ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

**NEUMANN JÁNOS
INFORMATIKAI KAR**

DIPLOMAMUNKA

OE-NIK Hallgató neve: **Gerőfi Máté Attila**

2022 Hallgató törzskönyvi száma: **T/007257/FI12904/N**

Óbudai Egyetem
Neumann János Informatikai Kar
Kiberfizikai Rendszerek Intézet

DIPLOMAMUNKA
FELADATLAP

Hallgató neve: **Gerőfi Máté Attila**
Törzskönyvi száma: T/007257/FI12904/N
Neptun kódja: [redacted]

A diplomamunka címe:

Gépi tanulási algoritmusok fejlesztése Hunting ELK rendszeren
Development of machine learning algorithms on Hunting ELK

Intézményi konzulens: Balázsné Dr. Kail Eszter
Külső konzulens:

Beadási határidő: 2021. december 15.

A feladat

Manapság az informatikai biztonság fontosságát nem lehet eléggé kihangsúlyozni, a világ folyamatainak digitalizációja során sok visszaélésre van lehetőség. Az ipar számtalan fizetős biztonsági alkalmazást kínál, azonban kevés jól működő nyílt forráskódú megoldással találkozhatunk. Egy ilyen, fejlesztésre szabadon elérhető komplex rendszer például a Hunting ELK, amely fejlett analitikai lehetőségekkel és gépi tanulási opciókkal is rendelkezik.
A szakdolgozat célja hálózati forgalom vizsgálatán alapuló behatolás detektáló alkalmazás fejlesztése, mely a lehető legkevesebb rendszerterheléssel képes a már ismert veszélyek, valamint a még ismeretlen támadások, ártalmas kódok felismerésére.
A hallgató feladata alapos irodalomkutatás alapján a behatolás detektáló rendszer megvalósítása és a Hunting ELK rendszerbe való implementálása.

A diplomamunkának tartalmaznia kell:

- Hunting ELK rendszer feltérképezése,
- már létező, gépi tanulási algoritmusokon alapuló biztonsági rendszerek megoldásainak ismertetése,
- a kiválasztott módszer alapján a rendszer megtervezése és implementálása,
- a rendszer tesztelése, eredmények értékelése,
- továbbfejlesztés lehetőségeinek vizsgálata.

ÓBUDAI EGYETEM * Neumann János Informatikai Kar * PH 1. Budapest

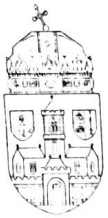
.....
Dr. habil. Lovas Róbert
intézetigazgató

A diplomamunka elévülésének határideje: **2023. december 15.**
(OE TVSz 55.§ szerint)

A diplomamunkát 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, 2022.05.15 .....

.....
hallgató aláírása

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar
9. sz. melléklet

KONZULTÁCIÓS NAPLÓ

Hallgató neve: Gerőfi Máté Attila .....

Neptun kód: [redacted] .....

Tagozat: Esti .....

Telefon: [redacted]

Levelezési cím (pl: lakcím): [redacted]

Szakdolgozat / Diplomamunka[1] címe magyarul:

Gépi tanulási algoritmusok fejlesztése Hunting ELK rendszeren .....

Szakdolgozat / Diplomamunka[2] címe angolul:

Development of machine learning algorithms on Hunting ELK.....

Intézményi konzulens: Balázsné Dr. Kail Eszter.....

Külső konzulens: .....

Kérjük, hogy az adatokat nyomtatott nagybetűkkel írja!

Alk.	Dátum	Tartalom	Aláírás
1.	2020.09.21	DIPLOMAMUNKA FELÉPÍTÉSE	Kail
2.	2020.11.10	ELŐZŐ TÉMÁKRÓL JAVASLATOK	Kail
3.	2020.11.24	KORREKCIÓK, JAVÍTÁSI LEHETŐSÉGEK	Kail
4.	2020.12.10	TERVEZÉSI MUNKA ÁTBESZÉLÉSE	Kail

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

A hallgató a Szakdolgozat I. / Szakdolgozat II. (BSc) vagy Diplomamunka 1 / Diplo-mamunka 2 / Diplomamunka 3 / Diplomamunka 4[3] tantárgy követelményét teljesítette, beszámolóra / védésre[4] bocsátható.

Budapest, 2020.12.10.....

Kail
.....
intézményi konzulens

[1] Megfelelő aláhúzandó!
[2] Megfelelő aláhúzandó!
[3] Megfelelő aláhúzandó!
[4] Megfelelő aláhúzandó!

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar
9. sz. melléklet

KONZULTÁCIÓS NAPLÓ

Hallgató neve: Neptun kód: Tagozat:

Gerőfi Máté Attila ..... [redacted] ..... Esti .....

Telefon: Levelezési cím (pl: lakcím):

[redacted]

Szakdolgozat / Diplomamunka[1] címe magyarul:

Gépi tanulási algoritmusok fejlesztése Hunting ELK rendszeren .....

Szakdolgozat / Diplomamunka[2] címe angolul:

Development of machine learning algorithms on Hunting ELK.....

Intézményi konzulens: Külső konzulens:

Balázsné Dr. Kail Eszter..... .....

Kérjük, hogy az adatokat nyomtatott nagybetűkkel írja!

Alk.	Dátum	Tartalom	Aláírás
1.	2021.03.04	TÉMÁVAL KAPCSOLATOS KIEGÉSZÍTÉSEK	Kail
2.	2021.04.08	EGYETEMI TÁVOLI ELÉRÉS	Kail
3.	2021.05.04	MEGVALÓSÍTÁSI LEHETŐSÉGEK	Kail
4.	2021.05.11	DOKUMENTÁCIÓ ÁTBESZÉLÉSE	Kail

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

A hallgató a Szakdolgozat I. / Szakdolgozat II. (BSc) vagy Diplomamunka 1 / Diplo-mamunka 2 / Diplomamunka 3 / Diplomamunka 4[3] tantárgy követelményét teljesítette, beszámolóra / védésre[4] bocsátható.

Budapest, 2021.05.11.....

Kail
.....
intézményi konzulens

[1] Megfelelő aláhúzandó!
[2] Megfelelő aláhúzandó!
[3] Megfelelő aláhúzandó!
[4] Megfelelő aláhúzandó!

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar
9. sz. melléklet

KONZULTÁCIÓS NAPLÓ

Hallgató neve: Neptun kód: Tagozat:

Gerőfi Máté Attila ..... [redacted] ..... Esti .....

Telefon: Levelezési cím (pl: lakcím):

[redacted]

Szakdolgozat / Diplomamunka[1] címe magyarul:

Gépi tanulási algoritmusok fejlesztése Hunting ELK rendszeren .....

Szakdolgozat / Diplomamunka[2] címe angolul:

Development of machine learning algorithms on Hunting ELK.....

Intézményi konzulens: Külső konzulens:

Balázsné Dr. Kail Eszter..... .....

Kérjük, hogy az adatokat nyomtatott nagybetűkkel írja!

Alk.	Dátum	Tartalom	Aláírás
1.	2022.03.10	TARTALOM KIEGÉSZÍTÉSE	Kail
2.	2022.04.07	FORMAI KÖVETELMÉNYEK ÁTNÉZÉSE	Kail
3.	2022.05.05	HELK RENDSZER LEHETŐSÉGEI	Kail
4.	2022.05.13	TOVÁBBFEJLESZTÉSI LEHETŐSÉGEK	Kail

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

A hallgató a Szakdolgozat I. / Szakdolgozat II. (BSc) vagy Diplomamunka 1 / Diplo-mamunka 2 / Diplomamunka 3 / Diplomamunka 4[3] tantárgy követelményét teljesítette, beszámolóra / védésre[4] bocsátható.

Budapest, 2022.05.13.....

Kail
.....
intézményi konzulens

[1] Megfelelő aláhúzandó!
[2] Megfelelő aláhúzandó!
[3] Megfelelő aláhúzandó!
[4] Megfelelő aláhúzandó!

Abstract

Az internet széleskörű elterjedése és lehetőségei magával vonzzák a veszélyeket is. Fontos kiemelni, hogy a hagyományos, ismert veszélyek felderítésével nem vagyunk védettek, mivel az informatika fejlődése miatt mindig akadnak újabb sebezhetőségek, amikről nem szerzünk időben tudomást. Szakdolgozatomban bemutatok egy Security Operation Center (SOC – Biztonsági Műveleti Központ) -ként működő HELK rendszert, melyen Elasticsearch, Logstash és Kibana modulok segítségével egy behatolásészlelő megoldást lehet létrehozni. Ennek tudnia kell elemezni a hálózati forgalmat, és saját elemzéseinek használata után jeleznie kell a támadásokat, majd grafikus formában is láthatóvá kell tennie mivel kapcsolatos anomáliát talált. Ezt követi gépi tanulási algoritmusok minőségének vizsgálata behatolásészlelő rendszer pontossága szempontjából. Eleinte többklasszifikációs megközelítéssel, majd a legjobb értékeket elértnek bináris klasszifikációs változata következik. Az eredmények kiértékelésekor láthatjuk, hogy az általunk vizsgált modellek közül a Random Forest volt a leghatékonyabb.

Tartalom

1 Bevezetés

A rosszindulatú szoftverek jelentős fejlődése az utóbbi években növeli a behatolás észlelő rendszerek fejlesztésének fontosságát. A támadások kifinomultabbá váltak, és a legfontosabb feladat az ismeretlen és megtévesztő, rosszindulatú programok azonosítása, mivel az idő haladtával készítőik egyre jobb technikákat alkalmaznak a felismerés kijátszására és az információ elrejtésére, megnehezítve ezzel egy behatolás észlelő rendszer általi azonosítást. Ezen kívül növekedett az úgynevezett nulla-napos (zero-day) támadások száma, melyeknél az alkalmazások olyan sebezhetőségét használják ki, ami még nem került publikálásra, az alkalmazás, illetve szoftver fejlesztője nem tud róla, vagy nincs rá jelenleg javítás, így közvetlen kockázatot jelent, míg be nem foltozzák. Ezek miatt a számítógépes biztonság az életünk részévé vált, mivel naponta használunk információs technológiával kapcsolatos eszközöket, alkalmazásokat [1] [2].

Az AV-Test szerint 2011-től 2020. november 20-ig több mint 1119 millió kártékony programot azonosítottak [3] míg 2019 3. negyedévében ezen támadásokból a nulladik napi változat eléri az 50%-ot [4], legtöbb Amerikát érinti. A „National Technology Security Coalition” 2020-as, 2019. évre vonatkozó statisztikája alapján globálisan nézve a crypto bányászattal kapcsolatos támadások a leggyakoribbak 38%-al, de 28%-uk botnettel, 27%-uk mobillal, 18%-uk banki támadással, és adatlopással kapcsolatos kártékony tevékenységet végez [5].

A statisztikák alapján látható, hogy a fenyegetések világszerte elterjedtek, és egy egyszerű kompromittálás megzavarhatja az üzleti vállalkozások alapvető szolgáltatásait vagy létesítményeit. A kiberbűnözők céljai között megtalálható az információlopás, törvénytelen bevételek eltulajdonítása, és újabb célpontok keresése. Az ismeretlen, kifinomult támadások miatt szükség van egy olyan összetett, komplex rendszerre, ami a hagyományos tűzfalak és vírusirtók képességeit jelentősen meghaladja.

2 Irodalomkutatás

2.1 Behatolás észlelő rendszerek

Behatolást meghatározhatjuk úgy, mint bármilyen jogosulatlan tevékenységet, ami károsít egy információs rendszert. Ez azt jelenti, hogy veszélyezteti az információ bizalmasságát, sértetlenségét, és rendelkezésre állását (CIA modell). Olyan tevékenységek is idetartoznak, amelyek a számítógépes szolgáltatásokat a törvényes felhasználókra nézve elérhetetlenné teszik. Egy IDS (Intrusion Detection System) egy olyan szoftver, hardware, vagy kombinált rendszer, ami a számítógép irányába végzett rosszindulatú tevékenységeket azonosítja, elemzi, -amire egy hagyományos tűzfal nem képes-, ezáltal a megfelelő lépéseket meg lehet tenni a számítógépes rendszer biztonságának érdekében. Több csoportjuk van, azonban ismertebbek a szignatúra alapú behatolás észlelő rendszerek (Signature-based Intrusion Detection System - SIDS), illetve az anomália alapú behatolás észlelő rendszerek (Anomaly-based Intrusion Detection System) [1].

2.2 Szignatúra-alapú behatolás észlelő rendszerek

Ezek mintaegyezésen alapulnak, ez alapján ismernek fel egy támadást. Ha egy támadás szignatúrája megtalálható a SIDS-nek az aláíró adatbázisában, akkor jelez, vagy riaszt. Ez esetben például a gazdagép naplóellenőrzése alapján, a rosszindulatú parancsok sorozatát lehet felismerni, amennyiben ez megtalálható azadatbázisban. Ez alapján ez másképpen tudásalapú detektálásnak is el lehet nevezni, működése hasonlít egy hagyományos vírusdefiníciós adatbázissal rendelkező vírusirtóéhoz. Ezek hatékonysága ismert támadások ellen megfelelő [6], viszont 0-napi támadások ellen nem, tekintettel arra, hogy az adatbázisban nem létezik a támadás szignatúrája [1].

A hagyományos SIDS hálózati csomagokat vizsgál, és összehasonlítja azt az adatbázisban levő szignatúrákkal, viszont ez a fajta technika nem tud olyan támadásokat azonosítani, melyek több csomagot átfognak. A modern kártékony programok kifinomultsága ezáltal szükségessé teheti egyszerre a több csomagban levő szignatúra begyűjtését. Ezeknek az szignatúráknak a létrehozására több módszer lehetséges, például van ahol állapotgépekkel készítik el [7] , vagy formális nyelvi karakterlánc mintaként [8]. A nulladik napi támadások emelkedett száma ezen rendszerek hatékonyságát folyamatosan csökkenti, tekintettel ara, hogy nincs hozzájuk előzetesen szignatúra. A célzott támadások, és szignatúráktól eltérő változatai tovább csökkentik a SIDS eredményességét. Egy lehetséges megoldás az anomália alapú behatolás észlelő rendszer, ami azt tárolja el, hogy hogyan kell viselkednie valaminek, ettől történő eltérés esetén riaszt [1].

2.3 Anomália alapú behatolás észlelő rendszerek

Az AIDS megoldása túllép a SIDS problémáján. Itt a rendszer normális működésének modelljét gépi tanulással, tudásalapú módszerekkel, vagy statisztika alapon hozzák létre, a megfigyelt viselkedés a tipikus, tanulttól való eltérés esetén anomáliának minősül, ami támadást jelenthet. Az AIDS fejlesztése két részből áll, a tanulási fázis, és a tesztelő fázis. Míg a tanulási fázisban egy normális hálózati forgalmi profilt használnak a betanításhoz, a tesztelési fázisban egy új adathalmazt használnak a rendszer fejlesztésére, hogy képes legyen korábban nem látott támadásokat is felismerni [1].

Az AIDS fő előnye a nulla napos támadások felismerésének képessége, mivel nem függ az aláírói adatbázistól, továbbá belső felhasználók esetén, a tipikus viselkedéstől való eltérést is tudja jelezni, ami behatolók munkáját megnehezíti, mivel nehéz imitálni egy személy rendszeres tevékenységét. Azonban az AIDS hátránya, hogy magas hamis pozitív riasztások is előfordulhatnak, mivel a nem megszokott, de új, szükséges tevékenységeket behatolásnak nézheti [1].

Mivel nincs egyértelmű osztályozása az AIDS-nek, ezért 5 alosztályt vizsgálunk:

- statisztika alapú
    - hálózati forgalmat statisztikai algoritmusok segítségével elemzi, nagy mennyiségű statisztika kell hozzá
- minta alapú
    - adatokban levő karaktereket, felépítést, mintát azonosítja, könnyű implementálni
- szabály alapú
    - egy támadási mintát használ a lehetséges támadás azonosításához, a szabályok mintaillesztése miatt sok a számítási igénye, és sok szabály kell az összes támadás felismeréséhez, de magas észlelési aránnyal, és alacsony álpozitív felismeréssel rendelkezik
- állapot alapú
    - események követése alapján tudja azonosítani a támadást, alacsony álpozitív arány
- heurisztika alapú
    - azonosít minden rendellenes tevékenységet, ami a szokásos tevékenységeken kívül esik, szüksége van tanulásra és tapasztalatra [1]

2.4 Behatolási adatforrások

	Előnyök	Hátrányok	Adatforrás
HIDS	• A HIDS ellenőrizheti a végpontok közötti titkosított kommunikációs viselkedést • Nincs szükség külön hardverre • A behatolásokat a gazdagép fájlrendszer, a rendszerhívások vagy a hálózati események ellenőrzésével észleli • Minden csomagot újra összerak • A teljes elemet nézi, nem csak a közvetítéseket	• Késések a támadások jelentésével • A gazdagép erőforrását fogyasztja • Telepíteni kell minden gazdagépre • Csak azt a gépet tudja figyelni, amelyre telepítve van	•Naplózza a rekordokat, a naplófájlokat, az Application Program Interface-t (API), a szabálymintákat, a rendszerhívásokat
NIDS	•A hálózati csomagok ellenőrzésével észleli a támadásokat • Nem szükséges telepíteni minden gazdagépre • Különböző gépeket ellenőrizhet egyszerre • Képes felismerni a hálózati protokollok legszélesebb tartományát	• Kihívás a titkosított forgalom támadásainak azonosítása • Dedikált hardverre van szükség • Csak a hálózati támadások azonosítását támogatja • Nehezen elemezhető nagysebességű hálózat • A legsúlyosabb fenyegetés a bennfentes támadás	•Egyszerű hálózatkezelési protokoll (SNMP) •Hálózati csomagok (TCP / UDP / ICMP), •Vezetési információs bázis (MIB) • Router NetFlow rekordok

1. ábra – HIDS és NIDS összehasonlító táblázat [1]

Az előző két említett módszer, SIDS és AIDS az IDS-eket a behatolások azonosítására használt módszerek alapján csoportosította, azonban másik megközelítéssel, a bemeneti adatforrások alapján észlelhető, normálistól eltérő tevékenységek azonosításával is lehet kategorizálni. Ez alapján két IDS csoportot különböztetünk meg, melyek a gazdagép alapú (Host-based IDS) HIDS , és hálózati alapú

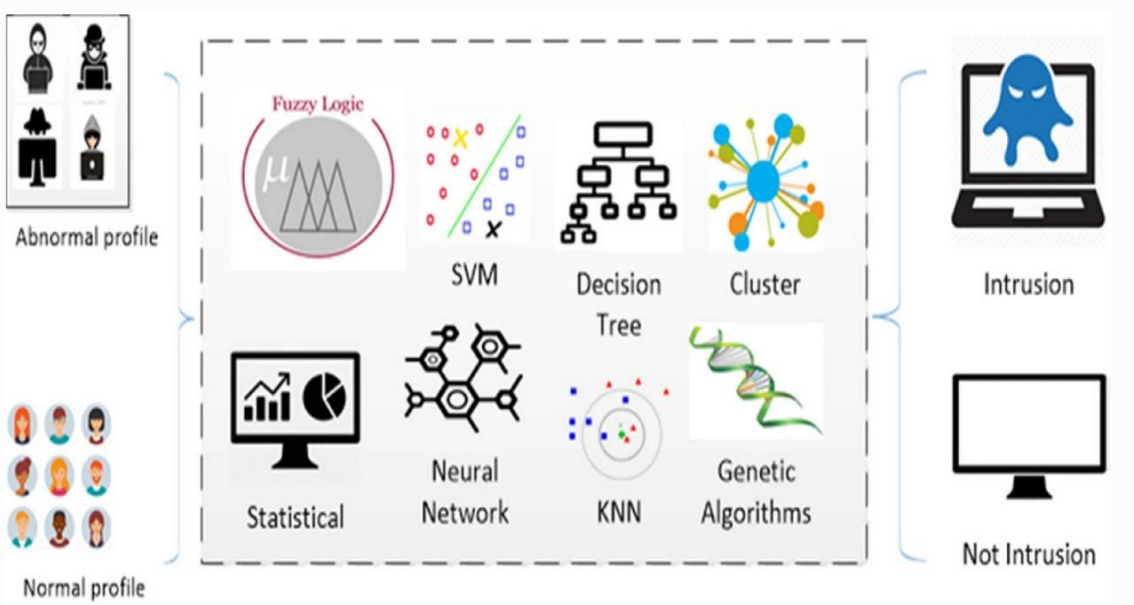
(Network-based IDS) NIDS. A HIDS a gazdagépről származó adatot ellenőrzi, beleértve az operációs rendszert, logokat, és azokat a belső támadásokat, melyek nem tartalmaznak hálózati forgalmat, míg a NIDS a hálózati forgalmat vizsgálja csomagkinyeréssel. Ezt fel lehet használni hálózathoz csatlakoztatott gépek ellenőrzésére is, és külső rosszindulatú tevékenységek figyelemmel kísérésére, melyek külső forrásból származnak, korai fázisban vannak, azaz még nem terjedtek át másik gépre. Azonban nagy sávszélességű hálózatban a nagy mennyiségű adatok miatt az ellenőrzés korlátozott kapacitással tud működni, egy lehetséges megoldás a NIDS, HIDS és tűzfal szimultán használata, amely megvédhet a külső és belső támadások ellen. Az 1. ábra a két technika közötti különbséget öleli fel [1].

2.5 AIDS implementálása

Az AIDS technikákat 3 fő csoportba lehet osztani. Statisztikán alapuló, tudás alapú, és gépi tanuláson alapuló. A statisztikán alapuló magában foglalja az adathalmaz összes adatrekordjának begyűjtését és vizsgálatát, valamint készít egy statisztikai modellt a normál felhasználói viselkedésből. A tudásalapú a rendszeradatokból megpróbálja azonosítani a kért tevékenységeket, mint például a protokoll specifikációkból és hálózati forgalmi példányokból, míg a gépi tanulási módszerek a tanító adatbázisból szerzik a mintaillesztési képességüket. Ezeknek a főosztályoknak még több alosztálya van. Mivel a szakdolgozatomban gépi tanulási algoritmusokkal kívánom megvalósítani a rendszert, ezért ezt a csoportot vizsgálom meg alaposan [1].

2.6 Gépi tanulási technikákon alapuló AIDS

A gépi tanulás egy olyan folyamat, ahol tudást nyerünk ki nagy mennyiségű adatból. A gépi tanulási modellek szabályok, módszerek, vagy összetett átviteli függvények készletéből állnak. Ezeket lehet használni viselkedésfelismerésre, viselkedés előrejelzésre, vagy adott minták keresésére az adatban [9].



2. ábra – Gépi tanulási technikákon alapuló AIDS fajtái [1]

A gépi tanulási alapon létrehozott IDS-ek fő célja a minták felderítése, és az adathalmaz alapján egy behatolás észlelő rendszerek építése. Ezt alapul véve ennek sok fajtája létezik, mint ahogy a 2. ábrán látható. Fő ismérvük, hogy nagyobb pontossággal, és kisebb emberi tudással segítenek nekünk egy behatolás észlelő rendszert kiépíteni. Általában két féle gépi tanulási módszert különböztetünk meg, a felügyeltet és a felügyelet nélkülit. Utóbbi egy olyan fajta technika mely bemeneti adathalmazból érdekes információt tud kinyerni osztály címkék nélkül, a bemeneti adatpontokat véletlenszerű változók halmazaként kezeli. IDS fejlesztés szempontjából a felügyelet nélküli tanításnál a behatolások azonosítása címkézetlen adatokkal való modell tanítást jelent [1].

2.7 Felügyelt tanulás behatolást észlelő rendszerben

Ezek a technikák címkézett képzési adatok felhasználásával észlelik a behatolásokat, általában két szakaszból állnak, melyek a tanulás, és tesztelés. Tanulási fázisban a releváns tulajdonságokat és osztályokat azonosítja, majd ezekből az adatokból tanul az algoritmus. Itt minden rekord egy pár, ami tartalmaz egy hálózati, vagy gazdagép adatforrást, és egy hozzá tartozó kimeneti értéket, azaz címkét, ami behatolás vagy normálist jelent. Ezután megkezdődhet a felesleges tulajdonságok/funkciók eltávolítása. A képzési adatokat használva a kiválasztott tulajdonságokhoz, egy felügyelt tanulási technikával az osztályozót betanítjuk, hogy megtanulja a bemeneti adatok, és a kimeneti értékek közötti kapcsolatot. A tesztelő fázisban a betanított modellt használják az ismeretlen adatok a behatolás, vagy a normális osztályba sorolására. A kapott osztályozó, ami ezzel modellé alakul, az adott bejövő adatokat a megfelelő osztályba tudja sorolni [1].

2.7.1 Genetikus algoritmusok

Az evolúciót veszik alapul, az optimalizálás heurisztikus megközelítése, minden lehetséges megoldás egy bit vagy kromoszómasorozatként van számontartva, a megoldások időről időre jobbak, köszönhetően a szelekciós és szülési operátorok segítségének, természetesen a fittebb egyedeket figyelembe véve. A genetikai algoritmusnak a behatolás besorolási problémájára való alkalmazásakor a kromoszóma kódolásának tipikusan két típusa van: az egyik a klaszterezés szerint bináris kromoszóma kódolási módszer létrehozására szolgál; egy másik a klaszterközpontot (a klaszterező prototípus mátrixot) egész számot kódoló kromoszómával határozza meg [1].

2.7.2 Mesterséges Neurális hálók

A mesterséges neurális hálók (Artificial Neural Networks - ANN) legszélesebb körben alkalmazott gépi tanulási módszer, sikeresnek bizonyult több rosszindulatú program felderítésében. A tanulás során leggyakrabban a backpropagation algoritmust használják, mely a hálózat hibájának gradiensét értékeli, annak módosítható súlyaihoz viszonyítva. A legfőbb probléma az ANN-es megoldásnál az észlelési pontosság, a ritka támadások esetében. A ritkább támadásokhoz nincs megfelelő adatkészlet, amiből oktatni lehetne, így nehezen tudja az ANN megtanulni ezeknek a tulajdonságait, csökkentve ezzel az észlelési pontosságot. Ez akkor tud rendkívül nagy gond lenni, ha egy user to root támadás esetén a sima felhasználó megszerzi a legmagasabb felhasználó jogosultságokat, és rosszindulatú tevékenységet végez az adott eszközön. Az ANN gyakran szenved továbbá helyi minimumoktól, ami növelheti a tanulási időt, azonban egy vagy több rejtett réteggel nemlineáris modelleket tud készíteni, amelyek a bemeneti attribútumok és osztályozási címkék közötti összetett kapcsolatokat képesek rögzíteni. Az ANN-eknek továbbá van 2 nagyobb változata, az ismétlődő (rekurrens), és konvolúciós neurális hálók, melyeket behatolás észlelő hálózatokban is lehet használni [1].

2.8 Gépi tanulási algoritmusok

Több gépi tanulási algoritmus merül fel behatolásészlelő rendszerek tervezésekor. Egyes algoritmusok ismertebbek, én ezeket veszem most figyelembe.

2.8.1 Logisztikus regresszió

20. század elején használták biológiai tudományokban, ezután számos társadalomtudományi alkalmazásban. Akkor használjuk, ha a függő változó (célváltozó) kategorikus. 3 típusa van, a bináris logisztikus regresszió, mely esetében a kategorikus válasznak csak 2 eredménye lehet, azaz például egy levél spam-e, vagy nem. Másik a multinominális logisztikus regresszió, mely esetében 3 vagy több kategória van, sorrend nélkül, például 3 közül melyik a preferáltabb étel. Harmadik az ordinális logisztikus

regresszió, melynél a sorrendet is figyelembe vesszük, erre példa egy film értékelése 1-től 5-ig. Ahhoz, hogy eldöntsük melyik osztályba tartozik az adat, egy küszöbértéket állíthatunk be. A döntési határ lehet lineáris, vagy nem lineáris [10].

2.8.2 Naive Bayes

A naive bayes osztályozó egy valószínűségi gépi tanulási modell, ami a Bayes-tételen alapul. Segítségével meg tudjuk határozni A bekövetkezésének valószínűségét, feltéve, hogy B bekövetkezett:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Itt B a bizonyíték, A pedig a hipotézis. Az itt megfogalmazott feltételezés az, hogy a prediktorok/jellemzők függetlenek. Vagyis az egyik jellemző jelenléte nem befolyásolja a másikat. Ezért nevezik naivnak.

Három fajtája van, egyik a multinominális naiv Bayes, melyet leginkább dokumentumbesorolási problémára használnak, például a dokumentum beletartozik-e adott kategóriába. Az előre jelzők, jellemzők a dokumentumban előforduló szavak gyakorisága. Másik a Bernoulli naiv Bayes, ami hasonló az előzőhöz, de a prediktorok logikai változók, az osztályváltozó előrejelzésére használt paraméterek pedig igen vagy nem értéket vesznek fel. Ezeket követi a Gauss naiv Bayes. Ha a prediktorok folytonos értéket vesznek fel, feltételezük, hogy a vett értékek a Gauss-eloszlásból vannak mintavételezve. Ez esetben viszont a fent említett képlet már nem helytálló [11].

2.8.3 SVM

SVM, más néven support vector machines („támogató vektorgépek”) felügyelt tanulási algoritmus, ami osztályozási és regressziós problémákra használható, kisebb adatkészletre, mert sokáig tart a feldolgozása. Az n dimenziós térben minden pont adatelemként jelenik meg, és az egyes jellemzők értéke az egy adott koordináta értéke. Hiper-sík segítségével különböztet meg két osztályt, több módszerrel. Robosztus, ezért nem probléma, ha a bemeneti adat nem választható szét, vagy nem lineáris. Előnye még, hogy memóriahatékony is. [12] [13].

2.8.4 Decision Tree

A döntési fa egy folyamatábra-szerű struktúra, amelyben minden belső csomópont egy jellemző tesztjét reprezentálja (például érmedobás esetén fej vagy írást kapunk), minden levélcsomópont egy osztálycímkét (ami az összes jellemző kiszámítása után hozott döntés) és az elágazások. olyan jellemzők konjunkcióit jelentik, amelyek azokhoz az osztálycímkékhez vezetnek. A gyökértől a levélig tartó utak klasszifikációs szabályokat jelentenek.

Előnye, hogy könnyű megérteni és használni, kategoriális és numerikus adatot is tud kezelni, outlierekre (kiugró értékekre) rezisztens, azonban szükség van valamilyen mérésre, hogy hogy teljesít, és paraméterek állításánál óvatosnak kell lenni. Fontos még, hogy torzított fákat tud létrehozni, ha néhány osztályból túlsúly van [14].

2.8.5 Random Forest

A „véletlenszerű erdő", ahogy a neve is sugallja, nagyszámú egyedi döntési fából (decision tree) áll, amelyek együttesként működnek. A random forest-ben minden egyes fa kiad egy osztály predikciót, és a legtöbb szavazatot kapott osztály lesz a modellünk előrejelzése. Úgy lehet elképzelni, mint egy bizottság, amiben nem korrelált modellek szavaznak, azaz az egyes fák hibáit a többi ki tudja javítani (abban az esetben, ha a többség nem hibázik rendszeresen). Fontos azonban, hogy a jellemzőinkben kell lennie valamilyen jelnek úgy, hogy az ezekből épült modellek ne véletlenszerűen találgassanak, valamint az egyes fák általi előrejelzéseknek (beleértve a hibákat) alacsony korrelációban kell lenniük egymással [15].

2.9 IDS értékelés

A behatolás észlelő rendszereket általánosan véve a következő teljesítmények alapján értékelik:

Valódi pozitív arány(TPR – True Positive Rate): a valódi előre jelzett, és az összes támadás aránya. Ha mindent jelzett a rendszer, akkor 1-es értéket kapnánk. Képlete [1]:

$$TPR = \frac{TP}{FN}$$

Álpozitív arány(FPR – False Positive Rate): a hibásan támadásnak minősített normális példányok, és az összes támadás aránya. Képlete [1]:

$$FPR = \frac{FP}{FP + TN}$$

Álnegatív arány(FNR – False Negative Rate): amikor a rendszer nem észleli a támadást, és normálisnak minősíti az anomáliát. Képlete [1]:

$$FNR = \frac{FN}{FN + TP}$$

Érzékenység (Recall) arány: valódi pozitív értékek elosztva a valódi pozitív és álnegatív értékek összegével. Minél közelebb van egyhez, annál jobb. Képlete [16]:

$$Recall = \frac{TP}{TP + FN}$$

Precízió (Precision): a valódi pozitív értéket elosztjuk a valódi pozitív és álpozitív összegével. Minél közelebb van 1-hez, annál jobb a modellünk. Képlete [16]:

$$Precision = \frac{TP}{TP + FP}$$

F1 érték (F1 score): precíziót megszorozzuk a recall-al, majd elosztjuk a kettő összegével, majd ezt az értéket megszorozzuk kettővel. Képlete [16]:

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Osztályozási arány (CR – Classification Rate/Accuracy): Az IDS pontosságát méri, mennyire tudja detektálni a normális vagy rendellenes forgalmi viselkedést. Értékével megtudjuk, hogy az összes valódi eredményhez képest, hány százalékos arányban jósolták meg helyesen az összes példányt. Képlete [1]:

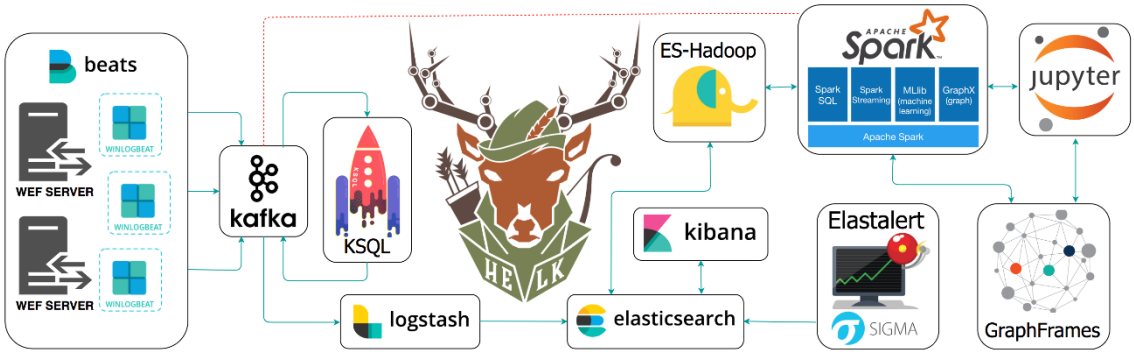
$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Vevő működési jellemzői (Receiver Operating Characteristic - ROC) görbe: egy kétdimenziós koordináta rendszerben az x tengelyen az álpozitív arány, az y tengelyen a valódi pozitív arány látható. A ROC görbén a valódi pozitív arányt az álpozitív arány függvényében ábrázoljuk a különböző határértékekre. A görbén található minden pont egy valódi és egy álpozitív aránypárt képvisel, melyek egy bizonyos döntési küszöbnek felelnek meg. Az osztályozás küszöbértéke változó, ezért a ROC egy másik pontját választják ki, egy különböző hamis jelző rátával és különböző valódi pozitív aránnyal. Egy tökéletes teszten, ahol nincs átfedés a két eloszlásban, a ROC görbe átmegy a bal felső sarokban, ami 100%ban fedi a pozitív és negatív értékeket [1].

Görbe alatti terület (AOC Area Under the Curve) bináris klasszifikációs feladatoknál használják, ROC görbe összefoglalója. Segítségével megtudjuk, mennyire tudja a modellünk a jó (normál) és rossz (támadás) adatokat megkülönböztetni. Minél közelebb van az értéke 1-hez, annál jobban, minél közelebb nullához, annál kevésbé [17].

2.10 HELK Rendszer

A Hunting ELK, vagy máshogy egyszerűsítve HELK (Hunting ElasticSearch, Logstash and Kibana) egy olyan nyílt forráskódú vadászplatform, mely olyan fejlett elemzési képességekkel rendelkezik, mint például az SQL deklaratív nyelve, grafikonok készítése, struktúrált streaming, illetve gépi tanulási lehetőségek az Apache Spark, és Jupiter notebookok segítségével, egy stack (verem) felett. Ez a projekt elsősorban kutatási célra lett kifejlesztve, de sok és rugalmas alapkomponensei miatt nagyobb környezetben is alkalmazható, megfelelő konfigurációval és infrastruktúrával méretezhető és telepíthető [18]. A HELK rendszer ábrázolása és alrendszereinek kapcsolata a 3. ábrán látható.



3. ábra– HELK rendszer alrendszereivel együtt [18]

2.10.1 ELK Rendszer

A rendszer fő alappillére nevéből adódóan 3 komponensből tevődik össze, melyek:

Elasticsearch

Az ElasticSearch egy nyílt forráskódú kereső és elemző motor mindenféle adatra kiterjedve, beleértve a szöveges, numerikus, strukturált és strukturálatlan adatot. Egyszerű REST API (Representational state transfer application programming interface)-jéről ismert, az Elastic stack fő komponense, segítségével adatokat tudunk elemezni, tárolni, vizualizálni. Felhasználható loggolásra és log- és biztonsági elemzésre [18].

A különböző funkciók elvégzésére memóriát / RAM-ot használ, pontosabban heap-et, mely a memóriának az a része, melyből a blokkok dinamikusan foglalhatók le, és szabadíthatók fel. Alkalmazása olyan programoknál esedékes, ahol az adathalmazok tárolására nincs szükség a teljes futási idő alatt, illetve melyek méretéről nincs elegendő információ indítás előtt, vagy futás közben is változik [19]. A heap feladatai közé tartozik az indexek nyomon követése, adatokon összesítő matematikai műveletek elvégzése, például összeadás, összesítések részösszegeinek kiszámítása, adott keresések végrehajtása, indexelt értékek eltolásának (offset) nyomon követése. A heap mennyiségének meghatározása a tervezett feladatoktól függ, beállítása lehet manuális és automatikus [18].

Logstash

A Logstash ingyenes szerveroldali adatfeldolgozó „futószalag” (pipeline) , ami több forrásból nyert adatokat alakít át, és küldi el az adott személyes „tárhelyre” (stash). Segítségével egységesíteni tudjunk a kívánt adatot, illetve módosítani tudunk a logformátumon [20] [21].

Kibana

A Kibana egy nyílt forráskódú frontend alkalmazás, az Elasticsearchben indexelt adatokon keresést és megjelenítést biztosít. Az Elastic stack tetején helyezkedik el,

közismertebb nevén az elastic stack diagramkészítője, felhasználói felületet biztosít az ELK klaszter monitorozásra, kezelésére, és biztonságossá tételére, valamint beépített megoldások központi hub-ja [20].

2.10.2 További HELK alrendszerek

Beats

A Beats nyílt forráskódú adatszállítók összessége, amik adatokat szállítanak a HELK Elasticsearch moduljának. Több úgynevezett „Beat” -je van, melyek telepítésével a hozzá kapcsolódó típusú adatfajtát tudja rögzíteni, mint például logfileok, elérhetőség, felhőadatok és hálózati forgalom [20].

Kafka

A Kafka rendszer egy esemény streamelő szolgáltatás, ami azt jelenti, hogy valós időben tud adatot gyűjteni olyan eseményforrásokból, mint felhőszolgáltatások, adatbázisok, szenzorok, mobil eszközök. Ezeket az esemény streameket későbbi felhasználás céljából tartósan eltárolja, feldolgozza, egyéb esetben valós időben és visszamenőlegesen is reagálni tud rájuk [22].

KSQL

A KSQL a Kafkához tartozó streamelő SQL motor. Folyamatos lekérdezéseket futtat, de nem a hagyományos SQL értelemben, hanem a streameket dolgozza fel. Ennek segítségével valós idejű monitorozásra képes valós idejű analitika mellett, ami lehetővé teszi az anomália detektálást [23].

ES-Hadoop

A Hadoop egy batch feldolgozó rendszer, amivel valós eredményeket kimutatni nehézkes. Az ES-Hadoop az Elasticsearch előnyeivel is rendelkezik, a Hadoop adatot indexelni lehet az Elastic stack-be, így könnyen tudunk keresni és utána Kibanával megjeleníteni [20].

Elastalert

Az Elastalert egy egyszerű keretrendszer, ami az Elasticsearchből vett adatok alapján riaszt, ha anomáliát, kiugrást, vagy számunkra érdekes mintát talál. Az Elasticsearch-et két fajta komponenssel, a szabály típusokkal és a figyelmeztetésekkel ötvözi. Az adatokra futtatott periodikus lekérdezés eredménye egy szabálytípushoz kerül, ami megvizsgálja, hogy van-e egyezés. Amennyiben igen, átkerül egy vagy több figyelmeztetőnek, ami az egyezés alapján határozza meg a következő cselekvést [24].

Jupiter Notebook

A Jupiter notebookok interaktív számítási környezetet biztosítanak a Python alapú Data Science alkalmazások fejlesztéséhez, korábban ipython notebook néven ismerték. Egy nyílt forráskódú webalkalmazás, amely segítségével például kódot, egyenleteket, és megjelenítést tartalmazó dokumentumokat tudunk létrehozni, és megosztani. Több területen lehet alkalmazni, többek között adatok tisztításánál és átalakításánál, statisztikai modellezésnél, adat vizualizációnál, gépi tanulásnál [25]. Előnyei:

- lépésről lépésre szemléltetheti az elemzési folyamatot azáltal, hogy lépésről lépésre rendezi például a kódot, képeket, szöveget, kimenetet stb.
- A data science-el („adat tudós”) foglalkozóknak segít dokumentálni a gondolkodás folyamatát, miközben az elemzési folyamatot végzi
- Az eredmény rögzíthető a notebook részeként is.
- A munkánkat másokkal is meg tudjuk osztani [25] [26]

Egyik ezen használható interaktív web alapú fejlesztői környezet a Jupyterlab, mely rugalmas, konfigurálható felhasználói felülettel rendelkezik, az adott munkafolyamattól függően testre szabható. Bővíthető és moduláris – olyan bővítmények készíthetők, amelyek új komponenseket adhatnak hozzá, és integrálódnak a már meglévőkkel [26].

Apache Spark

Az apache spark egy egységes elemző motor nagy mennyiségű adatfeldolgozáshoz. Magas szintű API-kat (Application Programming Interface) biztosít Java, Scala, Python, és R nyelveken, illetve egy elosztott általános célú motorja van. Támogatja még a Spark SQL-t, amit struktúrált adatfeldolgozáshpz, Mlib könyvtárat a gépi tanuláshoz, míg GrapY-et a gráf feldolgozáshoz lehet használni [27].

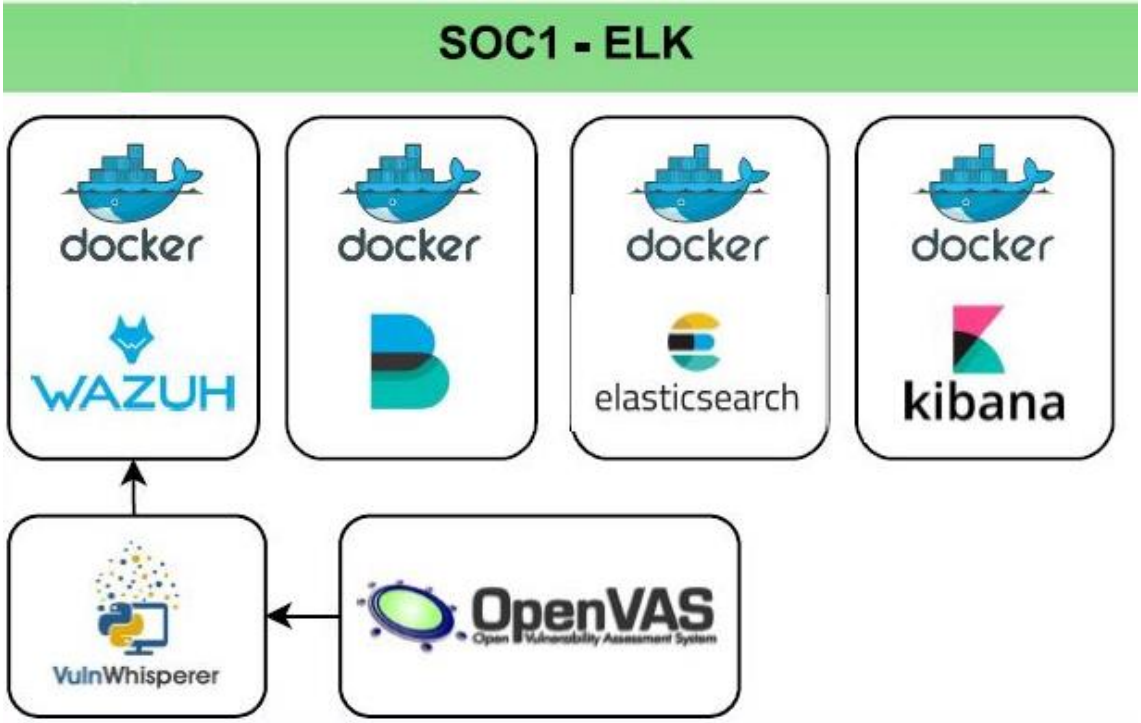
GraphFrames

A GraphFrames egy Apache Sparkhoz tartozó csomag, ami data frame (adatkeret) alapú gráfokat biztosít. A Spark data frame lehetőségeit bővíti, segítségével elérhetőek a gráf lekérdezések. Magas szintű API-t biztosít Java, Scala és Python nyelveken [20].

3 Tervezés

3.1 Megvalósítandó feladat

Az egyetem hálózatában volt található egy 4 gépből álló SOC (Security Operations Center – Biztonsági központ), az 1. gépre fel volt telepítve a HELK bizonyos részei. Az erre telepíthető a Jupyter notebook segítségével el lehet készíteni a betanult és elkészített IDS (Intrusion Detection System)-ünket. Az adott SOC 1-es gépen található HELK alapmoduljai az alábbi ábrán láthatóak:



4. ábra – ELK felépítése az egyetemi hálózatban

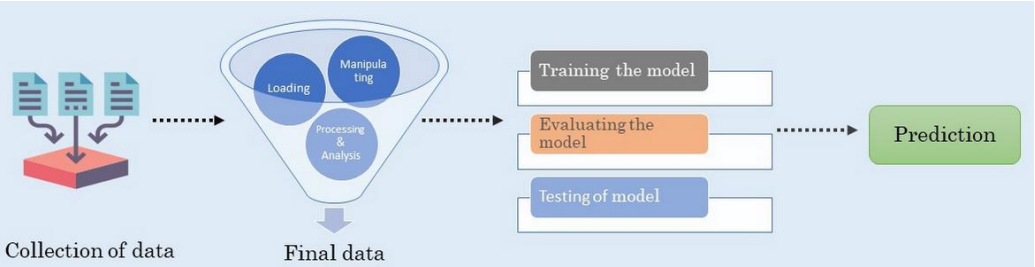
Távoli eléréssel az openVPN-en keresztül tudtuk elérni, és egy másik, SOC2-es elnevezésű gépre VNC-vel rá tudtunk csatlakozni.

A beat HELK almodullal be lehet állítani milyen típusú hálózati adatforgalmat szűrjünk, azt tovább lehet küldeni a logstash-nek, ami egységesíti az adatot, amelyben végül kereséseket tudunk végrehajtani, amennyiben szükséges. Kibanával különböző statisztikákat ábrázolhatunk, elastaltert modullal (jelenlegi ábrán nem látható) riasztásokat küldhetünk, amennyiben a rendszerünk anomáliát / hálózati fenyegetést / behatolást észlelt.

3.2 Gépi tanulás felépítése

Egy gépi tanulás az alábbi folyamatokból áll, ahogyan az alábbi ábrán grafikusan látható:

- adatok begyűjtése
- adatok manipulálása, tisztítása
- modell betanítása
- modell értékelése
- modell tesztelése
- predikció



5. ábra – Gépi tanulás folyamatai [28]

3.3 Adatok begyűjtése

Adatok begyűjtése történhet az UNB oldalról. Itt találhatóak a Kanadai Kiberbiztonsági Intézet adathalmazai, amelyeket világszerte használnak egyetemek és kutatók. Az adathalmazok ingyenesen letölthetőek néhány adatunk megadása után, miszerint mire tervezzük használni majd a letöltött fájl(okat). Évek szerint vannak felsorolva, az adott éveken belül több nap több típusú támadást generáltak és rögzítettek, átlagos felhasználói magatartással együtt. Ennek segítségével az adathalmazon belül meg lehet jelölni mi számít támadásnak és mi nem, ami le egyszerűsítheti a behatolás észlelő rendszerek értékelését, és realisztikusabb referenciaértékeket nyújt [29].

Az egyik populáris, UNB ISCX 2012 behatolásdetektáló kiértékelő adathalmaz a következő tulajdonságokkal rendelkezik:

- valósnak tűnő hálózat és azon keresztül történő forgalom a valós helyzet előállítása érdekében
- címkézett, jelölt adathalmaz, ezzel egyértelműen elkülönítjük az anomáliákat a normál forgalomtól, így nem kell saját magunknak felcímkézni
- teljes interakció rögzítése, hogy minél több információnk legyen a behatolásról
- hálózati forgalom eredeti állapotának megtartása, az anonimizálás és egyéb módosítások elhagyása a legnagyobb használhatóság érdekében

- változatos behatolási esetek, beleértve a méret, típus és gyakoriságot, hogy a modellünk minél több információval tudjon tanulni
- pcap formátumban elérhető fájlok több napra lebontva [29]

Újabb adathalmazok más támadásokat tartalmazhatnak, és egyeseknek dokumentációja részletes lehet. Az adatok konverziójára használhatunk egy úgynevezett CICFlowMeter programot, ami a pcap fájlok átalakítására is szolgálhat [29].

Másik populáris adathalmaz használati lehetőség a KDD Cup 1999 adat, amelyet a Harmadik Nemzetközi Tudásfelfedező és Adatbányászati Eszközök Versenyén használtak, ahol egy behatolás észlelő rendszert kellett a résztvevőknek fejleszteniük. Kilenc hétnyi nyers TCP-kiíratási adatot gyűjtöttek egy helyi hálózatban (LAN), amely egy tipikus amerikai légierő LAN-ját szimulálja. Úgy üzemeltették a helyi hálózatot, mintha valódi légierő-környezet lenne, de többszörös támadással fűszerezték [30].

A nyers betanítási adatok körülbelül négy gigabájt tömörített bináris TCP-kiíratási adatok voltak 7 hét hálózati forgalomból. Ezt mintegy ötmillió kapcsolati rekordra dolgozták fel. Hasonlóképpen, a kéthetes tesztadatok körülbelül kétmillió csatlakozási rekordot eredményeztek. A kapcsolat TCP-csomagok sorozata, amelyek bizonyos jól meghatározott időpontokban kezdődnek és végződnek, és amelyek között az adatok egy forrás IP-címre és egy cél IP-címre áramlanak valamilyen jól definiált protokoll szerint. Minden kapcsolat normál vagy támadásként van megjelölve, pontosan egy adott támadástípussal. Minden kapcsolati rekord körülbelül 100 bájtból áll. Többféle támadás található benne, beleértve a DoS (Denial of Service - szolgáltatásmegtagadás), R2L (Remote to Local – „Nem engedélyezett” távoli támadás), U2R (User to Root – rendszergazdai jogok megszerzése), Probing (Eszköz felderítése hibák kiaknázhatósága szempontjából). Legmagasabb százalékban a DDoS (elosztott szolgáltatásmegtagadás) - t tartalmazza az úgynevezett „neptune” formában. Az adathalmaznak több változata is van, gyakori még annak a 10%-os változatát használni gyorsabb feldolgozás érdekében [30].

3.4 Adatok manipulálása

A letöltött adatokat egységesíteni kell, és meg kell tisztítani, kiszedni belőlük a számunkra fölösleges információt, hogy a modell tanítása gyorsabban történjen, és ne tartalmazzon számunkra haszontalan információt. Ehhez az egyik lehetőség a pandas használata, mely egy gyors, rugalmas, hatékony, és könnyen használható nyílt forráskódú adatelemzési és manipulációs eszköz, pythonra épülve [31].

3.5 Modell betanítása

Egy gépi tanulás megvalósításához több keretrendszer áll rendelkezésünkre, attól függően, hogy mélytanulást akarunk-e alkalmazni, vagy egyéb gépi tanulási algoritmusokat:

3.5.1 PyTroch

A PyTorch egy viszonylag új mélytanulási keretrendszer, mely a torch-on alapszik. A Facebook AI kutatócsoportja fejlesztette ki, és nyílt forráskódként elérhetővé tették a GitHubon 2017-ben. Természetes nyelv feldolgozó alkalmazásokban is használják, flexibilis, hatékony a memóriakezelése, manapság sok mélytanulási projektnél használják, egyre népszerűbbé válik [32].

3.5.2 Keras

A Keras egy Pythonban íródott hatékony, magas szintű neurális hálózati API. Ezt a nyílt forráskódú neurális hálózati könyvtárat úgy tervezték, hogy gyors kísérletezésre legyen alkalmas mély neurális hálókkal. Tensorflow-n és Theanon is futtatható. 2017 közepén beintegrálták Tensorflowba, viszont külön is képes működni, így tulajdonképpen a Kerast egy csomagolóként (wrapper) használják a Tensorflow keretrendszeréhez. A Kerast könnyebb használni, azonban, ha speciális Tensorflow funkcionalitásra van szükségünk, akkor a Keras modellben azt is tudjuk alkalmazni. Gépi tanulási applikációk számára érdemesebb Tensorflowt, és mély neurális hálók esetében érdemesebb Kerast használni [32].

3.5.3 Theano

Gyors számítási képességekkel rendelkezik, mély neurális hálózati algoritmusok képzésére specializálódott. Cross-platform (több rendszeren használható), illetve a processzor (CPU) és GPU (grafikus processzor) -on futtatható, utóbbi használata esetén ez gyorsítja a tanulást [32].

3.5.4 Scicit-learn

A scicit-learn az egyik leghasznosabb és legrobosztusabb könyvtár gépi tanuláshoz Python nyelven. Hatékony eszközöket kínál a gépi tanuláshoz és statisztikai modellezéshez beleértve az osztályozást, regressziót, klaszterezést, előfeldolgozást és dimenziócsökkentést. Leginkább Pythonban íródott könyvtár, Numpy, SciPy és Matplotlib alapú.

3.5.5 FastAI

Egy olyan mélytanulási könyvtár, amely gyorsan és hatékonyan képes az eredmények elérésére a standard mélytanulási területen, alacsony szintű komponensei összeillesztésével új megközelítéseket lehet vele megvalósítani [33].

3.5.6 Tensorflow

A Tensorflow egy végpontok közötti, nyílt forráskódú mélytanulási keretrendszer, amit a Google fejlesztett ki 2015-ben. Androidon is használható, neurális hálózatok által használt szimbolikus matematikai könyvtár, leginkább adatfolyamok programozására alkalmas több feladaton keresztül. Gyorsan növekvő keretrendszer közösségi erőforrásokkal, könyvtárakkal és eszközökkel, melyekkel gépi tanulási alkalmazások felépítését és telepítését tudjuk végrehajtani. Az alábbi ábra egy összehasonlító táblázat a használható keretrendszerek / könyvtárak között, funkcióik tekintetében [32]:

	Keras	**Pytorch**	**TensorFlow**	**Scicit-learn**
API Szint	Magas	Alacsony	Magas és Alacsony	Magas
Architektúra	Egyszerű, olvasható	Komplex, kevésbé olvasható	Használata nem egyszerű	Olvasható
Adathalmazok	Kisebb adathalmazok	Nagy adathalmazok, nagy teljesítmény	Nagy adathalmazok, nagy teljesítmény	Nagy teljesítmény
Hibakeresés (Debuggolás)	Egyszerű hálózat, hibakeresés általában nem szükséges	Jó hibakeresési lehetőségek	Hibakeresés nehézkes	Normál hibakeresés
Tanult modelleket tartalmaz?	Igen	Igen	Igen	Igen
Népszerűség	Legnépszerűbb	Felfele ívelő	Népszerű	Népszerű
Sebesség	Lassú, alacsony teljesítmény	Gyors, magas teljesítmény	Gyors, magas teljesítmény	Gyors, magas teljesítmény
Milyen programnyelven íródott	Python	Lua	C++, CUDA, Python	Python

6. ábra – Környezetek összehasonlítása [32]

3.6 Modell értékelése és tesztelése

Modell értékelés és tesztelés alatt azt értjük, mennyire képes hatékonyan működni betanulás után, ennek értékét a 2.8. fejezetben leírt ROC görbével tudjuk vizualizálni.

3.7 Predikció

Ezalatt a modell kimenetét értjük, az eredményt, amelyet a modell kiad. IDS-ek esetén ez anomália, vagy normál adatforgalom lehet.

3.8 Előző munkák

Több megoldás is született behatolás észlelő rendszerekre, néhány gépi tanulási algoritmusokkal rendelkezőt az alábbiakban vizsgálok meg.

3.8.1 Shellcode szűrés

Egyik esetben az ANN-el próbálták megoldani a shellcode felismerést. Shellcodenak nevezzük azt a kódrészletet, amit hasznos teherként használnak támadáskor, így megnyitható egy shell, amibe kártékony kódot tudnak futtatni [34]. A problémafelvetésben szerepelt, hogy kis méretű, alacsony szintű kód, gyakran el van rejtve támadásoknál. Szignatúrás módszerek esetében tovább nehezíti a feladatot, hogy a nem kártékony kód bináris mintái is hasonlíthatnak támadásra, nehezen lehet megkülönböztetni, például Windows frissítéskor DLL-ek letöltése sok álpozitív eredményt hozott [35].

ANN tervezésnél a bájt szintű adatot az adathalmazból átkonvertálták integer értékekké, amit betápláltak egy mesterséges neurális hálózatba. A fájlok elején található „varázsszámokat” kikerülték, mivel az osztályozó ezt könnyen azonosíthatta volna, és könnyen hamisíthatók, kimondottan akkor, ha elrejtett héjkódot terveznek. 1000 bájtnyi összefüggő adatot adtak az ANN bemenetére, és a kezdeti eredmények megkülönböztethető mintákat mutattak különböző fájltípusok között, bár az azonos fájlok között is voltak jelentős eltérések [35].

A neurális hálózatot Matlabban implementálták, a hálózat megfelelő struktúráját rácskeresési folyamat során találták meg, ami végül egy több rétegű perceptront jelentett, kettő rejtett réteggel, ami egyenként 30 rejtett neuront tartalmazott. A hálózat szerkezetének optimalizálásához ismételt tízszeres keresztellenőrzést használtak, hogy értékeljék az osztályozó kialakítását, és visszaterjesztéses módszerrel tanították a hálózatot. Az adatkészlet jó és rossz fájlokat is tartalmazott, és a legjobban tanított mesterséges neurális hálózatnak sikerült kiszűrnie az összes rosszindulatú kódot, álpozitívok nélkül. Ezt utána 400.000 véletlenszerű fájlon is kipróbálták, hogy magasabb legyen a bemeneti elemszám, ebben az esetben 1.8% -nál jelzett tévesen [35].

A fenti megoldás az adatokat megszerezve offline módon működött, így ugyan ki tudta szűrni a kártékony kódot, de nem valós időben, és nem a hálózaton, így éles rendszeren nem tudna megfelelő védelmet nyújtani.

3.9 Random Forest modellezés behatolásészlelő rendszeren

Volt, aki az úgynevezett random forest algoritmussal készített egy behatolásészlelő rendszert. KDD 1999 adathalmazt használva az alábbi támadások észlelését tesztelték: DoS (Denial of Service - szolgáltatásmegtagadás), R2L (Remote to Local – „Nem engedélyezett” távoli támadás), U2R (User to Root – rendszergazdai jogok

megszerzése), Probing (Eszköz felderítése hibák kiaknázhatósága szempontjából). A munkát összehasonlították FSS – szimmetrikus bizonytalanság, J48 döntési fa, és a „véletlenszerű erdő” (Random Forest) algoritmusokkal. Az eredmények alapján a random forest-el mind a négy támadásra 99.67% pontosságot(accuracy)-t értek el, míg ez J48 esetén 99.24%-99.29% között mozgott, szimmetrikus bizonytalanságnál 99.63%-99-68% között [36].

3.9.1 Mély neurális hálókon alapuló IDS

Egy másik megközelítésnél két támadás ellen védekeztek neurális hálókkal, a DDoS, és a Malware ellen. Itt két részre osztották a feladatot, első volt a DDoS detektálás. A DDoS (Distributes Denial of Service) egy olyan elosztott szolgáltatás megtagadásos támadás, ahol több különböző rendszerről támadnak Dos támadással, ami egy szerver elárasztása TCP és UDP csomagokkal [37]. A modelljük 3 modulból áll:

Nyers adatok kinyerése, ami az adatok adatbázisba mentését jelenti, amit a következő modul tud felhasználni. Csomagokat, méretüket, és küldött/fogadott idejüket tartalmazza. Ezt követte a feature vector ábrázolás, ami az adatbázisból kiszedett adatokat összegyűjti adott időablakokban, és kiszámítja az ablakok közötti csomagok és bájtok mennyiségét. Az ablak mérete paraméterezhető. Minden ilyen ablakból 7 statisztikai adat kerül kinyerésre, és ezeket a vektorokat átadja a DNN (Deep Neural Network)-nek. Végül a klasszifikáció mély neurális hálók segítségével, amely a feature vektorokat dolgozza fel, 5 rétegből áll, bináris eredménnyel, azaz vagy támadást jelez, vagy normális forgalmat [38].

A másik támadás a malware ellen volt, így ennek a detektálása volt a feladat. Mint az előző példában, ugyan úgy 3 modult használ. Az egyik ugyanúgy a nyers adat kinyerés, ami a rendszerhívásokat (system calls) monitorozza, beleértve a monitorozó részeket, és fájlrendszerrel kapcsolatosokat. Ezeket sorozattá alakítja, minden applikációhoz a sajátját rendeli, és átadja a következő modulnak. Ezt követte a Feautre vektor ábrázolás, ami minden applikációhoz tartozó 4 egymás utáni system call-t kódol egy egyedi helyre a vektorban, ezeket átadja a következő rétegnek. Majd végül a Neurális háló klasszifikáció, ami egy 5 rétegből álló háló, bináris kimenettel rendelkezett [38].

Ezzel a megvalósítással DDoS támadásnál 99.79%-os pontosságot, Malwarenél 99.44%-os pontosságot tudtak elérni. Ez megfelelő pontosságúnak bizonyul, itt viszont csak 2 támadási lehetőséget vesznek figyelembe [38].

3.9.2 Time Delay Neural Network-ön alapuló IDS

A Time Delay Neural Network (TDNN) felhasználásával készült IDS-ben a host sweep, és a port scan támadás felismerését vizsgálták meg. Előbbi támadásnál a hálózaton történő számítógépeket próbálják felderíteni, melyekre később támadást tudnak indítani, utóbbi esetében egy adott számítógépen futó szolgáltatások felmérése történik. Port scan

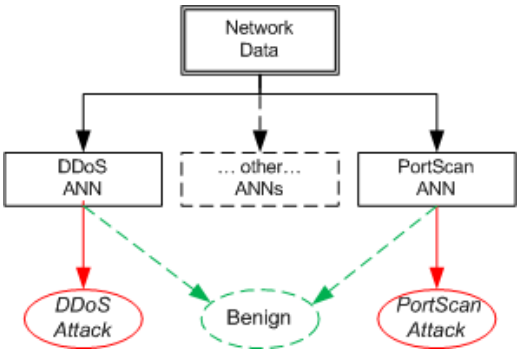
esetén amennyiben megtudjuk milyen portokat használ az adott eszköz, abból következtethetünk milyen hálózati alkalmazásokat futtat, melyekre célzottan lehet támadást indítani. A vizsgált IDS felépítésénél a packet capture (csomagszerzés) után tovább küldték őket az előfeldolgozónak, aminek két alfeldolgozója volt, célzottan az adott támadásra. Ezután a mintafelismerő modulnak továbbították, amelynek szintén két almodulja volt, ezt követte a klasszifikációs rész, végül a riasztási modullal fejeződött be. A host sweep előfeldolgozója 7 elemű neurális hálóból állt, míg a port scan változat 6 elemből. A klasszifikációs modul ugyanígy 4 és 5 elemből, míg a klasszifikációs rész kettőből. A rendszert SNORT [39] ellen tesztelték, és NMAP [40]-el generálták a forgalmat. Az eredmények alapján látható, hogy a két típusú támadás összes változatát felismerte az IDS, viszont a SNORT nem. Habár megjegyzendő, hogy a SNORT alkalmazás nem csak erre a kettő célzott támadásra szűr [41].

3.9.3 Rosszindulatú programok azonosítása dinamikus viselkedéssel

Ennél a tanulmánynál rámutattak arra, hogy a hagyományos észlelések nem elég hatékonyak a malware-ek új variánsaira. A készítők olyan technikákat használnak, amik több végrehajtási lehetőséget alkalmaznak, ami az eredményeknél rosszindulatú viselkedésnek minősül Ezt Malicious Behaviors and Outcomes (rosszindulatú magatartások és kimenetelek - MBO)-nak nevezik. Ezeket a kimeneteket modellezték, és egy osztályozót betanítottak, hogy ismeretlen malware variánsokat is felismerjen. Jó eredményt értek el új malware variánsok ellen, alacsony álpozitív aránnyal [42].

3.9.4 Neurális hálókon alapuló IDS

Ez a tanulmány is kiemeli a tanuláson alapuló IDS-ek fontosságát. A rendszerük még fejlesztés alatt áll, de a jelenlegi megoldásukban PortScan és DDoS támadás ellen tudnak védekezni, két neurális hálóval. 99.8%-os pontosságú felismerési aránnyal megfelelőek voltak az eredményeik. Korlátozást jelent azonban, hogy ez az IDS is jelenleg csak 2 támadás ellen tud védekezni, és a különböző támadások megjelenésével új ANN-t kell létrehozni, és implementálni a rendszerbe, valamint nincs a hálózatban védekezés, ha egy adott támadás variánsával kell szembenézni, vagy ha több támadás kombinációjával támadják a rendszert. Mint ahogy a 4. ábrán látható, a rendszer két támadását még tovább lehet fejleszteni több ANN-el [43].



7. ábra – Továbbfejlesztési lehetőségek több támadásra [43]

3.9.5 Automatizált támadások keresése ELK használatával

Ebben a projektben a behatolásészlelést két oldalról közelítették meg, egyik valamilyen intelligens vadász szoftver használata, másik a végpont monitorozása Sysmon, Windowsra telepíthető programmal. Utóbbinál a szokásos loggolást a rendszer kiszűri, és csak utána küldi el az ELK stacket futtató szervernek, mely elemzi a szülő folyamatokat és viselkedésüket, és a szerveroldalon is szűrőt alkalmaznak az analizáláshoz és fenyegetés kereséshez. 3 támadási fajtát vizsgáltak meg [44]:

- Ártó kód használata fájlok távoli eléréséhez megosztott meghajtón, ezt követően a fájlok törlése
- Távoli elérés a regisztrációs bejegyzések törléséhez
- Ártó kód használata magasabb szintű jogosultságadáshoz

A rendszert virtuális gépen tesztelték, és látható volt, hogy sikeresen azonosította mindhárom támadást. Ennek korlátozása viszont, hogy a rendszerhez használt Sysmon program Windowson használható, más rendszereken nem, illetve el kell küldenie az adatokat a HELK rendszernek. A küldemény csatorna támadásoknak lehet alávetve, ami megakadályozhatja a szerver és kliens közötti kapcsolatot, így nem tudja jelezni az észlelő rendszer, ha fenyegetés van jelen [44].

3.9.6 Multilayer perceptronon alapuló modell

Egyik bevezető példa a neurális hálókra a többrétegű perceptron modell. Ezt egyszerű felépítés jellemzi, segítségével megoldhatunk olyan problémákat, amiket nem lehet egy sima perceptronnal (például XOR művelet), valamint sok más nem-lineáris függvénnel. Az MLP (Multilayer Perceptron) egy mély, mesterséges neurális háló. Van egy bemeneti rétege ami a jel fogadására szolgál, egy kimeneti rétege ami a bemenettől függően ad egy döntést vagy előrejelzést, valamint a kettő között tetszőleges számú rejtett rétegek, amik a tényleges számolást végzik a modellben. Az egy rejtett réteggel rendelkezőkkel képesek vagyunk megbecsülni bármilyen folyamatos függvényt [45].

Felügyelt tanulási problémák megoldására használják, a tanulás magába foglalja a paraméterek, súlyok és torzítások (bias) konfigurálását, amelyek segítségével minimalizálhatjuk a hibát. Ennek példájára készült egy behatolásészlelő rendszer ReLu nevezetű nemlineáris aktivációs függvényt használ, modell pontossága 85%, így jelen formában ennél hatékonyabb megoldások is vannak, viszont felépítésének egyszerüsége könnyen átláthatóvá teszi [46].

3.9.7 Egyéb megvalósítások

Github oldalon több megoldás található IDS implementációra, több keretrendszer és könyvtár használatával. Egyik megoldásban pont azt figyelték meg, hogy FastAI, Keras-TensorFlow és Keras-Theano könyvtárak használatával betanított modelleknek mekkora a pontossága a 2018-as, IDS betanítására szánt adathalmazból. Ebből az évből több napot külön figyelembe véve az eredmények alapján azt látni, hogy a FastAI segítségével jutottak a legpontosabb eredményre [47].

Másik esetében a modell különbségeket vizsgálták meg, ott autoencodert, random forest (véletlenszerű erdő), logisztikai regressziós és ID3 döntési fa modellt alkalmaztak. Az eredményeiket összevetve az ID3 döntési fa bizonyult a legjobb választásnak, 99,74% pontossággal [48].

Vannak akik egyszerűbb megközelítéssel dolgoztak, és egy többrétegű perceptron modellel készítettek egy behatolásészlelő rendszert. Ezt pytorch-al oldották meg, leírás alapján 85%-os pontosságot lehetett vele elérni. Erre a munkára alapoztam ebben a félévben [49].

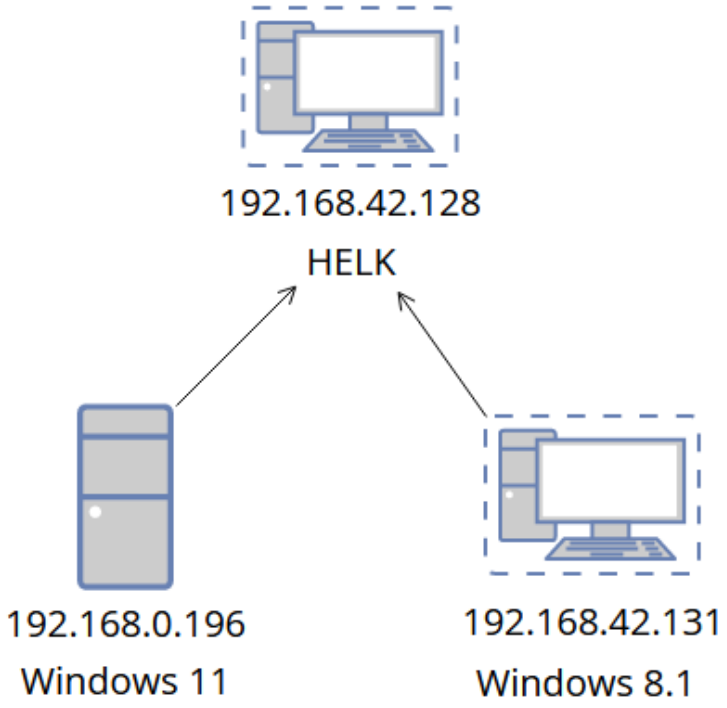
Mások igénybe vették az NVIDIA cég által elérhető CUDA alkalmazást és könyvtárakat, amelyek segítségével GPU számításokat is végezhettek, így gyorsabb volt a megvalósítás, mint hagyományosan, CPU használatával. Az IDS elkészítéséhez ezen kívül még a Tensorflow-GPU, és a Keras könyvtárakat is használták [50].

4 Megvalósítás I. – HELK oldal

Az alkalmazás kezdeti megvalósításához jelenlegi infrastruktúra a 8.ábrán látható.

A rendszerek telepítéséhez szükséges megfelelő hardver is, a HELK több szolgáltatás telepítéséhez minimum 8 GB RAM-ot ajánl, valamint 20 GB tárhelyet, de éles használatra 100 GB-ot. Esetemben 8 CPU szálat adtam neki 20 GB RAM-mal, és 120 GB tárhellyel, így megfelelően futottak a szolgáltatások.

Szoftveresen a hivatalos dokumentáció által javasolt Ubuntu 18.04-re telepítettem, és az alábbi infrastruktúrát használtam:



8. ábra – Infrastruktúra

Windows 11 gépen települt fel virtuális gépre a HELK, valamint ugyanúgy virtuális gépre külön települt fel a Windows 8.1, adatok szállításához a WinlogBeat és PacketBeat, valamint Sysmon alkalmazást használtam.

A jelenlegi fő feldolgozó keretrendszer megvalósítása egy Ubuntu 18.04-es virtuális gépre telepített HELK rendszeren történik. A HELK a 192.168.42.128-as IP címen elérhető, és ezen a címen érhetjük el a Kibanát, mely felület segítségével módosíthatunk az Elasticsearch és Logstash beállításain. Az egyszerűsítés kedvéért az Elasticsearch-be épített machine learning anomália detektálás API segítségével tudunk ilyen feladatot létrehozni., ilyenkor meg kell adni a vizsgálandó adathalmazt. Ezt kiválasztva meg kell adni az adat típusát, miszerint egy egyszerű idősoros adatot

vizsgálunk, vagy egy idősort több részre bontjuk kategóriánként. Lehetséges még haladó módban is használni, ahol manuálisan írhatjuk be az paramétereket. A „single metric” kiválasztása esetén ezt követően meg kell adni az időtartamot, mettől meddig szeretnénk vizsgálni az adatot, azon belül mit szeretnénk vizsgálni (kiugró adatot például), el kell nevezni a feladatot. Ezt egy validáció követi, majd a feladat kreálásával elindíthatjuk azt. Amennyiben igényünk tartja, valós időben lehet futtatni. Tekintettel arra, hogy az adatok több forrásból származnak, konfigurálnunk kell a beats Helk modult, ami küldi az adatokat a Helk-nek.

4.1 Adatok betöltése

Elemzés céljából a Helk-re adatokat küldhetünk Windowsos számítógépről is, például Windows saját loggoláshoz. Ez esetben a Sysmon, WinlogBeat és Packetbeat(hálózathoz) programokat lehet használni, amik parancssorosan telepíthetőek, szolgáltatásként futtathatóak, és előbbi a Windows eseménykezelőben is megtalálható ahogy logokat generál. Az esemény azonosítókkal ki lehet szűrni mely logokat nem kívánjuk bele adni, majd lehet részletezni mely programokat akarunk eltávolítani a loggolásból [51] [52] [53]. Jelenlegi beállítás alapján:

Log érték és jelentése	Nem loggolt alkalmazások
1 – folyamat (process) kreálás	Google updater, Microsoft monitoring agent, sysmon, nvidia display
2 – Fájlkészítési idő	
3- Hálózati kapcsolatok	Spotify, OneDrive, Officeclicktorun, Cortana, Intel management Engine, mmc.exe
5 – folyamat (process) leállítás	
6 – Kezelőprogram betöltés	Microsoft, Windows, VMware és Intel kezelőprogramok betöltése
7 - Képbetöltés	Hrome, Sysmon, mmc,VmWare, Windows defender, Onedrive
9 – Közvetlen tárhelyhozzáférés	Vmware, Google update
10 - Folyamathozzáférés	Sysmon, google update, Windows Defender, Vmware tools, Microsoft Monitoring, Onedrive
11 – Fájl létrehozás	Winlogbeat, Chrome, Onedrive
12,13,14 – Regisztrációs bejegyzés módosítása/hozzáadása/átnevezése/törlése	Google, Onedrive, Audio, Vmware
15 – fájlfolyam létrehozás	

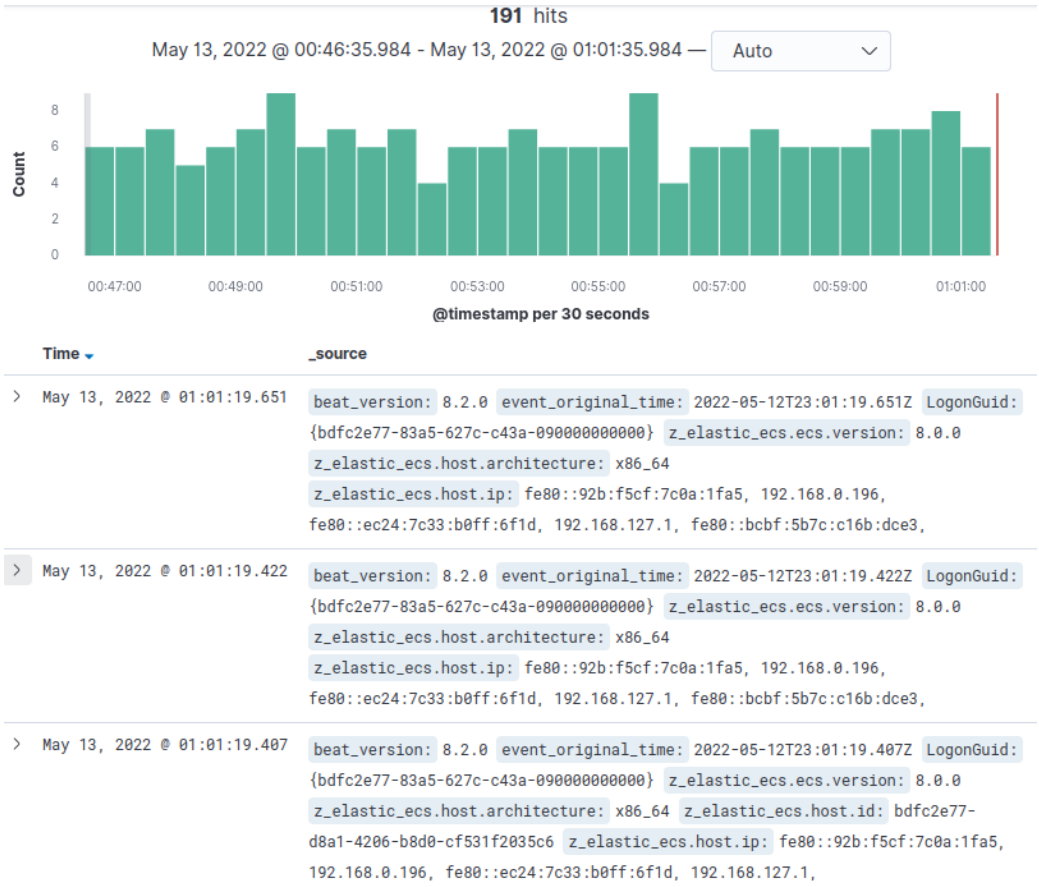
17,18 – cső(pipe) esemény létrejötte és csatlakozása	Azure, Microsoft Monitoring Agent
19,20,21 - WmiEvent	Wmi eseményekkel kapcsolatos loggolások
22 – Dns Query	
23 – Fájl törlése	Fájl törlések a C:\Windows\Temp, valamint C:\Users\ProgramData, Program Files mappából

9. ábra – Loggolás kiszűrése

Minél több ismert alkalmazás kiszűrése szükséges, mivel több GB log termelődik óránként.

A sysmon programot szolgáltatásként telepítettem, aminek futása így a szolgáltatások közül annak könnyű elindítását és leállítását eredményezi, míg mi a Windows Eseménykezelőjéből, az applikáció és szolgáltatási logokban, a Microsoft-on belül Windows mappában láthatjuk. Ezt követően kellett a fent említett konfigurációt hozzáadni és beállítani, hogy bizonyos folyamatok adatát ne rögzítse, ezáltal elkerüljük a számunkra hasztalan, nagy mennyiségű feldolgozandó adatot.

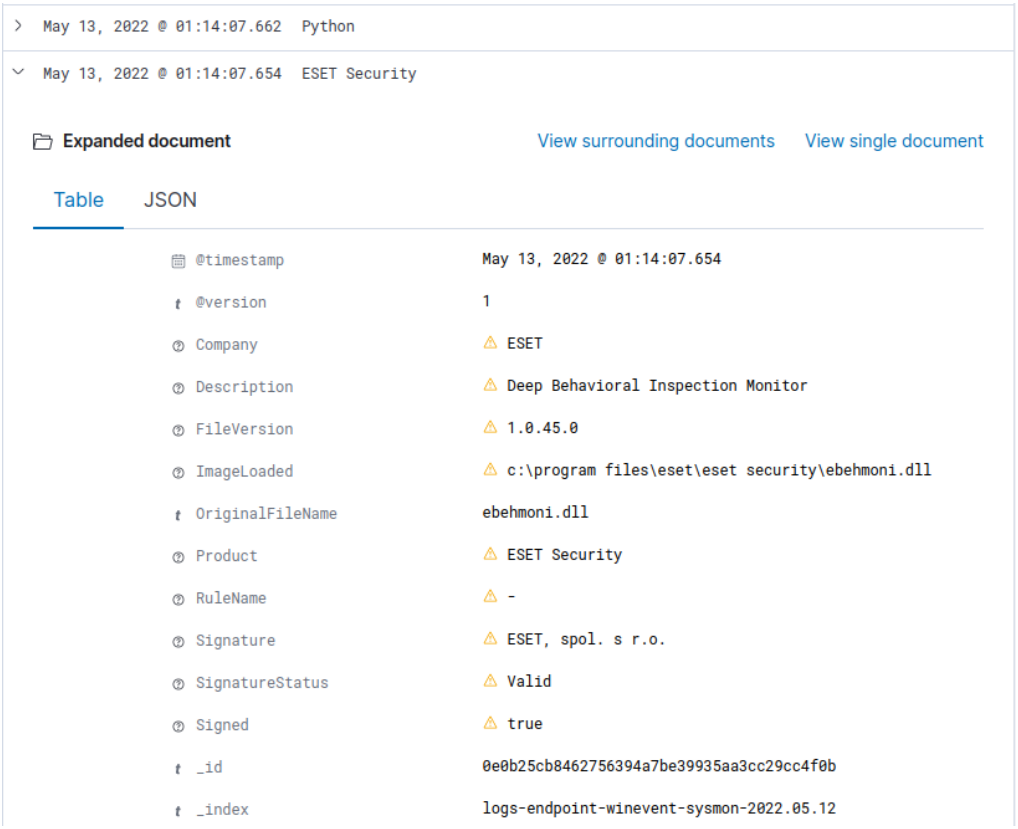
Ezt folytattam a Winlogbeat feltelepítésével a saját gépemre Powershell segítségével. A konfigurációs fájljának, a winlogbeat.yml-nek a módosítása szükséges, hogy a WinLogBeat el tudja küldeni a Sysmon által készített adatot, itt a HELK 192.168.42.128 ip címét kellett megadni az 5044-es porton, mivel a Helk egyik modulja, a Logstash ezen a porton figyel. Ezt követően a Helk-ben a Kibana Discover felületén keresztül látni a logok meglétét, a Winlogbeatünk verzióját, mely gépről küldtük, mi annak az azonosítója, és még sok mást. A kép felső részén egy grafikon látható az adott találatok számáról, alatta az előbb tárgyalt logok kibontásával részletesen látszik melyik folyamat mit csinált:



10. ábra – Adatloggolás

4.2 Anomália detektálás

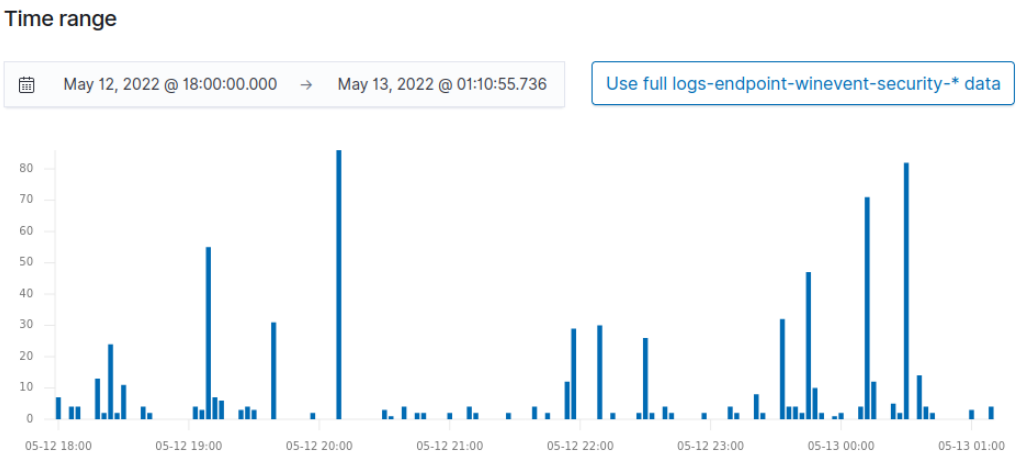
Szűrők alkalmazásával adott cég alkalmazására szűrhetünk, ezt kibontva részletesebb információkat nyerünk mivel történt esemény, ahogy az alábbi ábrán látható:



11. ábra – Eset cég alkalmazása

Ez esetben az Eset egyik dll-je dolgozott.

Sok ilyen apró adatból egy idő után az ELK egyik moduljával, az elasticsearch-el tudunk machine learning alapú anomáliadetektálást végezni. Ezen a felületen, az elasticsearch számára elérhető logokból kiválasztva, adott intervallum alapján a populációra szűrve, ezen belül további szűrőket kiválasztva és a vizsgált értéket (ez esetben windows biztonsági logokat) figyelembe véve a kiugró értékek szempontjából elkészítettem egy anomália detektálást. Ehhez az alábbi ábrán látható Windowsos biztonsági loggolást használtam több órán keresztül vizsgálva, mely esetén nincs anomália:



12. ábra – Órára lebontott adat elemzése

Ezek az adatok Windows 11-es rendszeren történő logokból épültek fel. Azonban ha kifejezetten a hálózaton levő csomagokat szeretnénk megvizsgálni, akkor ez esetben a PacketBeat nevű alkalmazásra van szükségünk. A Packetbeat használatát figyelembe vettem Windows Xp, Windows 7, és Windows 8.1 használata alatt is, de kompatibilitás, egyszerűség, sebezhetőség és megfelelő működés szempontjából utóbbira esett a választásom. A packetbeat.yml fájl konfigurálása és szolgáltatás telepítése után innen is küldi az adatokat a rendszerünk. Kibana oldalon létre kellett hozni egy index mintát (index pattern) ezekre az adatokra, hogy a Machine Learning modullal anomália detektálást tudjunk rá indítani. A Kibanának indexmintára van szüksége ahhoz, hogy a felfedezni kívánt Elasticsearch adatok elérje. Az indexminta kiválasztja a használni kívánt adatokat, és lehetővé teszi a mezők tulajdonságainak meghatározását [54]. Az így előállt lehetőségekkel létrehoztam egy ML (machine learning) jobot, ami a szolgáltatott adatok egész populációjára indít vizsgálatot, és a szokásoktól eltérő, kiugró értékekre beállított szűrővel megvizsgálja az anomália előfordulását. Ezt követte a job futása és beállítása élő feladatként, majd az anomália meghatározott szintű jelenlétekor email küldése saját címemre.

DoS/DDoS

A szolgáltatásmegtagadási (DoS) támadás olyan támadás, amelynek célja egy gép vagy hálózat leállítása, elérhetetlenné téve azt a rendeltetésszerű felhasználók számára. A DoS támadások ezt úgy érik el, hogy elárasztják a célpontot forgalommal, vagy olyan információkat küldenek neki, amelyek összeomlást váltanak ki. A DoS támadás mindkét esetben megfosztja a jogos felhasználókat az elvárt szolgálatatástól. Ennek egy további típusa az elosztott szolgáltatásmegtagadási (DDoS) támadás. Erről akkor beszélünk, ha több rendszer irányít egy szinkronizált DoS-támadást egyetlen célpontra. A lényeges különbség az, hogy ahelyett, hogy egy helyről támadnák, a célpontot egyszerre több helyről támadják [55]. A továbbiakban az eredményeknél látunk ennek észleléséről egy tesztet.

5 Megvalósítás II. – Modell oldal

5.1 Kdd99

Habár a KDD 1999-es adathalmaz már több mint 20 éves, még mindig használható tesztelésekre, emiatt az egyszerűség kedvéért ezzel dolgozom.

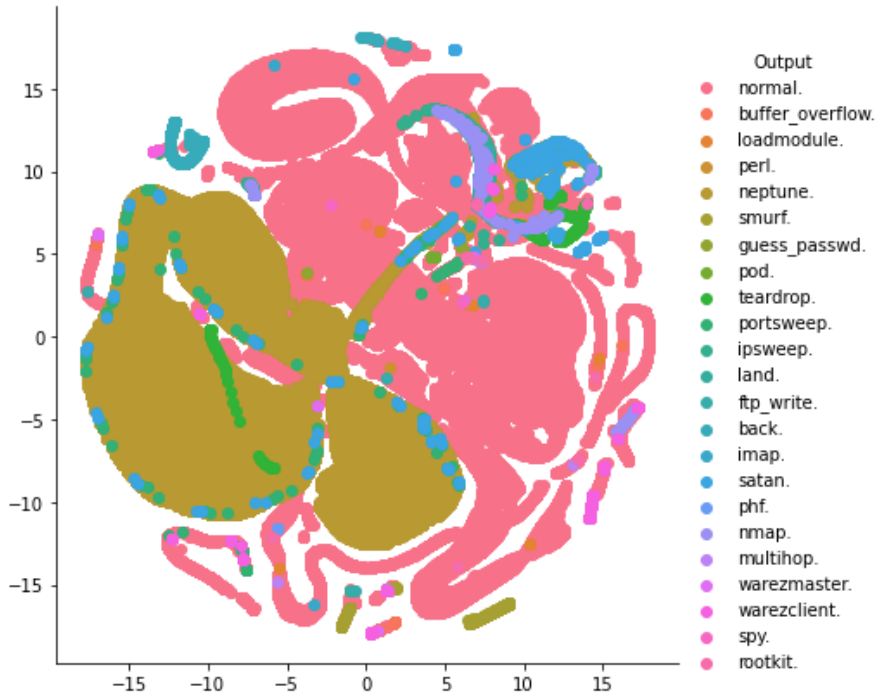
5.2 Adatfeldolgozás, tanulás

A KDD1999 adathalmaz többféle támadást tartalmaz, DoS-on belül is több típust, így kiszűrése az adott modellből felépített észlelő algoritmustól, és a tesztelő szoftver lehetőségeitől is függ.

Lépések:

1. Adat importálása
    a. A kddcup 10%-os adathalmaz importálása, feldolgozhatóság céljából ez egy kisebb fájl. 22 db. A rossz kapcsolatot (támadást) jelez, míg 1 db a jó kapcsolatot
    b. Adathalmazban szereplő null értékek és duplikátumok eltávolítása
2. Adat analizálása
    a. Mivel a támadások eloszlása nem egyenletes, ezért azt ki kell egyensúlyozni. Hogy több támadást is érzékelhető legyen a tanulásnál, ezt figyelembe kellett venni a modell kialakításánál.
    b. Dimenziócsökkentéssel megállapítjuk, hogy a kialakított 2D-s képhez nincs olyan lineáris függvény, amellyel a jó és rossz támadások elválaszthatóak lennének. Ennek a szemléltetésére használható a t-SNE függvény, mely 100-as perplexitás és 50-es iteráció mellett az

alábbi képet adja:



13. ábra – t-SNE

c. Az támadásokat tartalmazó adathalmazt többször két részre osztjuk: 75%-át tanuláshoz, 25%-át teszteléshez használjuk fel eleinte, majd megvizsgáljuk 80%-20%, és 70%-30%-al is.

3. Adat előkészítése
   a. 3 jellemzője van az adathalmazunknak, melyek protokoll, szolgáltatás, és valamilyen azonosítás (protocol, service, flag), melyeket vektorizálni fogunk „one-hot” kódolással, majd ezt megcsináljuk a „service” és „flag”-re is.
   b. Ezt követi az adatok standardizálása, ami egy vagy több attribútum újraskálázásának folyamata úgy, hogy azok átlagos értéke 0, míg szórásuk 1 legyen.
   c. A vektorizálás és standardizálás után összesítjük a jellemzőket (feature).
4. Gépi tanulási (ML) modellek implementálása az adathalmazra
   a. Gauss Naive Bayes
   b. Logistic regression
   c. SVM
   d. Decision Tree
   e. Random Forest

6 Eredmények

Az eredményeket két oldalról tudjuk megközelíteni.

- Helk észlelése
- A betanult modellek tesztelése az adathalmaz összes adatára

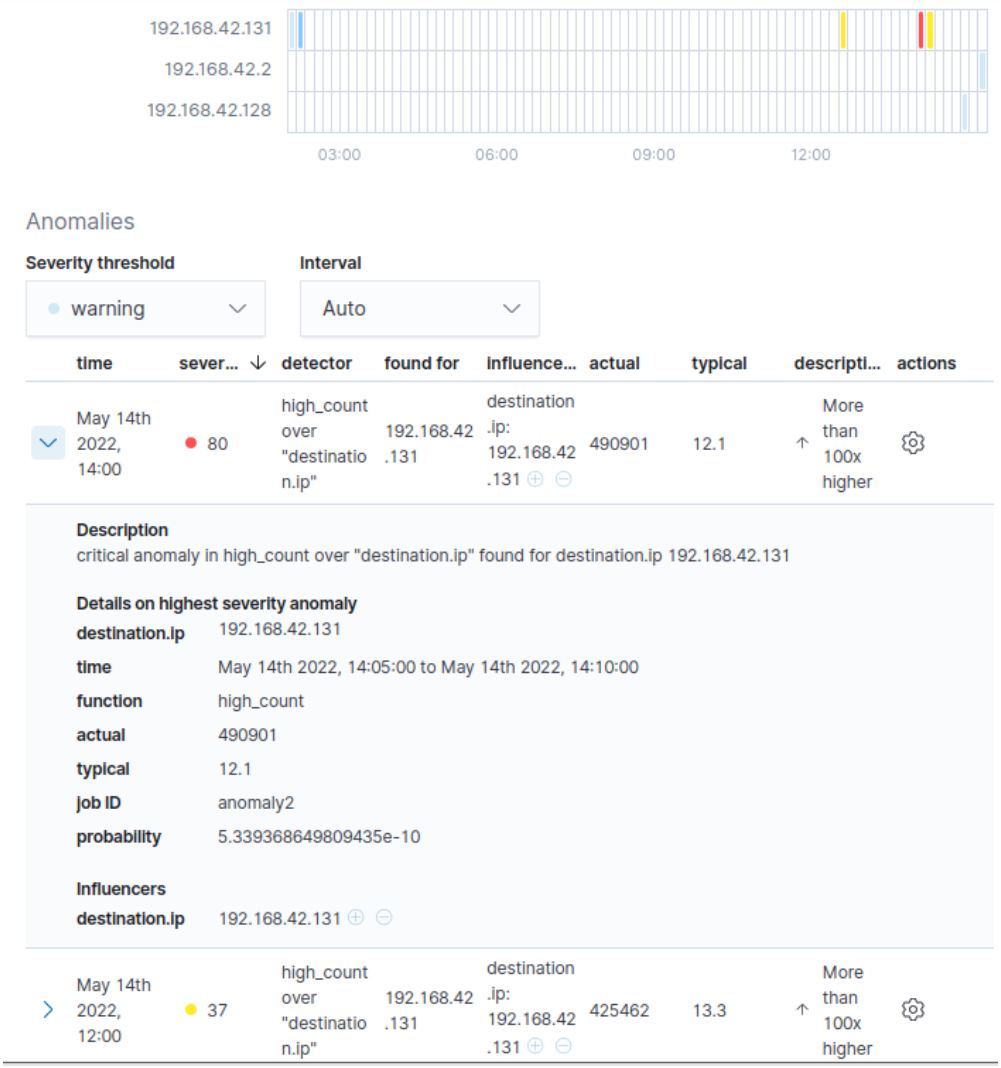
A való életben az első pontnak kiemelkedőbb fontossága, mivel valós adaton történik a teszt, másik esetében magán az adathalmazon. Való életi tesztnél viszont nem életszerű a több ezer különböző támadás lebonyolítása.

6.1 Helk oldal

A DoS támadáshoz több szoftvert tudunk használni (ezeket tesztelés céljából). Ilyen szoftver például a XOIC, mely használatában egyszerű, TCP/HTTP/UDP/ICMP támadást tud indítani adott IP-re és Portra [56]. Az alkalmazásakor a futtatandó felületen például Windosw 8.1-re 5400-as porttal az alábbi képet látjuk a támadás megkezdésének sikerességéről:

14. ábra – Xoic DoS támadás

A nemrég említett XOIC alkalmazással indítottam Windows 8.1-re a 192.168.42.131-es ip címmel és 5400-as porttal egy UDP DoS támadást, mely észlelhető volt az adatpopuláció eltéréseire létrehozott feladat által. Az alábbi ábrán az látható, hogy az adott gépre a cél ip cím szerint kiugró értékekre 14:05 és 14:10 közötti időszakra egy 80-as, előtte 37-es erősségű anomália lett rögzítve:



15. ábra – Sikeres anomália detektálás

Itt a szűrést úgy állítottam be, hogy a cél a .131-es ip című gép legyen az egyszerűség kedvéért, és mivel nem futott semmilyen hálózatot terhelő alkalmazás a Windows 8.1-es gépen, így könnyű volt kiszűrni. Természetesen a Windows tűzfalat érdemes kikapcsolni. A .131-es gép sávjában még az elején kékkel is jelez egy anomáliát, ez is ugyanúgy DoS támadás volt, csak rövidebb ideig, a 14:05-kor érzékelt támadást így nagyobb veszélynek érzékelte. A populációra indított anomáliadetektálás ezáltal sikeresnek tekinthető, azonban, ha folyamatosan támadás alatt lett volna az eszköz napokon keresztül, nem szűrte volna ki a detektáló a jelenlegi beállításokkal.

6.2 ML modellek és értékelésük

A betanult modell pontosságát a „confusion matrix” ábrázolásával tudjuk szemléltetni, számértékek esetében az FPR-nek minél kisebbnek kell lennie, TPR, F1 értéknek, Pontosság és Recallnak pedig közelebbnek kell lennie 1-hez. Az alábbi tesztek alapján látható, hogy többosztályos osztályozás esetén a tanulás / tesztelés felbontásának mértéke ugyan változik, de általánosságban kijelenthető, hogy nem jelentős mértékben. A hiperparamétereken nem változtattam, a méréseknél ugyanazok voltak, segítségükkel lehet finomhangolni a tanulást.

6.2.1 Gauss Naive Bayes

Hiperparaméterek:

var_smoothing:[10**x for x in range(-9,3)] – nullás valószínűség elkerülése érdekében

Tanulás/Tesztelés	FPR	TPR	F1_érték	Precízió	Recall
80-20	0.0519	0.9947	0.9675	0.9634	0.9743
75-25	0.0492	0.9934	0.968	0.9638	0.9742
70-30	0.0515	0.9945	0.9677	0.9639	0.9744

6.2.2 Logistic regression

Hiperparaméterek:

alpha:[0.001, 0.01, 0.1, 1, 10, 20, 30] – L1 vagy L2 „büntetés” (penalty)-hez adott súly kontrollálásához

penalty:['l1', 'l2'] – túltanulás ellen regularizáló

Tanulás/Tesztelés	FPR	TPR	F1_érték	Precízió	Recall
80-20	0.0306	0.998	0.9796	0.9767	0.9842
75-25	0.0308	0.9987	0.9811	0.978	0.9853
70-30	0.0376	0.9987	0.979	0.9821	0.9836

6.2.3 SVM

Hiperparaméterek:

alpha:[10**x for x in range(-8,3)] – L1 vagy L2 „büntetés” (penalty)-hez adott súly kontrollálásához

penalty:['l1', 'l2'] – túltanulás ellen regularizáló

Tanulás/Tesztelés	FPR	TPR	F1_érték	Precízió	Recall
80-20	0.005	0.9986	0.9961	0.996	0.9964
75-25	0.0042	0.9985	0.9969	0.9969	0.9969
70-30	0.004	0.9977	0.9965	0.9966	0.9965

6.2.4 Decision Tree

Hiperparaméterek:

max_depth:[5, 10, 20, 50, 100, 500] – A fák maximum mélysége

min_samples_split:[5, 10, 100, 500] – A minimum minta, ami kell egy belső csomópont kettéosztásához.

Tanulás/Tesztelés	FPR	TPR	F1_érték	Precízió	Recall
80-20	0.0011	0.9994	0.9988	0.9988	0.9988
75-25	0.0012	0.9985	0.9984	0.9985	0.9983
70-30	0.001	0.9993	0.9987	0.9987	0.9988

6.2.5 Random Forest

Hiperparaméterek:

max_depth:[5, 10, 100, 500, 1000] – A fák maximum mélysége

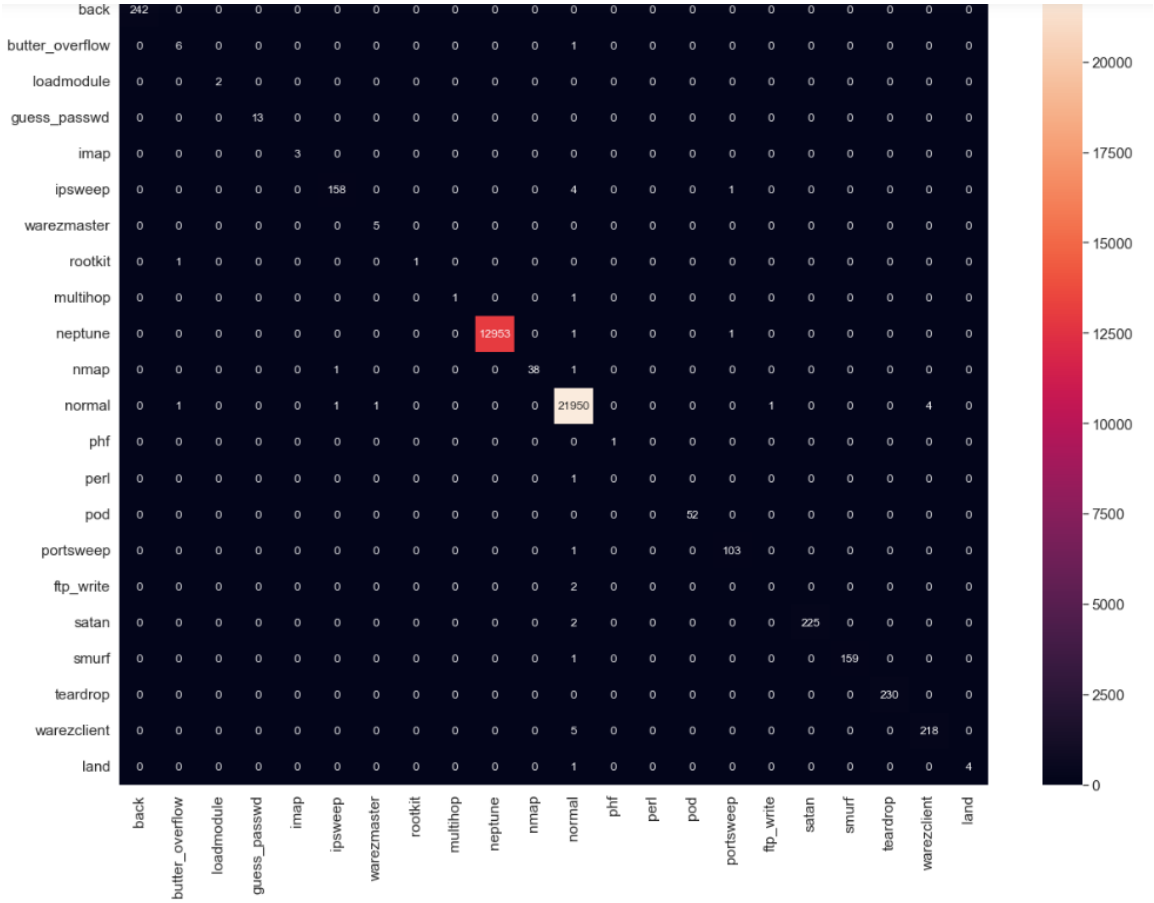
n_estimators: [5, 10, 50, 100, 500] – A fák száma az erdőben

min_samples_split:[5, 10, 100, 500] – A minimum minta, ami kell egy belső csomópont kettéosztásához.

Tanulás/Tesztelés	FPR	TPR	F1_érték	Precízió	Recall
80-20	0.0012	0.9997	0.9991	0.9991	0.9992
75-25	0.0015	0.9996	0.9991	0.999	0.9991
70-30	0.0018	0.9997	0.9989	0.9988	0.999

Az eddigi vizsgálat alapján a Random forest-es modell bizonyult a legpontosabbnak, több mérés alapján.

Továbbá szemléltethetjük a teljesítményét konfúziós mátrix-szal, mely az algoritmus performanciáját szemlélteti grafikusan. Ezen láthatóak az eltalált normál és támadásnak számító észlelések. X tengely a tervezett osztályba sorolást látni, míg az Y tengelyen a támadás / normál észlelés valódi jellemzőjét látjuk. A fehérrel jelölt érték a támadások sokaságát jelenti, ez esetben 21950 volt normál esemény, amit annak is minősített, valamint 12953 „neptune” támadás, amit érzékelt, és valóban az volt. Másik esetben az X tengelyen látható normál oszlopban az ipsweep sorában látni 4 elemet, ez azt jelenti nem érzékelte támadásnak az adatot.



16. ábra – Konfúziós mátrix

A jelenlegi megoldásokban úgy vettük figyelembe a modellek teljesítményét, hogy azokat is néztük, az adott támadásokat mennyire jól tudják osztályozni, azaz nem mint egy bináris klasszifikációs probléma. Azonban, ha csak arra vagyunk kíváncsiak, hogy az adott támadás „jó” azaz normálnak számít-e vagy nem, akkor az előbb említett módszert kell alkalmaznunk. Mivel a többklasszifikációs tesztelések esetén a Random Forest-el értem el a legjobb eredményt, így ennek vizsgálom meg a bináris formáját.

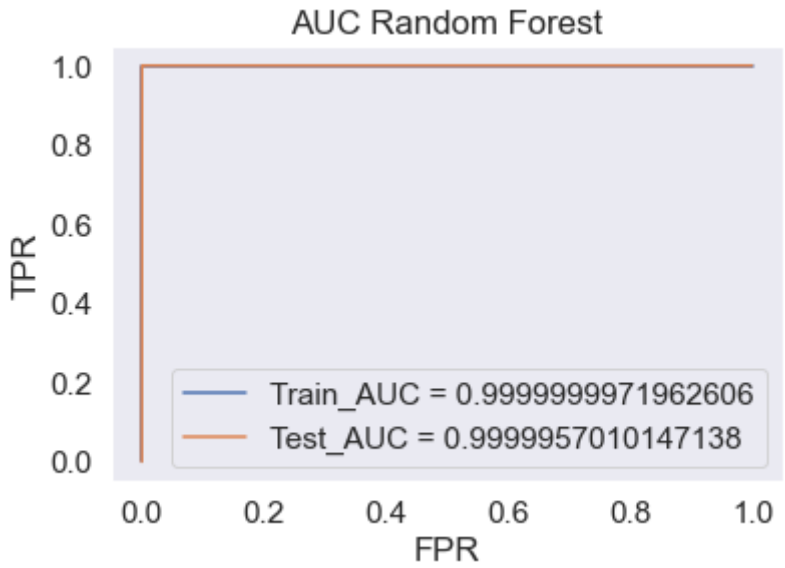
Hiperparaméterek ugyanazok

max_depth:[5, 10, 100, 500, 1000]

n_estimators: [5, 10, 50, 100, 500]

min_samples_split:[5, 10, 100, 500]

FPR	TPR	F1_érték	Pontosság	Recall
0.0017	0.9996	0.9991	0.9991	0.9991



17. ábra – AUC görbe bináris klasszifikáció esetén

A 17. ábrán látható az AUC görbe, mely egy összesítő, minél közelebb van az értéke 1-hez, annál jobban tud különbséget tenni „jó” és „rossz” kapcsolatok között. Ez alapján látható, hogy itt nagyon közel van 1-es értékhez, így ez egy megfelelő modellnek tekinthető. A fenti táblázat alapján is a 0.9991-es pontosságot tekintve látjuk, hogy jó a szűrése.

7 Továbbfejlesztési lehetőségek

A HELK, mint egy manapság egyre jobban elterjedt SOC (Security Operation Center – Biztonsági Műveleti központ) rendkívül sok lehetőséget rejt magában a valós idejű adatelemzés miatt, Machine Learning moduljával kifejezetten személyre szabható gépi tanulási feladatok készíthetőek. Fontos megjegyezni, hogy ezek a megoldások az Elasticsearch saját fejlesztése, azonban nem garantált, hogy minden esetben ez észleli a leggyorsabban, és leghatékonyabban a veszélyeket. Részletesen konfigurálható, azonban ez a modul fizetős, nyílt forráskódú változata jelenleg nem létezik. Saját fejlesztésű gépi tanulási tulajdonságokkal rendelkező modul, vagy rendszer esetén fent maradhatna a nyílt forráskódúság előnye, valamint a modellek implementálása során rendszeresen frissíteni lehetne azt, így hosszabb távon nagyon intelligens behatolás észlelő vagy megelőző rendszert lehetne kiépíteni a biztonsági központ sok előnye mellett. Jelenleg ez az Elasticsearch almodul nem rendelkezik teljes támogatottsággal saját, külső forrásból történő egyedi modellek egyszerű implementációja iránt, így saját modul implementálása emiatt is érdemes. Ekkor az általunk elkészített észlelőket éles adaton is ki tudjuk próbálni, és meg tudjuk vizsgálni betanulásuk hatékonyságát, és ingyenessé tétele segítené az egyetemi kutatások könnyebbé tételét az informatikai biztonság iránt érdeklődőknek.

8 Összegzés

Manapság az internetes támadások egyre nagyobb veszélyt jelentenek, de a hagyományos módszerek egyre kevésbé lesznek megbízhatóak. Ekkor jöhet szóba a gépi tanulás, mellyel manapság egyre jobb eredményeket érhetünk el, valamint emellé társul még a SOC (Security Operation Center – Biztonsági Műveleti központ), mely adatok, logok, hálózati forgalom elemzésével, gépi tanulással, komplex módszerekkel megfelelő védelmet tud nyújtani az újabb támadások ellen, több védelmet nyújtva, mint egy hagyományos behatolásészlelő vagy behatolásmegelőző rendszer. Egyik ilyen SOC a Hunting ELK (HELK) ami 3 fő moduljából, az Elasticsearch, Logstash, és Kibanából áll. Egy Windows 11-es gépről rendszerlogokat, Windows 8.1-es gépről hálózati logokat küldtem elemzésre a HELK-nek. Elasticsearch modulon belül készítettem egy feladatot, mely élő adatokra fut, és az adatpopuláció jellemző értékeitől eltérő értékek esetén anomáliát jelez. A Windows 8.1-es virtuális gépemre indítottam egy szolgáltatás megtagadásos (DoS) támadást, melyet a feladat sikeresen kiszűrt. Windows 10-es gép esetén a rendszerlogokban nem volt anomália.

A gépi tanulás sikerességének bizonyítása után megvizsgáltam egy másik megközelítést, mely esetén mi lenne, ha mi biztosítanánk a betanulandó, ismertebb gépi tanulási modelleket, így a HELK már előre betanult módszerrel végezné a detektálást. A modellek betanításához használt adathalmaz egy tipikus amerikai légierő helyi hálózatát szimulálja 7 hét alatt, némi támadással megfűszerezve. Az eredményeknél arra jutottam, az úgynevezett Random Forest-en alapuló modell tudja kiszűrni legjobban a támadásokat, ezen belül a leghatékonyabb az a változata, ami bináris klasszifikációval működik, azaz nem vizsgálja meg éppen milyen támadás történt, csak azt, hogy nem normál kapcsolatnak számít. Ezzel a megoldással sikerült 99,9%-os precíziót elérni.

9 Conclusion

Nowadays, cyber attacks are an increasing threat, but traditional methods are becoming less reliable. That's where machine learning comes in, which is getting better and better these days, along with the SOC (Security Operation Center), which can provide adequate protection against newer attacks by analyzing data, logs and network traffic with machine learning and complex methods, resulting on more protection than a traditional intrusion detection or prevention system. One such system is Hunting ELK (HELK) which consists of 3 main modules, Elasticsearch, Logstash, and Kibana. During my work, I sent system logs from a Windows 11 machine and network logs from a Windows 8.1 machine to HELK for analysis. Within the Elasticsearch module, I created a task that runs on live data and reports anomalies for values that differ from the typical values of the data population. I launched a denial of service (DoS) attack on my Windows 8.1 virtual machine, which was successfully filtered by the task. On a Windows 10 machine, there was no anomaly in the system logs.

After demonstrating the success of machine learning, I investigated another approach, where we provide the more familiar machine learning models to be learned, so that HELK performs detection using a pre-learned method. The data set I trained the models with simulates a typical US Air Force local area network over 7 weeks, with some attack added. In the results, I found that the model based on the so-called Random Forest is the best at detecting attacks, and the most effective version of this model is the one that uses binary classification, i.e. it does not look at exactly what attack happened, only that it is not a normal connection. With this solution, 99.9% precision was achieved.

10 Irodalomjegyzék

[1] A. Khraisat, I. Gondal, P. Vamplew és J. Kamruzzaman, „Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity,* %1. kötet2, %1. szám1, 17 07 2019.

[2] Y. Keshet, „Cynet," 20 Március 2020. [Online]. Available: https://www.cynet.com/blog/half-of-the-malware-detected -in-2019-was-classified-as-zero-day-threats-making-it-the-most- common-malware-to-date/. [Hozzáférés dátuma: 22 November 2020].

[3] „AV-Test," 2020. [Online]. Available: https://www.av-test.org/en/statistics/malware/. [Hozzáférés dátuma: 22 November 2020].

[4] H. N. Security, „As malware and network attacks increase in 2019, zero day malware accounts for 50% of detections," 13 December 2019. [Online]. Available: https://www.helpnetsecurity.com/2019/12/13/network-attacks-2019/. [Hozzáférés dátuma: 22 November 2020].

[5] C. S. t. LTD, „NTSC," 2020. [Online]. Available: https://www.ntsc.org/assets/pdfs/cyber-security-report-2020.pdf. [Hozzáférés dátuma: 22 November 2020].

[6] C. Kreibich és J. Crowcroft, „Honeycomb: creating intrusion detection signatures using honeypots," *ACM SIGCOMM Computer Communication Review,* %1. kötet34, %1. szám1, pp. 51-56, Január 2004.

[7] X. A. Liu, R. C. Meiners, J. Patel, E. Norige és E. Torng, „Fast Regular Expression Matching Using Small TCAMs for Network Intrusion Detection and Prevention Systems," *Conference: 19th USENIX Security Symposium,* pp. 111-126, Szeptember 2010.

[8] P.-C. Lin, Y.-D. Lin és Y.-C. Lai, „A Hybrid Algorithm of Backward Hashing and Automaton Tracking for Virus Scanning," *IEEE Transactions on Computers,* %1. kötet60, %1. szám4, pp. 594-601, 22 Április 2020.

[9] X. Du és S. Dua, Data Mining and Machine Learning in Cybersecurity, 1 szerk., London: CRC Press, 2011, pp. 86-88.

[10] S. Swaminathan, „Logistic Regression — Detailed Overview," 15 03 2018. [Online]. Available: https://towardsdatascience.com/logistic-

regression-detailed-overview-46c4da4303bc. [Hozzáférés dátuma: 10 05 2022].

[11] R. Gandhi, „Naive Bayes Classifier,” 05 05 2018. [Online]. Available: https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c. [Hozzáférés dátuma: 01 04 2022].

[12] A. Yadav, „SUPPORT VECTOR MACHINES(SVM),” 20 10 2018. [Online]. Available: https://towardsdatascience.com/support-vector-machines-svm-c9ef22815589. [Hozzáférés dátuma: 02 05 2022].

[13] W. Education, „SVM algoritmus,” 2022. [Online]. Available: https://hu.education-wiki.com/4245111-svm-algorithm. [Hozzáférés dátuma: 10 05 2022].

[14] P. Yadav, „Decision Tree in Machine Learning,” 13 11 2018. [Online]. Available: https://towardsdatascience.com/decision-tree-in-machine-learning-e380942a4c96. [Hozzáférés dátuma: 14 05 2022].

[15] T. Yiu, „Understanding Random Forest,” 12 06 2019. [Online]. Available: https://towardsdatascience.com/understanding-random-forest-58381e0602d2. [Hozzáférés dátuma: 28 04 2022].

[16] H. N. B, „Confusion Matrix, Accuracy, Precision, Recall, F1 Score,” 10 12 2019. [Online]. Available: https://medium.com/analytics-vidhya/confusion-matrix- accuracy-precision-recall-f1-score-ade299cf63cd. [Hozzáférés dátuma: 01 04 2022].

[17] A. Vidhya, „AUC-ROC Curve in Machine Learning Clearly Explained,” 16 06 2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/. [Hozzáférés dátuma: 01 05 2022].

[18] R. Rodriguez, „The HELK,” 2020. [Online]. Available: https://thehelk.com/intro.html. [Hozzáférés dátuma: 12 12 2020].

[19] P. Fórum, „PC Fórum,” [Online]. Available: https://pcforum.hu/szotar/heap. [Hozzáférés dátuma: 12 12 2020].

[20] Elastic, „Elastic.co,” 2020. [Online]. Available: https://www.elastic.co. [Hozzáférés dátuma: 12 12 2020].

[21] Stackshare, „Elasticsearch vs Logstash,” 2020. [Online]. Available: https://stackshare.io/stackups/elasticsearch-vs-logstash. [Hozzáférés dátuma: 12 12 2020].

[22] Apache, „Kafka,” 2017. [Online]. Available: https://kafka.apache.org/intro. [Hozzáférés dátuma: 12 12 2020].

[23] N. Narkhede, „Introducing KSQL: Streaming SQL for Apache Kafka,” 28 08 2017. [Online]. Available: https://www.confluent.io/blog/ksql-streaming-sql-for-apache-kafka/. [Hozzáférés dátuma: 12 12 2020].

[24] Easyalert, „ElastAlert - Easy & Flexible Alerting With Elasticsearch,” 2014. [Online]. Available: https://elastalert.readthedocs.io/en/latest/elastalert.html. [Hozzáférés dátuma: 12 12 2020].

[25] Tutorialspoint, „Machine Learning - Jupyter Notebook,” 2020. [Online]. Available: https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_jupyter_notebook.htm. [Hozzáférés dátuma: 12 12 2020].

[26] Jupyter, „Jupyter,” 18 11 2020. [Online]. Available: https://jupyter.org/. [Hozzáférés dátuma: 12 12 2020].

[27] Apache, „Apache Spark,” 2018. [Online]. Available: https://spark.apache.org/. [Hozzáférés dátuma: 12 12 2020].

[28] P. Mahto, „PANDAS For Machine Learning,” 31 07 2020. [Online]. Available: https://medium.com/mlpoint/pandas-for-machine-learning-53846bc9a98b. [Hozzáférés dátuma: 15 05 2021].

[29] UNB, „Datasets,” UNB, [Online]. Available: https://www.unb.ca/cic/datasets/index.html. [Hozzáférés dátuma: 15 05 2021].

[30] I. University of California, „KDD Cup 1999 Data,” 28 10 1999. [Online]. Available: https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. [Hozzáférés dátuma: 15 05 2021].

[31] pandas, „pandas,” 12 04 2021. [Online]. Available: https://pandas.pydata.org/. [Hozzáférés dátuma: 15 05 2021].

[32] J. Terra, „Keras vs Tensorflow vs Pytorch: Understanding the Most Popular Deep Learning Frameworks,” 31 05 2021. [Online]. Available: https://www.simplilearn.com/keras-vs-tensorflow-vs-pytorch-article. [Hozzáférés dátuma: 15 05 2021].

[33] F. AI, „fast.ai,” 2021. [Online]. Available: https://www.fast.ai/. [Hozzáférés dátuma: 15 05 2021].

[34] C. S. Technology, „The shellcode generation,” *IEEE,* %1. kötet2, %1. szám5, pp. 72-76, 04 10 2004.

[35] A. Shenfield, D. Day és A. Ayesh, „Intelligent intrusion detection systems using artificial neural networks,” *ICT Express,* %1. kötet4, %1. szám2405-9595, pp. 95-99, 06 2018.

[36] N. Farnaaz és M. A. Jabbar, „Random Forest Modeling for Network Intrusion Detection System,” in *Procedia Computer Science,* 2016.

[37] Cloudflare, „What is a DDoS Attack?,” 2020. [Online]. Available: https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/. [Hozzáférés dátuma: 12 12 2020].

[38] D. Chamou, P. Toupas, E. Ketzaki, S. Papadopoulos, M. K. Giannoutakis, A. Drosou és D. Tzovaras, „Intrusion Detection System based on Network Traffic using Deep Neural Networks,” *24th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks,* pp. 1-6, 17 10 2019.

[39] Snort, „Snort,” 2020. [Online]. Available: https://www.snort.org/. [Hozzáférés dátuma: 13 12 2020].

[40] Nmap, „Nmap,” [Online]. Available: https://nmap.org/. [Hozzáférés dátuma: 13 12 2020].

[41] B. Subba, S. Biswas és S. Karmakar, „A Neural Network Based System for Intrusion Detection and Attack Classification,” *International Conference on Advances in Intelligent Systems,* %1. szám148-04, pp. 1-6, 02 2014.

[42] H. Bai, C.-z. Hu, X.-c. Jing, N. Li és X.-y. Wang, „Approach for malware identification using dynamic behaviour and outcome triggering," *Information Security, IET,* %1. kötet8, pp. 140-151, 01 03 2014.

[43] A. Andreatos és V. Moussas, „A Novel Intrusion Detection System Based on Neural Networks," 01 2019. [Online]. Available: https://www.researchgate.net/publication/335995823_A_Novel_ Intrusion_ Detection_System_Based_on_Neural_Networks. [Hozzáférés dátuma: 13 12 2020].

[44] A. M. Shibani és E. Anupriya, „Automated Threat Hunting Using ELK Stack – A Case Study," *Indian Journal of Computer Science and Engineering,* %1. kötet10, pp. 118-127, 10 2019.

[45] C. Nicholson, „A Beginner's Guide to Multilayer Perceptrons (MLP)," 2020. [Online]. Available: https://wiki.pathmind.com/multilayer-perceptron. [Hozzáférés dátuma: 15 05 2021].

[46] N. Bharat, „Network Intrusion Detection System using Pytorch," 16 05 2019. [Online]. Available: https://github.com/bharatnishant/IDS. [Hozzáférés dátuma: 15 04 2022].

[47] R. Basnet, R. Shash, C. Johnson, L. Walgren és T. Doleck, „Towards Detecting and Classifying Network Intrusion Traffic Using Deep Learning Frameworks," %1. kötet7, %1. szám104, pp. 1-17, 12 2019.

[48] A. Javaid, Q. Niyaz, W. Sun és M. Alam, „A Deep Learning Approach for Network Intrusion Detection System," *EAI Endorsed Transactions on Security and Safety,* %1. kötet3, 3-5 12 2015.

[49] B. Nishant, „Network Intrusion Detection System using Pytorch," 15 10 2020. [Online]. Available: https://github.com/bharatnishant/IDS. [Hozzáférés dátuma: 10 05 2021].

[50] M. Tamim-Ul-Haq, „Intrusion Detection System using Deep Learning," 05 01 2019. [Online]. Available: https://github.com/tamimmirza/Intrusion-Detection-System-using-Deep-Learning. [Hozzáférés dátuma: 10 05 2021].

[51] Microsoft, „Sysmon," 11 05 2022. [Online]. Available: https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon. [Hozzáférés dátuma: 14 05 2022].

[52] Elasticsearch, „Packetbeat,” [Online]. Available: https://www.elastic.co/beats/packetbeat. [Hozzáférés dátuma: 13 05 2022].

[53] Elasticsearch, „Winlogbeat,” [Online]. Available: https://www.elastic.co/guide/en/beats/winlogbeat/ current/winlogbeat-installation-configuration.html. [Hozzáférés dátuma: 14 05 2022].

[54] Elastic, „Create an index pattern,” 2022. [Online]. Available: https://www.elastic.co/guide/en/kibana/7.17/index-patterns.html. [Hozzáférés dátuma: 05 05 2022].

[55] P. A. Networks, „What is a denial of service attack (DoS)?,” 2022. [Online]. Available: https://www.paloaltonetworks.com/cyberpedia/what-is-a-denial-of-service-attack-dos. [Hozzáférés dátuma: 05 05 2022].

[56] AppNee, „Xoic,” 11 02 2021. [Online]. Available: https://appnee.com/xoic/. [Hozzáférés dátuma: 10 05 2022].

[57] „Neural network models (supervised),” Scikit-learn, 2020. [Online]. Available: https://scikit-learn.org/stable/modules/neural_networks_supervised.html. [Hozzáférés dátuma: 15 05 2021].

[58] „GNU Wget,” Free Software Foundation, 04 08 2020. [Online]. Available: https://www.gnu.org/software/wget/. [Hozzáférés dátuma: 16 05 2021].

[59] H. L. Arash, „CICFlowmeter,” 08 03 2021. [Online]. Available: https://github.com/ahlashkari/CICFlowMeter. [Hozzáférés dátuma: 10 05 2021].

[60] M. Vasiq, „Netflow2CSV,” 30 03 2018. [Online]. Available: https://github.com/vasiqmz/netflow2csv. [Hozzáférés dátuma: 15 05 2021].

11 Rövidítések függelék

Rövidítés	Jelentés
AIDS	Anomaly Based Intrusion Detection System
ANN	Artificial Neural Network
API	Application Programming Interace
AUC	Area Under the Curve
CR	Classification Rate
DDoS	Distributes Denial of Service
DNN	Deep Neural Network
FNR	False Negative Rate
FPR	False Positive Rate
HELK	Hunting ElasticSearch, Logstash and Kibana
HIDS	Host Based Intrusion Detection System
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
MBO	Malicious Behaviors and Outcomes
MIB	Management Information Base
MLP	Multilayer Perceptron
NIDS	Network Based Intrusion Detection System
REST	Representational State Transfer
ROC	Receiver Operating Characteristic
SIDS	Signature Based intrusion Detection System
SNMP	Simple Network Management Protoco
TCP	Transmission Control Protocol
TDNN	Time Delay Neural Network
TPR	True Positive Rate
UDP	User Datagram Protocol

12 Ábrajegyzék