Önvédelmi mechanizmus fejlesztése (például watchdog riasztás hozzáadása, Nagi-

os rendszerhez illesztés).

Nem tervezett alkalmazás leállás esetén figyelmeztető üzenet megjelenítése, küldése.

Talán a legtechnikásabb továbbfejlesztési lehetőség - figyelembe véve a titkosított csomagok forgalmazását - az IP csomagok vizsgálata, IP fejléc ellenőrzése, és az abból szerzett információk alapján minősíteni a kimenő forgalmat.

# 9. ÖSSZEFOGLALÁS

Szakdolgozatomban arra kerestem választ, hogy a zombihálózat ellen milyen módon védekezhetünk, illetve lehetséges-e egyedi, saját módon védekezni ellene. Vizsgáltam azt is, ha lehetséges a saját elképzelés alapján történő védekezés, akkor az milyen megoldással valósítható meg.

A zombihálózatok működési elvének birtokában a védekezési lehetőségek feltárása után arra a megállapításra jutottam, hogy a kártékony zombi kódok közös működési jellemzője a kapcsolódási folyamat indítása a zombivezérlő felé. Erre a tulajdonságra építve saját fejlesztésű alkalmazás készíthető.

Az egyedi fejlesztésben jól használható védekezési lehetőséget találtam, ennek működőképességét a szakdolgozatban saját fejlesztésű alkalmazáson keresztül szemléltettem.

A szakdolgozat értékét abban látom, hogy a felvetett zombihálózat problémára olyan használható megoldást ad, amely újszerű módon akadályozza a kártékony zombihálózat működését.

# 10. SUMMARY

In my thesis, I was looking for how to protect against the zombie network or whether it is possible to protect against it in a unique way. I also investigated the solution if it is possible to protect it from my own idea.

In accordance with the principle of operation of zombie networks, after exploring the possibilities of defense, I came to the conclusion that the common operating feature of malicious zombie codes is the initiation of the connection process to the zombie controller. Based on this property, can create a self -developed application.

In the individual development I found a useful defense option, the functionality of which I illustrated in the thesis through a self-developed application.

I see the value of the thesis in the fact that the zombie network raised provides a solution to a problem that is new to the operation of the malicious zombie network.

# Hivatkozások


[1] Reuters, „U.s. fbi says it disrupted russian hackers.” Megtekintve: https://www.reuters.com/world/us-fbi-says-it-foiled-cyberattack-by-russian-hackers-2022-04-06/ (2022/04/06).

[2] L. Faragó and T. Mészáros, „Anonim rendszer botnet forgalomfelismerése és szűrése.” Megtekintve: https://math.bme.hu/~slovi/farago_meszaros_TDK.pdf (2021/05/21).

[3] Z. Illyési, „Botnetek kialakulása, használatuk, trendjeik.” Megtekintve: http://hadmernok.hu/archivum/2008/2/2008_2_illesi.pdf (2022/04/18).

[4] IHInet, „Botnet avagy zombi gépek.” Megtekintve: http://ihimulti.hu/erdekes-hirek/botnet-avagy-a-zombik.html (2022/05/08).

[5] I. Csizmadia Darab, „Zombihálózatot tessék.” Megtekintve: https://antivirus.blog.hu/2019/03/22/zombihalozatot_tessek (2021/05/21).

[6] I. Comodo Group, „Internet security pro.” Megtekintve: https://www.comodo.com/home/internet-security/internet-security-pro.php (2022/05/07).

[7] I. . c. ReadTheDocs, „psutil dokumentáció.” Megtekintve: https://psutil.readthedocs.io/en/latest/ (2022/02/18).

[8] Code13, „Windows api - user mode matched kernel mode.” Megtekintve: https://code13.tistory.com/123 (2021/12/12).

[9] B. Dang, A. Gazet, E. Bachaalany, and S. Josse, *Practical Reverse Engineering: x86, x64, ARM, Windows Kernel, Reversing Tools, and Obfuscation*. John Wiley and Sons., 2014. ISBN 978-1-118-78731-1.

[10] Microsoft, „Windows apc.” Megtekintve: https://docs.microsoft.com/en-us/windows/win32/sync/asynchronous-procedure-calls3 (2021/12/12).

[11] IredTeam, „Code injection.” Megtekintve: https://www.ired.team/offensive-security/code-injection-process-injection/early-bird-apc-queue-code-injection (2022/03/02).

[12] I. Vectra AI, „Vectra security applience.” Megtekintve: https://www.vectra.ai/products/platform (2022/14/02).

[13] P. Ször, *A vírusvédelem művészete.* SZAK Kiadó, 2010. ISBN 978-963-9863-16-3.

# ÁBRAJEGYZÉK

# TÁBLÁZATOK JEGYZÉKE

# függelék A Forráskód

```
# Tavoli-kapcsolodas-folyamat-bezaro-alkalmazas_Win-2022
# Szakdolgozat celjabol fejlesztett bemutato celu alkalmazas
# AProgs, 2022

# csomagok beolvasasa
# protokollok kezelese:
# from socket import AF_INET, SOCK_STREAM, SOCK_DGRAM
# rendszer folyamatok, adatok kezelese:
import psutil
# idozites kezelese:
import time

# foprogram belepesi pont
def main():
# valtozok elore definialasa, nullazas
Folyamat_neve = {}
raddr = ""

# IP folyamatok es azok adatainak azonositasa psutil strukturabol
for IP_folyamat in psutil.process_iter(['pid', 'name']):
Folyamat_neve[IP_folyamat.info['pid']] = IP_folyamat.info['name']

# TCP IPv4 folyamatok azonositasa psutil strukturabol
for Kapcsolodas in psutil.net_connections(kind='tcp4'):

# tavoli IP cim kivalasztasa, folyamat nev megadasa
if Kapcsolodas.raddr:
raddr = '%s:%s' % (Kapcsolodas.raddr)
name = Folyamat_neve.get(Kapcsolodas.pid)

# valogatasi feltetel megadasa, melyik folyamat zarodjon be
#(pelda: svchost kivetel)
if raddr != "" and Kapcsolodas.pid != 0 and raddr.find('127.0.0.1')
```

```
and name.find('svchost'):
# folyamat tulajdonsagok kiirasa - csak kepernyo modban lathato
print (raddr, Kapcsolodas.pid, name)
# tiltott kommunikacio definialasa, kiirasa kepernyore es bezaras
Program_bezar = psutil.Process(Kapcsolodas.pid)
print (Program_bezar)
#Program_bezar.terminate() - terminate() megakadalyozhato,
#kill() nem megakadalyozhato muvelet
Program_bezar.kill()

# 5 masodperc varakozas stack ujraolvasas elott
time.sleep(5)

# kezdesre ugras
main()

#belepesi pont a programba
if __name__ == '__main__':
main()
```