Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Hálózati Rendszerek és Szolgáltatások Tanszék

Ádám Zsombor

VPN ÉSZLELÉSI TECHNIKÁK HATÉKONYSÁGÁNAK ÉRTÉKELÉSE

KONZULENS

Dr. Pekár Adrián

BUDAPEST, 2024

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

Alulírott **Ádám Zsombor**, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Kelt: Budapest, 2024. 12. 10.

...…….
Ádám Zsombor

Kivonat

A szakdolgozat célja a virtuális magánhálózat (Virtual Private Network: VPN) forgalom észlelésének és azonosításának hatékonyságát vizsgálni, különös tekintettel a modern gépi tanulásos módszerek alkalmazására, valamint egy új, hatékonyabb módszer kidolgozása és implementálása. A téma aktualitását és fontosságát a VPN technológiák széleskörű elterjedése adja, amelyek egyaránt használhatók adatvédelemre és káros tevékenységek álcázására. Ez a kettős szerep különösen indokolttá teszi az ilyen forgalmak pontos azonosítására irányuló kutatásokat, mivel a VPN-ek helytelen vagy elégtelen detektálása komoly biztonsági kockázatokat jelenthet. A dolgozat három különböző gépi tanulásos megközelítést vizsgál a VPN és nem VPN forgalmak osztályozására, különböző jellemzőkkel és algoritmusokkal. Az eredmények alapján a dolgozat rámutat, hogy a gépi tanulásos modellek képesek magas pontosságú osztályozásra, esetenként a 100%-os pontosságot is elérve. Az eredmények igazolják a módszerek alkalmazhatóságát és hatékonyságát is. Az implementált megoldások hozzájárulnak a hálózati biztonság növeléséhez, mivel hatékony eszközöket kínálnak a VPN forgalom detektálására. Ez nemcsak a biztonsági rendszerek fejlesztését segíti, hanem az internethasználat megbízhatóságát és átláthatóságát is javítja, jelentős lépést téve a biztonságosabb hálózati környezet megteremtése felé.

Abstract

The aim of this thesis is to examine the efficiency of detecting and identifying Virtual Private Network (VPN) traffic, with a particular focus on the application of modern machine learning methods, as well as to develop and implement a new, more effective approach. The relevance and importance of this topic stem from the widespread adoption of VPN technologies, which can be used both for data protection and to conceal malicious activities. This dual role makes research into the accurate identification of such traffic particularly justified, as improper or insufficient detection of VPNs can pose significant security risks. The thesis explores three different machine learning approaches for classifying VPN and non-VPN traffic, utilizing various features and algorithms. Based on the results, the thesis demonstrates that machine learning models are capable of high-accuracy classification, achieving up to 100% accuracy in certain cases. The findings validate both the applicability and effectiveness of these methods. The implemented solutions contribute to enhancing network security by providing efficient tools for VPN traffic detection. This not only supports the development of security systems but also improves the reliability and transparency of internet usage, representing a significant step towards creating a safer network environment.

1 Bevezetés

A szakdolgozat a virtuális magánhálózat (Virtual Private Network: VPN) forgalom észlelésének és azonosításának hatékonyságával foglalkozik, különös figyelmet fordítva a modern gépi tanulásos módszerek alkalmazására. Célja, hogy egy új, hatékonyabb módszert dolgozzon ki, amely pontosabb és megbízhatóbb az eddig használt megoldásoknál. A dolgozat különböző gépi tanulási megközelítéseket is vizsgál a VPN és nem VPN forgalom osztályozására.

A VPN technológiák elterjedése egyrészt növelte az adatvédelmet és a hálózati biztonságot, másrészt új kihívásokat teremtett, mivel egyes esetekben káros tevékenységek álcázására is használják őket. Ezek a problémák különösen fontosak az IT biztonság és a hálózati védelem területén, hiszen a VPN-ek jelenléte megnehezíti a forgalom típusának azonosítását, ezáltal veszélyeztetve a hálózati rendszerek stabilitását és biztonságát.

A dolgozat három különböző gépi tanulásos módszert alkalmaz, melyek mindegyike eltérő jellemzőket és algoritmusokat használ a VPN és nem VPN forgalom megkülönböztetésére. A kutatás során valós forgalmi adatokon felhasználásával kerül sor az algoritmusok hatékonyságának mérésére és egy új megközelítés kidolgozására. Az implementációt Python alapú keretrendszerek segítségével valósítottam meg, amely lehetővé teszi az eredmények reprodukálását és összehasonlítását.

Az eredmények szerint a gépi tanuláson alapuló modellek képesek a forgalmakat magas pontossággal osztályozni, ezáltal hatékony eszközt biztosítva a VPN-ek észleléséhez. Az alkalmazott megközelítések közül az újonnan kidolgozott módszer legalább olyan jó, vagy jobb teljesítményt nyújtott a vizsgált esetekben, ezzel bizonyítva annak hatékonyságát.

Az implementált megoldások nemcsak a hálózati biztonságot növelhetik, hanem tovább finomíthatják a VPN forgalom azonosításának jövőbeni fejlesztéseit is. A kutatás eredményei támogatják a kiberbiztonsági szakemberek munkáját, és hozzájárulnak a VPN-ekkel kapcsolatos fenyegetések hatékonyabb kezeléséhez, így közvetlen hatással lehetnek a modern hálózatbiztonsági rendszerek fejlődésére.

A dolgozat a következőképpen tagolódik: A 2. fejezetben bemutatom általánosságban a VPN-t, annak alkalmazásait, illetve a különböző VPN protokollokat. A 3. fejezet a VPN észlelési technikákkal foglalkozik, mint az IP elemzés, mély csomagvizsgálat és gépi tanulás, valamint itt részletezem a dolgozathoz kapcsolódó korábbi kutatásokat is. Az adathalmaz tulajdonságait, a három VPN észlelési eljárást, továbbá ezek eredményeit és összehasonlítását a 4. fejezetben írom le. Végül a dolgozatot értékelem és elemzem, valamint a továbbfejlesztési lehetőségeit tárgyalom.

2 Kapcsolódó technológiák

2.1 A hálózati biztonság fejlődése és a VPN szükségessége

Az internet őskorában még nem volt szükség a hálózati forgalom elrejtésére, mivel ekkor még kevés felhasználó csatlakozott a hálózathoz, és a csomópontok többsége megbízhatónak számított. Az internetes kommunikáció biztonságos volt, mert a fenyegetések száma és összetettsége alacsony volt. Azonban, ahogy az internet elérhetőbbé vált és felhasználók milliárdjai számára vált mindennapi szükségletté, a potenciális biztonsági kockázatok is drámai mértékben megszaporodtak.

Általános jelenség, hogy amikor kialakul egy széles körben használt szolgáltatás, a növekvő népszerűséggel arányosan jelennek meg a rosszindulatú szereplők is. Ezért idővel elengedhetetlenné válik a szolgáltatások biztonságának garantálása, különösen az érzékeny személyes adatok védelme érdekében. A mai internet esetében ez a probléma különösen égetővé vált, számos vállalat és egyén próbál hasznot húzni az átlagfelhasználók adataiból.

Manapság az adathalászat, az online megfigyelés és a kiberbűnözés olyan szintre fejlődött, hogy szinte minden internetfelhasználót érinthet. A nagy technológiai cégek, például a Meta [1] és a Google [2], hatalmas adatbázisokat építenek fel felhasználóik viselkedéséről, szokásairól és személyes információiról. Míg ezek az adatok általában célzott reklámokhoz vagy üzleti elemzésekhez kerülnek felhasználásra, sokszor a felhasználók tudta és kifejezett beleegyezése nélkül történik az adatgyűjtés.

Ezzel párhuzamosan, hackerek és kiberbűnözők szofisztikált támadásokat indítanak szerverek, vállalatok és magánszemélyek ellen, hogy érzékeny adatokat szerezzenek meg. Ezek az adatok később gyakran bűncselekményekhez, például identitáslopáshoz vagy zsarolóprogramok használatához vezetnek. További kihívást jelentenek a nyilvános Wi-Fi hálózatok, amelyek különösen veszélyesek lehetnek az átlagos felhasználók számára, hiszen ezeken a hálózatokon gyakran könnyű célpontot jelentenek az adatlopásra specializálódott támadóknak.

Ezek a fenyegetések megteremtették az igényt olyan megoldásokra, amelyek biztosítják a felhasználók online tevékenységének magánéletét és védelmét. Így mára egy egész iparág épült az online biztonságra, különösen a VPN-szolgáltatók köré.

2.2 A VPN technológia áttekintése

Sok definíciója van a VPN-nek, nincs egy mindenki által elfogadott egységes, de [3]-ban így fogalmazzák meg: „A VPN olyan technológia, amely biztonságos kommunikációt biztosít egy adott helyek csoportja vagy egy zárt felhasználói csoport között, egy megosztott hálózati infrastruktúra kihasználásával.”

Eredeti célja ennek a technológiának az volt, hogy egy privát hálózatot távolról is biztonságosan el lehessen érni. Ami egy céges hálózatnak különösen fontos mivel, ha csak szimplán rá lenne csatlakoztatva az internetre, akkor az nyilvánvalóan komoly biztonsági kockázattal járna, hiszen bárki számára elérhetővé válna az adott cég privát hálózata. Ha csak jelszavas védelmet alkalmaznak, akkor is sokkal kevésbé biztonságos, mintha alapból csak egy VPN-csatornán keresztül érjük el a cég belső hálózatát.

2.3 A VPN felhasználási területei

Az említett céges, egyetemi és állami szervekhez tartozó VPN hálózatok a mai napig az egyik fő felhasználási módja ennek a technológiának. Emellett viszont sok más felhasználási mód is kialakult az évek során.

A VPN-szolgáltatók az elmúlt néhány évben gombaként nőtték ki magukat, a már korábban említett fokozódó internetes fenyegetettség miatt. Általában egy adott havi vagy éves előfizetési díjért cserébe több a felhasználók biztonságát és kényelmét célzó szolgáltatásokat kínálnak. Egyik legjobban eladható szolgáltatásuk, hogy több VPN szervert üzemeltetnek különböző országokban, amelyekhez a felhasználók igényeik szerint csatlakozhatnak, így kihasználva az egyes szoftverek és weboldalak országonkénti dinamikus árazását, továbbá streaming szolgáltatók országonkénti eltérő kínálatát.

Másik lehetséges felhasználása a VPN-nek a cenzúra megkerülése. Egyes országokban korlátozva van, hogy milyen weboldalak érhetőek el az internetet használók számára. A legékesebb példája ennek az úgynevezett Kínai Nagy Tűzfal (Great Firewall of China), amely meghatározott belépési pontokon szűri a bejövő tartalmakat IP, Domain Name System (DNS) és még kulcsszavak alapján is így megakadályozva, hogy egyes oldalakat elérjenek a felhasználók [4]. VPN használatával sokszor kikerülhetőek ezek a korlátozások [5]. Egyes személyek tarthatnak attól is, hogy különböző állami szervek lekövetik az online tevékenységeit, ennek a megkerülésére is alkalmas lehet a VPN.

A technológia előnyei mellett ugyanakkor meg kell említeni a káros velejáróját is, mert a VPN-csatorna ahogy álcázni tud egy ártalmatlan forgalmat, ugyanígy tud akár egy célzott támadást is elrejteni, ezért fontos kutatási terület az, hogy azonosítani lehessen, hogy egy adott hálózati kommunikáció VPN-en keresztül zajlik-e [6].

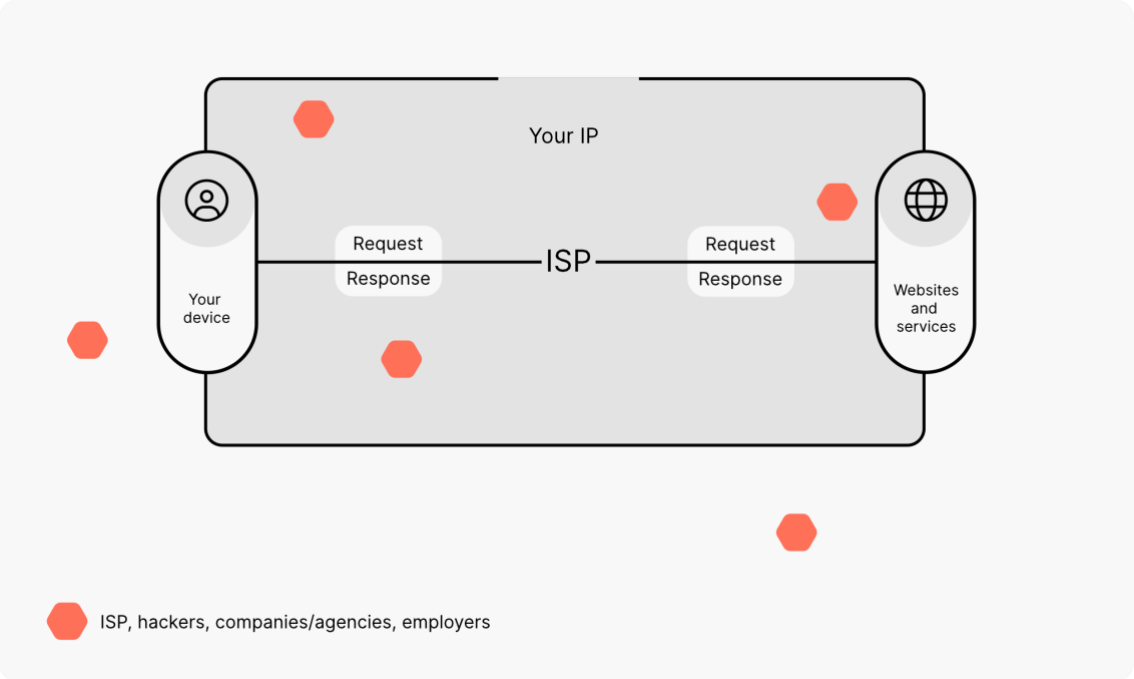
2.4 A VPN technológia működése részletesen

A VPN segítségével a rendszergazda létrehozhat egy helyi hálózatot olyan számítógépek között, amelyek eltérő hálózati szegmenseken helyezkednek el. Ezek az eszközök közös helyi hálózaton lehetnek, de akár az interneten keresztül egymástól nagy távolságra is csatlakozhatnak. A VPN forgalom csatornázott adatfolyam, amely különbözik az átlagos csatornán kívüli egyéb hálózati forgalmaktól.

2.4.1 Hálózati forgalom VPN nélkül

Egy VPN nélküli kliens és szerver közötti hálózati forgalom közvetlenül, titkosítás nélkül zajlik. A kliens, például egy böngésző vagy más alkalmazás, kapcsolatot kezdeményez a szerverrel az interneten keresztül. Amikor a kliens egy adatcsomagot küld a szerver felé, az adatcsomag először áthalad a helyi hálózaton (Local Area Network: LAN) vagy a hozzáférési pontokon, például egy Wi-Fi routeren, majd az adatcsomag az internetszolgáltató (Internet Service Provider: ISP) hálózatára kerül, ahol továbbítják a cél IP-címe felé, amely a szerveré.

Az adatcsomag az interneten különböző útválasztókon (routereken) halad keresztül, amelyek az optimális útvonalat választják ki a cél elérése érdekében. Ez az útvonal azonban nem garantáltan biztonságos, mivel az adatcsomagok titkosítatlan formában is mozoghatnak, és potenciálisan ki vannak téve lehallgatásnak, manipulációnak vagy támadásoknak, például közbeékelődéses (Man-in-the-Middle: MITM) támadásoknak.



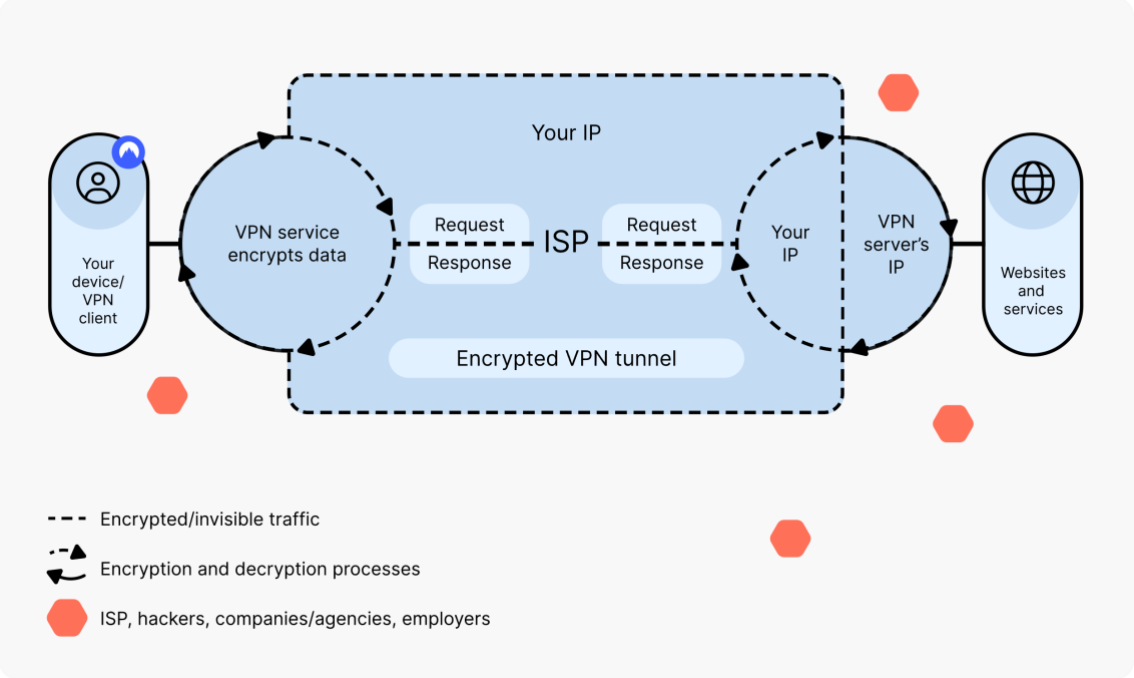
2-1. ábra: Hálózati forgalom VPN nélkül [7]

Amikor a szerver megkapja a kliens kérését, feldolgozza azt, és válaszokat küld vissza ugyanazon az útvonalon. Ez azt jelenti, hogy a válasz is ugyanazokon a nyilvános hálózati csomópontokon halad keresztül, mielőtt elérné a klienst. Az ilyen típusú kommunikációban az adatok biztonsága az alkalmazott protokolloktól függhet, például a Hypertext Transfer Protocol Secure (HTTPS) használata biztosítja a weboldalak és a böngészők közötti titkosítást, de maga az adatkapcsolat továbbra is sebezhető maradhat a hálózat többi rétegén.

2.4.2 Hálózati forgalom VPN-en keresztül

Egy VPN-en keresztüli kliens és szerver közötti kommunikáció lényegesen biztonságosabb, mint a hagyományos, VPN nélküli adatforgalom. A VPN használatakor a kliens először kapcsolatot létesít a VPN szerverrel, amely egy titkosított User Datagram Protocol (UDP) vagy Transmission Control Protocol (TCP) csatornát hoz létre a kliens eszköze és a szerver között. Ez a csatorna biztosítja, hogy az adatcsomagok titkosított formában haladjanak, így azok tartalmát útközben nem lehet lehallgatni vagy manipulálni. Ezen a csatornán keresztül bármilyen szokásos hálózati forgalom működhet.

Amikor a kliens egy adatcsomagot küld a cél szerver felé, az adatot először titkosítja a kliens, majd a titkosított csomagot elküldi a VPN szervernek, ami egy titkosított csatornán belül biztonságosan továbbítódik. A szerver ezt követően tovább küldi az adatokat a cél IP-cím felé az interneten keresztül, de itt már a csomag a VPN szerver IP-címét tartalmazza, így elrejtve annak igazi forrását. A válaszadás fordított irányban történik: a cél szerver által küldött adatcsomag először eljut a VPN szerverhez, amely titkosított csatornán keresztül továbbítja azt a kliens felé, ahol az adat visszafejtése megtörténik.



2-2. ábra: Hálózati forgalom VPN-en keresztül [7]

Bizonyos VPN megoldások hitelesítést is végeznek, amely két különálló részből áll, és eltérő szerepeket tölt be. Az első a felhasználói vagy rendszerszintű hitelesítés, amelynek célja, hogy biztosítsa, hogy csak jogosult felhasználók csatlakozhassanak a szolgáltatáshoz. Ezt meg lehet valósítani például felhasználónként kiadott tanúsítványokkal vagy felhasználónév és jelszó kombinációjával. Ezen felül egyéni szabályok is definiálhatók az adott felhasználók számára, például meghatározott útvonalak, tűzfalszabályok, illetve különböző szkriptek alkalmazásával.

A hitelesítés másik eleme a kommunikációs adatfolyam további biztonságát biztosítja. Itt az adatok küldése előtt minden csomag egyedi aláírást kap. A címzett rendszer ellenőrzi, hogy a beérkező csomagok aláírása helyes-e, mielőtt a csomag tartalmát (payload) dekódolná. Ez a módszer lehetővé teszi, hogy a rendszer kizárólag az érvényes csomagokkal foglalkozzon, így időt és erőforrást spórol, mivel a szabályoknak nem megfelelő csomagokat eleve elutasítja. A megoldás hatékony védelmet nyújt például a szolgáltatásmegtagadási (Denial-of-service: DoS) támadásokkal és a közbeékelődéses támadásokkal szemben, feltéve, hogy az aláírásokhoz használt kulcsok biztonságban maradnak.

A legnagyobb különbség tehát a titkosításban és az anonimitásban rejlik. Míg a VPN nélkül az adatcsomagok nyilvános hálózatokon titkosítatlanul haladhatnak, és az internetszolgáltatók vagy harmadik felek könnyedén hozzáférhetnek azok tartalmához, addig egy VPN használatakor a hálózati forgalom enkriptálva van, így még akkor is, ha valaki megpróbálja lehallgatni az adatokat, azok értelmezhetetlenek maradnak. Emellett a VPN szerver által biztosított új IP-cím elrejtheti a kliens eredeti földrajzi helyét és identitását, ami további biztonságot és adatvédelmet nyújthat [3],[6],[8].

2.5 VPN protokollok

A VPN protokollok kulcsszerepet játszanak abban, hogy a virtuális magánhálózatok hogyan biztosítják a biztonságos és privát adatkommunikációt. Ezek a protokollok határozzák meg, hogy a VPN miként titkosítja és továbbítja az adatokat a kliens és a szerver között, valamint milyen szintű biztonságot, sebességet és kompatibilitást kínál. A következő oldalak átfogó képet nyújtanak a legismertebb VPN protokollokról, köztük a Point-to-Point Tunneling Protocol (PPTP), amely egyszerűsége miatt hosszú ideig népszerű volt. Az Internet Protocol Security (IPsec), amely megbízható titkosításáról ismert. A Secure Sockets Layer / Transport Layer Security (SSL/TLS) VPN-ek, amely széles körben elterjedt az online adatvédelemben. Az OpenVPN, amely rugalmas és nyílt forráskódú megoldást nyújt, valamint a WireGuard, amely az új generációs VPN-ek gyorsaságát és hatékonyságát képviseli. Ezek az eltérő protokollok más-más előnyökkel és felhasználási területekkel rendelkeznek [6],[9].

2.5.1 PPTP

A PPTP az egyik legrégebbi VPN protokoll, amelyet a Microsoft fejlesztett ki 1999-ben. Az egyszerűsége és az operációs rendszerek széles körű támogatottsága miatt kezdetben népszerű megoldás volt a virtuális magánhálózatok kialakítására. A hálózati rétegben működik, és egy TCP-alapú vezérlőcsatornát használ a kapcsolat felépítésére (1723-as porton keresztül), míg az adatcsatorna a General Routing Encapsulation (GRE) protokollt alkalmazza az adatforgalom továbbítására.

2.5.1.1 PPTP működése

A protokoll két csatornát használ: egy vezérlőcsatornát a kapcsolat felépítéséhez és fenntartásához, valamint egy adatcsatornát a tényleges adatforgalom továbbítására. A kapcsolat létrehozásához a kliens és a szerver a TCP kézfogást elvégzi, majd vezérlőüzeneteket cserélnek. Amennyiben a verziók kompatibilisek, a kapcsolat sikeresen létrejön, és elkezdődhet az adatforgalom. Ha a két fél egyszerre próbál kapcsolatot létesíteni, a magasabb IP-címmel rendelkező eszköz folytathatja a kapcsolatfelvételt.

2.5.1.2 PPTP előnyei és hátrányai

Legnagyobb előnye az egyszerűsége. A protokoll könnyen beállítható, még technikai háttértudás nélkül is, és alacsony erőforrás-igényének köszönhetően gyors kapcsolatot biztosít. A legtöbb modern operációs rendszer, köztük Windows, Linux, Unix-alapú rendszerek, iOS és Android, beépített támogatást nyújt a PPTP-hez, ami tovább csökkenti a konfigurációs időt.

Azonban mára elavultnak számít, mivel jelentős biztonsági hiányosságai vannak. A protokoll gyenge titkosítási algoritmusokat alkalmaz, amelyek érzékenyek a „brute-force” támadásokra. A hitelesítési folyamat is problémás, mivel a Microsoft Challenge-Handshake Authentication Protocol-t (MS-CHAP) használja, amely számos biztonsági rést tartalmaz. Bár az Extensible Authentication Protocol-al (EAP) és X.509 tanúsítványokkal növelhető a biztonság, ezt nem minden kliens támogatja. További hátrány, hogy a PPTP által használt GRE protokoll rosszul működik Network Address Translation (NAT) alapú hálózatokon, ami kompatibilitási problémákhoz vezethet. Emellett a PPTP forgalma könnyen azonosítható és blokkolható tűzfalak által, mivel nem használ szabványos VPN portokat [8],[10],[11].

2.5.2 IPsec

Az IPsec egy olyan protokollcsoport, amely az IP csomagok titkosításával és hitelesítésével biztosítja a hálózati kommunikáció biztonságát. Az Open Systems Interconnection (OSI) modell hálózati (3.) vagy adatkapcsolati (2.) rétegében működik. PPTP-hez hasonlóan az IPsec is két csatornát használ, egy kontroll csatornát (UDP 500 vagy 4500 porton alapértelmezetten) a kapcsolat kialakításához és egy adat csatornát az adatok továbbításához. Főként két üzemmódban működik: csatorna mód és transzport mód. Csatorna módban az egész IP csomag titkosítást kap egy új IP fejléccel együtt, míg transzport módban csak a csomag tartalma kerül titkosításra.

2.5.2.1 IPsec működése

Működése során kulcsfontosságú szerepet tölt be az Internet Key Exchange (IKE) protokoll, amely a titkosítási kulcsok cseréjéért és a biztonsági paraméterek egyeztetéséért felel. A kapcsolat felépítése két szakaszban zajlik: az első fázisban megtörténik a hitelesítés és a titkosítási kulcsok cseréje, míg a második fázisban létrejön az adatcsatorna, amely biztosítja a titkosított adatátvitelt. Két fontos protokollt használ: az Encapsulated Security Payload (ESP) az adatok titkosítására és integritásának biztosítására szolgál, míg az Authentication Header (AH) az adatok hitelességét ellenőrzi.

2.5.2.2 IPsec előnyei és hátrányai

Az IPsec egyik legnagyobb előnye a magas szintű biztonság. A protokoll transzparens módon, a hálózati rétegben valósítja meg a titkosítást, így bármilyen alkalmazással kompatibilis. Emellett széles körű platformtámogatottsággal rendelkezik, és részletesen konfigurálható biztonsági szabályokat kínál. A transzparens működés előnye, hogy automatikusan lefedi az összes hálózati forgalmat, anélkül, hogy az alkalmazások szintjén beállításokra lenne szükség.

Ugyanakkor az IPsec használatának hátrányai is vannak. A konfiguráció bonyolult, amelyhez szakértelem szükséges. A titkosítás és hitelesítés többlet sávszélességet és számítási kapacitást igényel, ami lassíthatja a hálózat teljesítményét. További problémát jelenthet a NAT alapú hálózatokkal való korlátozott kompatibilitás, mivel nem kezeli jól az IP-címek módosítását. Emellett az eltérő gyártók implementációi nem mindig kompatibilisek egymással [8],[9],[12].

2.5.3 SSL/TLS VPN-ek

Az SSL/TLS VPN-ek az SSL vagy a modernebb TLS protokollra épülő virtuális magánhálózatok. Elsődleges előnyük, hogy modern böngészők segítségével működnek, így nincs szükség külön kliensszoftverre, ami megkönnyíti a használatukat és növeli a platformfüggetlenségüket. Két típusa létezik: a portálalapú megoldások, amelyek egy weboldalon keresztül biztosítanak hozzáférést korlátozott számú erőforráshoz, és a csatornaalapú VPN-ek, amelyek teljes hálózati szolgáltatásokat tesznek elérhetővé, akár böngészőn kívüli alkalmazások számára is. Népszerűsége annak köszönhető, hogy a HTTPS-hez hasonló protokollokat használnak, amelyek széles körben elfogadottak és támogatottak, még gyakran ugyanazt a TCP 443-as portot is használják.

2.5.3.1 SSL/TLS működése

A kapcsolatfelépítés során a felhasználó hitelesíti magát a VPN szerver felé, majd a szerver és a kliens a TLS kézfogás segítségével megállapodik a titkosítási paraméterekről és kulcsokról. Ez a folyamat X.509 tanúsítványokat, valamint felhasználónév-jelszó párosokat vagy egyszeri jelszavakat használ a biztonság érdekében. A kapcsolat létrejötte után az összes adat titkosított formában kerül továbbításra, biztosítva annak sértetlenségét. A munkamenet végeztével a kapcsolat biztonságosan megszakad.

2.5.3.2 SSL/TLS VPN-ek előnyei és hátrányai

Az SSL/TLS VPN-ek fő előnyei közé tartozik a széles körű kompatibilitás, mivel szinte minden modern böngésző támogatja a TLS protokollt, így nincs szükség további szoftverre. Ezenkívül erős titkosítást nyújtanak, amely megvédi az adatforgalmat az illetéktelen hozzáféréstől, és lehetőséget biztosítanak a hozzáférés finomhangolására.

Hátrányaik közé tartozik, hogy kizárólag az általuk kezelt csatornákat védik, így más alkalmazások nem élvezik az SSL VPN nyújtotta biztonságot. Továbbá nagy adatforgalom esetén a böngészők teljesítménye csökkenhet, és nyilvános eszközök használata esetén további kockázatok merülhetnek fel [8],[13],[14].

2.5.4 OpenVPN

Az OpenVPN egy nyílt forráskódú, széles körben használt VPN protokoll, amely az SSL/TLS protokollra épül, hogy biztonságos és privát adatforgalmat biztosítson hálózatok között. Támogatja az X.509 tanúsítványokat és előre megosztott kulcsokat, továbbá kiegészíti a biztonságot Hash-based Message Authentication Code (HMAC) és hash algoritmusokkal a csomagok integritásának ellenőrzésére. Kompatibilis a legnépszerűbb operációs rendszerekkel, mint a Linux, Windows, macOS, valamint iOS és Android, ugyanakkor minden eszközön kliensszoftver telepítését igényli.

2.5.4.1 OpenVPN működése

Működése során az OpenVPN először hitelesíti a klienst és a szervert digitális tanúsítványok, felhasználói azonosítók vagy a nyilvános kulcsú infrastruktúra segítségével. Ezt követően titkosított csatornát hoz létre, amelyen keresztül az adatforgalom halad. Az adatcsomagokat egy extra rétegbe burkolja, amely biztosítja az útvonal-információk és a tartalom védelmét harmadik felek hozzáférésével szemben. Az adatok titkosítása után azok a VPN szerveren keresztül jutnak el a végső célállomásra, így a kliens eredeti IP-címe rejtve marad.

2.5.4.2 OpenVPN előnyei és hátrányai

Az OpenVPN számos előnnyel rendelkezik. Magas szintű biztonságot nyújt, amely a nyílt forráskódú fejlesztés révén könnyen ellenőrizhető és javítható. Kompatibilis a legtöbb eszközzel és hálózati környezettel, akár NAT-olt hálózatokban is működhet. Rugalmasan konfigurálható különböző titkosítási protokollokkal, és támogatja mind a TCP, mind az UDP forgalmat, alapértelmezett beállításként az UDP 1194-es portot használja. Emellett olyan fejlett funkciókat kínál, mint a hardveres tokenek és különféle hitelesítési mechanizmusok.

Hátrányai közé tartozik, hogy a telepítés kliensoldali szoftvert igényel, ami bonyolultabbá teszi a használatát a kliens nélküli megoldásokhoz képest. Konfigurációja manuális beállításokat igényelhet, ami időigényes lehet, és teljesítménye lassabb lehet más protokollokhoz, például a WireGuardhoz képest. Nagyobb rendszerek esetén kevésbé skálázható, és régebbi hardvereken az erőforrásigény problémát jelenthet [8],[15].

2.5.5 WireGuard

A WireGuard egy modern, hálózati rétegszintű VPN protokoll, amelyet Jason Donenfeld fejlesztett ki azzal a céllal, hogy gyorsabb, biztonságosabb és egyszerűbb alternatívát kínáljon a hagyományos VPN megoldások, például az OpenVPN és az IPSec helyett. A protokoll kiemelkedik kompakt kódstruktúrájával: mindössze körülbelül 4000 sorból áll, szemben más protokollok több százezer soros kódjával. Ez a letisztult kialakítás nemcsak könnyen kezelhetővé és ellenőrizhetővé teszi, hanem kisebb erőforrásigényt is eredményez, miközben kiemelkedő sebességet és biztonságot biztosít.

2.5.5.1 WireGuard működése

A ChaCha20Poly1305 algoritmust használja az adatok hitelesített titkosítására UDP protokollon keresztül, és egy fejlett NoiseIK kulcscsere mechanizmust alkalmaz a biztonságos kapcsolat megteremtésére. A szolgáltatásmegtagadásos támadások elleni védekezés érdekében egy továbbfejlesztett IP kötési süti rendszert használ. Az alapértelmezett portja az 51820. A protokoll az összes jelentős operációs rendszert támogatja, beleértve a mobil- és asztali platformokat, ami univerzálisan alkalmazhatóvá teszi.

2.5.5.2 WireGuard előnyei és hátrányai

Az előnyei közé tartozik a kiemelkedő teljesítmény, amely gyorsabb adatfolyamot és kisebb eszközterhelést eredményez, ezáltal javítva az eszköz akkumulátorának élettartamát. A WireGuard a legmodernebb kriptográfiai megoldásokat alkalmazza, biztosítva a felhasználók számára a biztonságos adatforgalmat. Emellett lehetővé teszi a gyors újracsatlakozást, mivel nem igényel bonyolult kézfogási hitelesítést, így a forgalom megszakítás nélkül folytatható.

Hátrányai között említhető, hogy a protokoll nem a legjobb választás azoknak, akik maximális adatvédelmet keresnek, mivel az UDP használata miatt a forgalom könnyen azonosítható VPN adatként [9],[16].

2.6 Összefoglalás

A mai internetes környezetben a hálózati biztonság kritikus kérdéssé vált. A fejezetben bemutattam a hálózati biztonság fejlődésének szükségességét, kiemelve a növekvő fenyegetések, például az adatszivárgás és a kiberbűnözés hatását. A VPN technológia ezen problémák megoldására nyújt eszközt, biztosítva a titkosított adatforgalmat és a felhasználók anonimitását.

A VPN-ek sokféle felhasználási területen bizonyítják hasznosságukat, a magánélet védelmétől kezdve az üzleti adatok biztonságos továbbításáig. A VPN technológia részletes működéséről is szó volt, összehasonlítva a VPN nélküli és VPN-en keresztüli forgalmat, megvilágítva a titkosított csatornák szerepét az adatvédelemben. Valamint különböző VPN protokollokat is ismertettem, kiemelve azok előnyeit és hátrányait.

Ugyanakkor a VPN-ek azonosítására és osztályozására irányuló fejlett technikák iránti igény egyre fontosabbá válik. Az ilyen eszközöket gyakran használják rosszindulatú célokra is, például tiltott tartalmak elérésére vagy támadások elrejtésére, ami kiemeli az észlelés és azonosítás jelentőségét. Az azonosítás képessége elengedhetetlen a rosszindulatú tevékenységek észlelésében és a hálózati szabályozások betartatásában, különösen a modern hálózatbiztonság szempontjából.

A következő, fejezet részletesen bemutatja a VPN-ek észlelésére alkalmazott technikákat, kezdve az IP elemzésétől, a mély csomagvizsgálaton át, egészen a gépi tanuláson alapuló módszerekig. Ez a fejezet azonosítja a jelenlegi módszerek erősségeit és gyengeségeit, egyben megalapozva a dolgozat további irányainak és fejlesztéseinek.

3 VPN észlelési technikák

A VPN észlelése egyre fontosabbá válik a hálózati biztonság és az adatvédelmi szabályozások szempontjából. A VPN-ek lehetővé teszik a felhasználók számára, hogy titkosított kapcsolatokat létesítsenek, ezáltal védve a személyes adatokat és elkerülve a megfigyelést az interneten. Ugyanakkor gyakran használatosak a hálózatok körüli blokkok megkerülésére, a földrajzi korlátozások áthidalására, illetve az online anonim aktivitás biztosítására, akár kiberbűnözők által is. Továbbá lehetővé teszi, hogy egyetlen felhasználó több száz, különböző országokból származó IP-címhez férjen hozzá, így megkerülve a korlátozásokat. A VPN használat gyakran összefüggésbe hozható botokkal végzett támadásokkal és automatizált fenyegetésekkel is. Ezért a VPN használatának észlelése kulcsfontosságú azok számára, akik figyelemmel szeretnék kísérni a hálózati forgalmat, például a kiberbiztonsági szakemberek, internetszolgáltatók vagy cégek.

A VPN forgalom észlelésének több különböző technikája létezik, amelyek célja annak felismerése, hogy egy adott adatcsomag VPN-en keresztül kerül-e továbbításra. Az észlelés alapja különböző módszerekre építhet, a forgalom elemzésére, a titkosítási algoritmusok és a protokollok vizsgálatára, a forrás és cél IP-címek monitorozására, valamint a csomagok viselkedésének és mintázatának elemzésére. Az ilyen technikák segítségével lehetséges a VPN forgalom kiszűrése, ami különösen fontos lehet a nem kívánt adatforgalom blokkolásához, a bűnözői tevékenységek elleni védelemhez, vagy a vállalati biztonsági előírások betartásához.

3.1 IP elemzés

Az IP elemzés hatékony technika a VPN forgalom azonosítására, mivel az IP-címek és azok viselkedési mintáinak vizsgálatával tárja fel az árulkodó jeleket. Ennek során a VPN-ek által használt IP tartományokat elemzik, amelyek sokszor könnyen felismerhetők nyilvános adatbázisok segítségével. A nagyobb VPN-szolgáltatók, mint a NordVPN vagy az ExpressVPN, által használt IP-címek gyakran szerepelnek ilyen adatbázisokban, lehetővé téve azok azonosítását. Ezeket az adatbázisokat ugyanakkor folyamatosan frissíteni kell, mert a VPN-szolgáltatók is folyamatosan váltogatják IP-címeiket.

A geolokációs eltérések szintén árulkodóak lehetnek, hiszen a VPN-t használó felhasználók IP-címének földrajzi helye sok esetben nem egyezik meg a valós tartózkodási helyükkel. A dinamikus IP-címek figyelése szintén az IP elemzés fontos eleme. Ha egy eszköz rövid időn belül gyakran váltogatja az IP-címeket, az VPN használatra utalhat [17].

A VPN forgalom gyakran állandó vagy strukturált mintázatot követ, például egyetlen IP-címről származó, nagy mennyiségű forgalom formájában, ami a közös IP-használatra utalhat, ezért fordított DNS elemzés is árulkodó lehet, hiszen az IP-címekhez társított domainnevek sokszor tartalmaznak „vpn” vagy „proxy” előtagokat, amelyek a VPN-szolgáltatásokhoz kapcsolhatóak. Az IP-címekhez rendelt portok és protokollok elemzése is segíthet az azonosításban, hiszen egyes VPN-ek specifikus portokat használnak.

Az IP elemzés egyik legnagyobb előnye az egyszerűsége és gyorsasága, hiszen alacsony erőforrásigénnyel végezhető el, ha nem számítjuk bele az adatbázis naprakészen tartását. A hamis pozitív értékek elméletben nem lehetségesek (szintén az adatbázis állapotától függ), ami egy nagyon fontos szempont, mivel így nem blokkol olyan forgalmat, ami valójában nem is VPN forgalom.

Az IP elemzés kihívásokat is rejt magában. A VPN-ek által használt IP-címek gyakran változnak, így az adatbázisok naprakészen tartása kifejezetten rosszul skálázódik, mivel minden egyes VPN-szolgáltatóra meg kell ismételni a folyamatokat, továbbá egyes VPN-szolgáltatók dinamikusan generált vagy obfuszkált IP-címeket alkalmaznak, ami még tovább nehezíti a pontos azonosítást.

3.2 Mély csomagvizsgálat

A mély csomagvizsgálat (Deep Packet Inspection: DPI) egy olyan fejlett hálózati elemzési technológia, amely lehetővé teszi az adatforgalom alapos vizsgálatát a csomagok tartalmának szintjén. Míg a fentebb tárgyalt IP elemzés csak az IP fejlécben található alapvető információkat elemzi (például forrás- és célcím, protokolltípus), a DPI képes a teljes csomag tartalmát is átvizsgálni, ezáltal nemcsak a csomagok származási és rendeltetési helyét lehet azonosítani, hanem a bennük lévő adatokat, például alkalmazás- vagy felhasználói szintű információkat is.

A mély csomagvizsgálat különösen hasznos VPN-ek észlelésében, mivel képes azonosítani a titkosított adatfolyamokat, például az OpenVPN, WireGuard vagy IPSec által használt mintázatokat. Ezeket a protokollokat gyakran a csomagstruktúra, az időzítés és a titkosítás jellemzői alapján lehet felismerni. További előnye, hogy lehetőséget nyújt a forgalom szűrésére, például adott típusú csomagok blokkolására, ami hatékony eszközzé teszi a nem kívánt VPN forgalom kiszűrésében.

Ugyanakkor a DPI-nek vannak korlátai is. A titkosított forgalom vizsgálata során a protokoll jellemzői azonosíthatók, de az adatok tartalma titkosítva marad, így a konkrét adatokat nem lehet megismerni. Felismerési pontosságával kapcsolatban is vannak kivetnivalók, mivel egy VPN szerver megváltoztathatja az IP/TCP/UDP fejléceket a küldött csomagokon, hogy ne lehessen VPN protokollhoz társítani a kinyert adatokat. Az alkalmazása azonban adatvédelmi aggályokat is felvethet, mivel a hálózati forgalom mélyreható elemzése invazív lehet, különösen olyan esetekben, amikor személyes vagy bizalmas adatok kerülnek vizsgálat alá [18].

Az OpenDPI és az nDPI a mély csomagvizsgálati technológiák népszerű implementációi. Az OpenDPI célja a hálózati forgalom átláthatóbb elemzése és a különböző alkalmazások adatfolyamainak felismerése volt. Az nDPI, amely az OpenDPI utódjaként tekinthető, kiterjeszti annak funkcionalitását, és fejlettebb protokollfelismerési képességekkel rendelkezik, beleértve a titkosított és VPN forgalom azonosítását is. Az nDPI a titkosított protokollokhoz, például az SSL/TLS és a modern VPN megoldásokhoz speciális felismerési algoritmusokat használ, amelyek figyelembe veszik a forgalom szerkezetét, az alkalmazott portokat, valamint a szokásos viselkedési mintákat. Mindkét eszköz nyílt forráskódú, és gyakran használják kiberbiztonsági rendszerekben, hálózati monitorozó eszközökben és forgalomelemzési projektekben, ahol a protokoll-azonosítás és a forgalom elemzése kiemelt fontosságú [19].

3.3 Szofisztikált megközelítések

A gépi tanulás az adatvezérelt mesterséges intelligencia egyik ága, amely algoritmusokat használ a mintázatok és összefüggések automatikus felismerésére, nagy mennyiségű adatban. Az algoritmusok tanulási folyamatuk során példákból és tapasztalatokból származó adatokat használnak a predikciók vagy döntések javítására. A gépi tanulás három fő típusa a felügyelt tanulás, ahol címkézett adatokat használ az algoritmus, a felügyelet nélküli tanulás, amelyben a cél, a minták és a struktúrák felismerése címkézetlen adatokban, és a megerősítéses tanulás, ahol a rendszer egy adott környezetben történő cselekvések után kapott visszacsatolásból tanul [20].

A gépi tanulás a VPN forgalom detektálásában is ígéretes megoldásnak bizonyul. A gépi tanulási algoritmusok képesek a forgalmi adatokban rejlő mintázatok és anomáliák felismerésére, mint például a csomagméret, az időbeli eloszlás, a kapcsolat tartama vagy az adatfolyam dinamikája. A titkosított adatfolyamok elemzésekor a gépi tanulás képes lehet egyedi mintázatok felismerésére anélkül, hogy szükség lenne a titkosítás feloldására. Bár a gépi tanulás ígéretes technika, annak eredményessége nagyban függ az alkalmazott modellek minőségétől, az adatfeldolgozás alaposságától és az algoritmusok megfelelő paraméterezésétől. A hatékonyság növelése érdekében különös figyelmet kell fordítani a tanítási adatok minőségére és az algoritmusok optimalizálására.

Előnyei közé sorolható a már említett rugalmas mintázatfelismerése. Könnyen adaptálódik, ami alatt azt kell érteni, hogy folyamatosan fejleszthető új adatokkal való tanítás során. Ezzel a technikával könnyedén automatizálható a hálózati forgalmak elemzése. Legnagyobb előnye talán a gépi tanuláson alapúló technikának az, hogy képesek egyszerre több jellemzőt figyelembe venni és ezek komplex összefüggéseiben osztályozni a VPN forgalmat.

Számos előnye mellett ugyanakkor hátrányai is vannak ennek az eljárásnak is. Egyik legjelentősebb, hogy nagy mennyiségű és jól felcímkézett adatra van szükség hozzá, hogy megfelelőképp rá tudjon a forgalom mintázataira tanulni, általában ez is szokott a legnagyobb akadály lenni az eljárás megvalósításában. Ezek a modellek általában nem tudják 100% bizonyossággal osztályozni a forgalmat (bár szokott közelíteni hozzá), elő szoktak fordulni téves pozitív és negatív eredmények is. Nagy erőforrást igényelnek, mivel minél nagyobb az adatbázis annál több számítási kapacitásra is van szükség. Valamint az is probléma tud lenni, hogy egyes modellek „feketedoboz" jellegűen működnek (pl. Neurális háló), ezért nehéz megérteni pontosan mi alapján döntenek.

Ezen hátrányok mellett is a jelenlegi legjobb módszer arra, hogy VPN forgalmat detektáljunk, ami abból is feltételezhető, hogy a témában a jelenlegi kutatások többsége különböző jellemző halmazok alapján, de gépi tanulással próbálja osztályozni a VPN forgalmakat. A következőkben ezeknek a kutatásoknak egy részéről is szó lesz.

4 Kapcsolódó kutatások

A hálózati forgalom osztályozása régóta az informatikai kutatások egyik központi területe, amely az évek során jelentős fejlődésen ment keresztül. A korai kutatásokban elsősorban a mély csomagvizsgálat alapú technikák domináltak, mint például a [21]-ben bemutatott Peer-to-peer (P2P) forgalom azonosítása vagy a [22]-ben ismertetett Bloom-filterek párhuzamos alkalmazása. Bár ezek a módszerek hatékonyak voltak a titkosítatlan forgalom elemzésében, komoly hátrányokat szenvedtek a titkosított adatfolyamok kezelésében, és implementációjuk nagy erőforrásigénnyel járt. A DPI egyszerűsítésére irányuló törekvések, például reguláris kifejezések alkalmazása [23], szintén nem tudták teljes mértékben orvosolni a módszer korlátait.

A gépi tanulás alapú megközelítések megjelenése forradalmasította a hálózati forgalom osztályozását, különösen a statisztikai jellemzőkön alapuló módszerek terén. Például a [24] és a [25] tanulmányok bemutatják a k-közép klaszterelemzés és más gépi tanulási algoritmusok alkalmazását hálózati forgalom osztályozására. A VPN forgalom detekciójában különösen fontos szerepet játszanak a titkosított adatfolyamok időbeli és statisztikai jellemzői, amelyeket a [26] és [27] munkák részletesen bemutatnak. Ezek a kutatások megmutatták, hogy a gépi tanulás képes hatékonyan felismerni a VPN forgalom mintázatait, még titkosított környezetben is.

Az ISCXVPN2016 adatbázis [28] kulcsfontosságú szerepet játszott a VPN detekciós kutatások előmozdításában. Az adatbázis egy egységes referenciát biztosít a különböző módszerek összehasonlításához, lehetővé téve az objektív mérési környezetet. Ennek köszönhetően számos kutatás, például a [28] és a [27], azonos adathalmazon tudta validálni a módszereit, elősegítve a VPN detekciós technikák fejlődését.

A hálózati forgalom elemzésének megkönnyítéséhez jelentősen hozzájárult az NFStream keretrendszer [29], amely egy rugalmas, nyílt forráskódú eszköz Packet Capture (PCAP) fájlok feldolgozására. Az NFStream alkalmazása lehetővé tette a hálózati forgalom statisztikai jellemzőinek hatékony kinyerését és integrálását gépi tanulási modellekbe. Például a [30] kutatás bemutatja, hogyan használható az NFStream DoS támadások detektálására.

Az innovatív megközelítések közül figyelemre méltó a gráfalapú modellek használata VPN szerverek viselkedésének azonosítására, amelyet a [31] dolgozat mutat

be, valamint a FlowPic módszer [32], amely az adatfolyamokat képpé alakítja, és képklasszifikáló modellekkel végzi az osztályozást. Ezek a technikák azt mutatják, hogy a hagyományos statisztikai jellemzők mellett az új, nem konvencionális módszerek is jelentős szerepet játszanak a VPN detekcióban.

Különösen érdekes két kutatás [33],[34], amelyek hatékonyan kombinálják a statisztikai jellemzőkön alapuló elemzést és a gépi tanulást. Az [33] tanulmány entrópia-ujjlenyomatokat és csomaghosszokat használ a VPN forgalom detekciójára, míg a [34] a csatorna protokoll karakterisztikáit elemzi. Mindkét módszer jelentős előnye, hogy a teljes adatfolyam helyett csak az első N csomagra támaszkodik, így gyors és hatékony osztályozást biztosít valós idejű alkalmazások számára is.

A jelen dolgozat ezen eredményekre épít, és célja, hogy a fent említett két módszert reprodukálja és összehasonlítsa egy új, saját fejlesztésű megközelítéssel. Az új módszer folyamatstatisztikákra alapozva alkalmaz gépi tanulási algoritmusokat a VPN forgalom detekciójára, azzal a céllal, hogy növelje az osztályozási pontosságot.

5 Módszertan

Az előbb tárgyalt módszerek mérlegelésével arra jutottam, hogy a dolgozat céljának a gépi tanulásos megközelítés felel meg leginkább, mivel jelenleg ebben az eljárásban látszik a legnagyobb potenciál.

A már korábban is említett két cikknek a saját magam által megvalósított feldolgozásukat fogom tárgyalni ebben a fejezetben. Először az első cikket, amely a mintavételi entrópia-ujjlenyomatokat (Sample Entropy Fingerprint: SEF) készíti el az egyes forgalmak első N csomagja alapján, valamint emellett felhasznál egy másik csomaghossz-sorozat (Packet Length Sequence: PLS) jellemzőt is [33]. Majd a második cikket, ami az úgynevezett csatorna protokoll jellemzőket (Tunneling Protocol Characteristics: TPC) használja fel a VPN forgalom kiszelektálására [34]. Ezeken felül az NFStream [35] keretrendszerrel készítettem egy harmadik lehetséges VPN forgalom detektáló megoldást. Itt az NFStream beépített jellemzőit használják fel a gépi tanulási modellek.

Mind a három esetben három különböző tanuló algoritmuson vizsgáltam az eljárások hatékonyságát: Döntési fa (Decision Tree: DT), Véletlen erdő (Random Forest: RF) és Neurális háló (Neural Network: NN). Azért ezeket a modelleket választottam, mert az előmunka kutatás során ezek tűntek általánosságban a leghatékonyabban teljesítő modelleknek, valamint mindegyik könnyen implementálható Python nyelven külső könyvtárak használatával. Ugyanaz az alap adathalmazt használtam mindhárom eset során.

5.1 Adathalmaz

5.1.1 Nyilvánosan elérhető adathalmazok

Kulcsfontosságú lépés volt az adathalmaz kiválasztása. A VPN forgalom osztályozásának reprodukálható kutatásait akadályozza, hogy kevés nyilvános VPN által titkosított forgalmi adat érhető el. Az ISCXVPN2016 adatállomány, amelyet a Canadian Institute for Cybersecurity (CIC) tett közzé [36], az első nyilvánosan elérhető VPN adathalmaz volt, és sok kutató használta gépi tanulási modellek betanítására és tesztelésére. Alaposabb vizsgálatok alapján azonban kiderült, hogy az ISCXVPN2016 adatállományban jelentős problémák vannak [37]. Egyrészt a VPN-nek jelölt csomagok

esetenként titkosítatlan adatokat tartalmaznak, például a Hypertext Markup Language (HTML) szövegeket, amelyek könnyen megtekinthetők. Másrészt egy VPN forgalomhoz tartozó csomagfájlok gyakran több forgalmat tartalmaznak, holott egyetlen folyamot várnánk a VPN kliens és szerver között.

További kutatómunka során talált adathalmaz a CIC-Darknet2020 a Kaggle weboldalon [39], amely szintén előszeretettel használt adathalmaz a hálózati forgalomanalízis során és tulajdonképpen két CIC által létrehozott adathalmazt fűz össze és dolgoz fel bizonyos mértékben, a korábban már említett ISCXVPN2016-ot és a ISCXTor2016-ot [40].

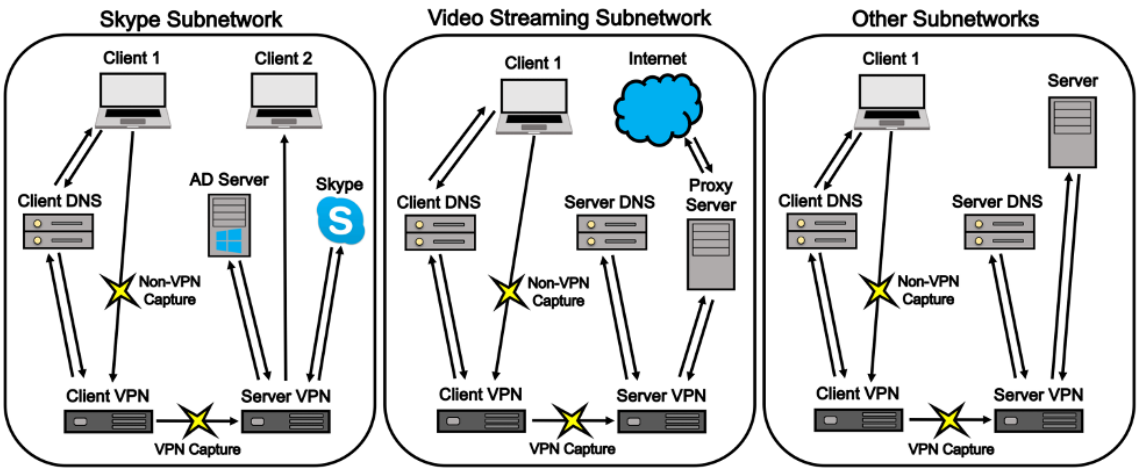
A University of South Bohemia által készített „Encrypted VPN Dataset" adathalmaz [41], amelynek nagy előnye, hogy különböző protokollokat alkalmazó VPN forgalmakat tartalmaz, és ezek megfelelően vannak elkülönítve, viszont ennek formátuma JavaScript Object Notation (JSON), amelyet közvetlen az NFStream nem tud feldolgozni, ezért szükséges lett volna átalakítani PCAP formátumúvá.

5.1.2 Választott adathalmaz

A szakdolgozathoz végül a Massachusetts Institute of Technology (MIT) által létrehozott „VPN/Non-VPN Network Application Traffic" (VNAT) adathalmazt használtam [38], amelyben külön válogatva találhatóak VPN és nem VPN hálózati forgalmat tartalmazó PCAP fájlok. Általános paraméterei, hogy 36,1 GB méretű, 33 711 darab folyammal és 272 órányi csomagfelvétellel rendelkezik, öt adatforgalmi kategóriában. Az adatállomány létrehozásához minden forgalmi kategóriához virtuális alhálózatokat hoztak létre, amelyek tartalmaznak egy klienst, egy DNS szervert, egy VPN klienst és egy VPN szervert. A VPN forgalmat a VPN kliens és szerver között rögzítették, míg a nem VPN forgalmat külön, az alkalmazási réteg és a VPN kliens között. Egyes forgalmakat manuálisan vettek fel, míg másokat szkriptek segítségével [37].

Forgalom kategóriák	*Alkalmazások*	*Méret (MB)*	*Adatfolyamok*
Videó adatfolyam	Vimeo, Netflix, Youtube	2626	1764
VoIP (Voice over IP)	Zoiper	103	617
Chat	Skype	185	1301
Irányítás és vezérlés	SSH, RDP	622	13 599
Fájl küldés	SFTP, RSYNC, SCP	32 601	16 430
Összesen	-	*36 137*	*33 711*

5-1. táblázat: Adathalmaz forgalmi kategóriáinak részletes adatai [37]



5-1. ábra: Adathalmazhoz használt hálózati konfigurációk és forgalom felvételi pontok [37]

Az adathalmaz pozitívuma, hogy jól szervezetten vannak a PCAP fájlok elkülönítve egymástól, valamint, hogy nagy mennyiségű adat található benne, ami elengedhetetlen egy gépi tanuláson alapuló probléma esetében. Ugyanakkor ez sem egy tökéletes állomány, mivel feltételezhetően csak egyféle VPN protokollal rögzítették a teljes adatállományt, mert nem említik a cikk során, hogy pontosan milyen VPN protokollt alkalmaztak. Továbbá nem fednek át teljes mértékben a VPN és nem VPN forgalmak. Ez alatt a „nonvpn_netflix_capture1.pcap” fájlt példának vehetjük, amelynek nincsen VPN-es párja, de a „nonvpn_netflix_capture2.pcap” és „vpn_netflix_capture2.pcap” fájlokból mindkettő létezik.

5.1.3 Adathalmaz feldolgozása Bash szkripttel

Ahhoz, hogy könnyedén fel lehessen dolgozni a PCAP fájlokat és ne egyesével kelljen egy Python szkriptet futtatni, létrehoztam egy bash szkriptet. A „pcap_gobbler.sh” szkript első argumentumnak bekér egy mappát, amelyben az összes feldolgozandó PCAP fájl található. Második argumentuma a kimeneti Comma-separated values (CSV) típusú fájl neve. Végül pedig harmadik argumentumnak a futtatandó Python szkriptet kell megadni. Röviden összefoglalva a „pcap_gobbler.sh” végigiterál a megadott mappában lévő PCAP fájlokon, amelyekre mind meghívja a megadott Python szkriptet, amely egy NFStreamer [42] implementációt valósít meg. A Pyhton szkript eredményét egy ideiglenes CSV fájlba tölti be, amelyhez sorba hozzáfűzi a többi feldolgozott CSV fájlt.

Alább látható hogyan lehet általánosságban meghívni a Bash szkriptet:

```
> ./pcap_gobbler.sh <pcap_mappa> <kimeneti_csv> <python_szkript>
```

A „pcap_gobbler.sh” pszeudokódja:

```
Ha nincs három argumentuma a hívásnak:
     Kilépés.

Ha nem található a Python szkript:
     Kilépés.

Ha nem található a mappa:
     Kilépés.

Ha már létezik a megadott kimeneti CSV fájl:
     Létező CSV törlése.

Iterálj végig a megadott mappában lévő PCAP fájlokon:

     Ha nincs PCAP fájl a mappában:
          Kilépés.
     Elágazás vége.

     Python szkript meghívása az aktuális PCAP fájllal.

     Ha létre lett hozva a „temp.csv”:

          Ha ez az első fájl:
               A „temp.csv” átnevezése a megadott kimeneti CSV fájl nevére.
          Egyébként:
               A „temp.csv” tartalmának hozzáfűzése a kimeneti CSV fájlhoz.
          Elágazás vége.

     Elágazás vége.
Ciklus vége.

A „temp.csv” fájl törlése.
```

5.2 Sample Entropy Fingerprint és Packet Length Sequence

5.2.1 NFPlugin

A cikk reprodukálásához [33] szükség volt az alap NFStream feldolgozás mellett, egy NFPlugin [42] létrehozására is, mivel maga a csomagok adat részének a tartalma az, amelyből a SEF-t ki lehet számolni. A csomagok IP fejléc nélküli mérete sem szerepel alapértelmezetten a jellemzők között, ami pedig a PLS paraméter. A plugin működése röviden, hogy egyrészt egy adott adatfolyamnak a megadott első N csomagjából kiveszi az IP fejléc nélküli nyers bájt típusú adatokat, másrészt az első N csomagnak az IP fejléc nélküli méretét is felveszi a végleges jellemzők közé.

Az NFStreamer-t a következő beállításokkal futtattam:

```
NFStreamer(source='PCAP fájl neve',
           decode_tunnels=False,
           n_dissections=20,
           splt_analysis=40,
           statistical_analysis=True,
           accounting_mode=1,
           n_meters=0,
           idle_timeout=120,
           active_timeout=180000,
           udps=PayloadFeatures(length=40)
          ).to_csv('temp.csv')
```

A „decode_tunnels” értékét hamisra állítottam, hogy a VPN-csatorna intakt maradjon a feldolgozás során. A „statistical_analysis” értéke igaz, hogy több statisztikai jellemző legyen elérhető számunkra. Az „accounting_mode”-t az IP rétegre állítotam. Az „active_timeout”-t olyan magas értékre állítottam be, hogy a hosszabb VPN forgalmak se legyenek szétvágva. Továbbá az „udps”, ahol saját plugin adható meg és az „splt_analysis” jellemzők esetében is 40 feldolgozandó csomagot állítottam be, mivel a cikk nem részletezi különösebben, hogy az első hány csomagot vizsgálták a feldolgozás során. A maradék jellemzők alapértelmezett értékek.

5.2.2 SEF kiszámításának algoritmusa

A SEF kiszámításának menete a következő: A folyamat során először a csomagok adattartalmát hexadecimális karakterláncként kell kódolni, majd egy pufferbe összefűzni, egészen addig amíg el nem fogynak a vizsgált csomagjai az adatfolyamnak vagy ideális esetben, ha eléri a kívánt puffer méretet. Majd egy fix ablakméret (ami a cikkhez hasonlóan 32 bájt) bájtonkénti ablakeltolásával az adott ablakra kiszámol egy entrópia

értéket. Az entrópia kiszámításánál a Shannon entrópia képlet egy módosított változatát alkalmazzuk az úgynevezett mintavételi entrópiát, amit a 5-1. egyenlet mutat.

$$\hat{H}_N^{MLE}(X) = -\sum_{i=1}^{m} \hat{p}\,(x_i) log_2 \hat{p}(x_i); \quad \hat{p}(x_i) = \frac{n_i}{N}$$

5-1. egyenlet: Mintavételi entrópia [33]

Ami pszeudokód formájában így néz ki:

```
Függvény mintavételi_entrópia(ablak):
     Egyedi értékek lista kiválasztása az ablakból.

     Az egyedi értékek számosságának kiszámítása az ablakan.

     Az egyes egyedi értékekre a valószínűség kiszámítása, a számosság
     értéke osztva az ablak elemeinek számával.

     Ha csupán egy egyedi elemből áll az ablak:
          Visszatérés 0 értékkel.
     Elágazás vége.

     A mintavételi entrópia kiszámítása a 5-1. egyenlet szerint, ahol p̂(x_i)
     a valószínűség értékének felel meg.

     Visszatérés az entrópia értékével.
Függvény vége.
```

Az algoritmus a kiszámított entrópiát ezután egy határértékkel hasonlítja össze, és számolja hány ablak lépte túl a határt. A végén, ha egy bizonyos százaléka az entrópia értékeknek átlépte a határértéket, amit én 99,5%-ra állítottam, akkor megjelöli VPN gyanúsnak az algoritmus az adott adatfolyamot. A puffer mérete ez esetben 256 bájt, mivel ez az ideális érték, ha az ablak mérete 32 bájt [43]. A határérték (θ) kiszámításához az entrópia-ujjlenyomat következő paramétereire van szükség: μ (középérték), σ (szórás) és t (hányszor vesszük a szórás értéket). Ezeknek a paramétereknek az értékeit is átvettem [43], amelyeket eredetileg a Monte-Carlo módszerrel számítottak ki.

$$\theta = \mu - t \times \sigma$$

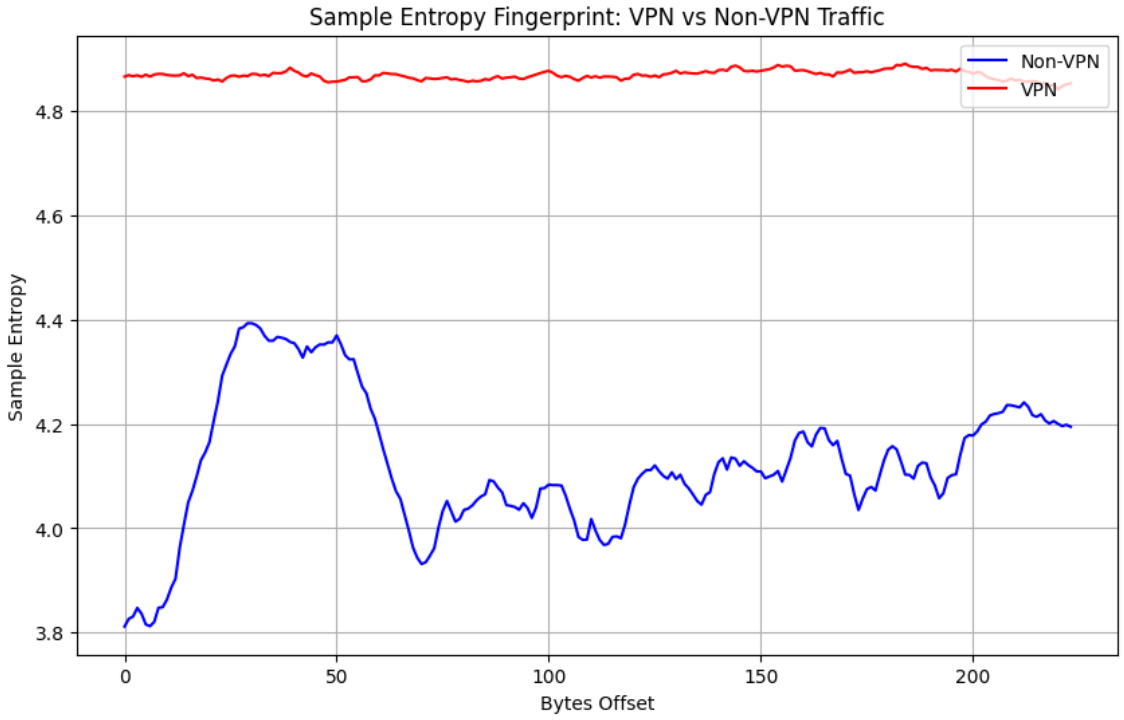
5-2. egyenlet: Entrópia határérték [33]

A tárgyalt esetben ezen paraméterek a következő értékeket veszik fel: μ = 4,8817, σ = 0,08134 és t = 3, amelyekből következik a képlet alapján, hogy θ = 4,6377.

A SEF kiszámításának algoritmusa pszeudokóddal:

```
Függvény sample_entropy_fingerprint(adat, puffer_méret, ablak_méret, mű,
szigma, t):
     Iterálj végig az egyes adatfolyamokon:

          Iterálj végig az egyes csomagokon az adatfolyamban:

               Ha a csomag tartalma nem üres:
                    Iterálj végig a csomag tartalmának bájtjain:
                         Alakítsd át a bájtot hexadecimálissá.
                         Fűzd hozzá a pufferhez a hexa értéket.
                    Ciklus vége.
               Elágazás vége.

               Ha a puffer mérete >= puffer_méret:
                    Vágd le a puffert puffer_méret nagyságúra.
                    Fejezd be a ciklust.
               Elágazás vége.

          Ciklus vége.

          A számosság értéke legyen nulla.

          Iterálj végig a pufferen az ablak eltologatásával:

               Vedd ki az ablak méretének és helyzetének megfelelő részt a
               pufferből.

               Számold ki az entrópiát a mintavételi_entrópia(ablak)
               függvénnyel.

               Add hozzá az entrópia értéket az ujjlenyomat listába.

               Ha az entrópia > (mű – t * szigma):
                    A számosságot növeld eggyel.
               Elágazás vége.

          Ciklus vége.

          Ha a számosság > (puffer_méret – ablak_méret + 1) * 0.995:
               Az adatfolyamot jelöld meg VPN-nek.
          Egyébként:
               Az adatfolyamot jelöld meg nem VPN-nek.
          Elágazás vége.

          Add hozzá az entrópia-ujjlenyomatot az entrópia-ujjlenyomatok
          listájába.

     Ciklus vége.

     Térj vissza az entrópia-ujjlenyomatok listájával és a jelölő listával.
Függvény vége.
```

Az eredeti cikk algoritmusától eltér ez a megvalósítás néhány helyen. Először is minden adatfolyamnak a puffer méretét azonos nagyságúra vágja a fenti algoritmus. Így kiküszöbölhetjük azt a problémát, hogy különböző méretű SEF értékek jöjjenek létre, kivéve, ha nincs elég csomag az adatfolyamban, hogy a puffert feltöltse, mert akkor kialakulhatnak rövidebb SEF-ek is. Eredetileg addig adta hozzá az egész csomagok adatait, amíg el nem érte a puffer_méret értéket, méghozzá ez lehetett több is, így kialakítva kisebb és nagyobb puffereket adatfolyamonként. Az algoritmus másik fejlesztését a VPN forgalmak megjelölésénél végeztem, mivel itt az eredeti algoritmusban az összes entrópia értéknek nagyobbnak kellett lennie, mint a határérték, tehát egy kiugró értéket sem tartalmazhatott. Az implementált algoritmus esetében viszont elég, ha csak az entrópia értékek 99,5%-a határérték feletti, javítva ezzel a felismerés pontosságán.



5-2. ábra: SEF értékek középértéke az ablak eltolása szerint VPN és nem VPN forgalmaknál [44]

A kiszámított értékek középértékét a 5-2. ábra szemlélteti, az ablakok eltolása szerint, az összes feldolgozott adatfolyamra. A gráfon jól láthatóan elkülönülnek a VPN és nem VPN forgalmak, és az átvett 4,6377-es határérték is jó vágóértéknek bizonyul. Mivel ezek csak középértékei egy nagy halmaz elemeinek, ezért ez a grafikon nem mutatja a nyilvánvalóan jelen lévő olyan forgalmakat, amelyek kiugróak és eltérnek az átlagtól.

modell	*accuracy*	*precision*	*recall*	*f1-score*
SEF algoritmus	0.999138	0.669072	0.905497	0.738539

5-2. táblázat: SEF kiszámítási algoritmusának klasszifikációs eredménye [44]

A 5-2. táblázatban megfigyelhetjük, hogy a SEF algoritmus során, az adatfolyamok megjelölése mennyire volt sikeres. Egészen meglepően jó eredményt tudott ez az egyszerű algoritmus generálni, annyi hátránnyal, hogy sok számolási kapacitást igényel a SEF értékek kiszámítása. A precízió (precision) alacsony értékét az arányaiban magas hamis pozitív értékek okozzák, tehát viszonylag sok nem VPN forgalom lett VPN-nek kategorizálva, így nagyjából 2,5x annyit jelölt VPN-nek, mint amennyi eredetileg van az adathalmazban. Ez azonban még nem elég jó ahhoz, hogy ténylegesen használni lehessen VPN detektálásra, mivel magas a hamisan detektáltak száma. A határérték növelésével lehetséges, hogy csökkenne a hamis pozitívok száma, viszont az akkor más mutató kárára.

Végül sikeresen kiszámítottuk a Sample Entropy Fingerprint értékeket az adatfolyamokra, ugyanakkor ez a jellemző még nem használható a modellek betanítására, mivel listát nem tudnak értelmezni, csupán szám típusú értékeket, ezért dimenziót kell csökkenteni.

5.2.3 Entrópia-ujjlenyomat jellemzők létrehozása

Ebben a fejezetben - részben szakítva az eredeti cikk megoldásával - nem csak azokra az adatfolyamokra futtattam az entrópia-ujjlenyomat jellemzők létrehozását, amelyeket nem VPN-nek jelölt meg az algoritmus, hanem az összes adatfolyamra, így az egész adathalmazt be lehetett adni a gépi tanulási modelleknek.

Az alap koncepciója a jellemző előállításának, hogy először csonkítja az ujjlenyomat listáját az adatfolyamoknak, majd a főkomponens-analízis (Principal Component Analysis: PCA) [45] lineáris dimenzióredukciós technikával csökkenti a dimenzióját a csonkított listának, így X számú jellemzőt kapva, amelyeket már fel tudnak a modellek dolgozni.

A tényleges algoritmus célja az, hogy a legjobb csonkítást megtalálja bizonyos intervallumok között, amellyel potenciálisan a legpontosabb modellt lehet felépíteni. Ezt úgy végzi el, hogy végigmegy 10-es csomagugrásokkal az első csomagtól az 50. csomagig az összes lehetséges csonkítási kombinációkon. Minden egyes csonkításra dimenziót csökkent PCA-val a variancia 95%-át megtartva, majd az újonnan létrehozott jellemzőkkel együtt lefuttat egy Random Forest klasszifikációt. A klasszifikációnak a pontossági (accuracy) értéke alapján értékeli az algoritmus a modell jóságát, elmenti a csonkítási paramétereket, ha az adta az eddigi legjobb értéket. Végül azt a csonkítást alkalmazza amelyik a leghatékonyabbnak bizonyult, és ezután végzi el a már említett dimenzió csökkentést és jellemzők felvételét az algoritmus.

modell	*accuracy*	*precision*	*recall*	*f1-score*
Decision Tree	0.999344	0.68169	0.735094	0.704964
Random Forest	0.999458	0.6873	0.588164	0.619864
Neural Network	0.999515	0.499757	0.5	0.499879

5-3. táblázat: SEF dimenziócsökkentése utáni gépi tanulási modellek klasszifikációs eredményei [44]

A sima SEF algoritmushoz képest a Decision Tree és a Random Forest esetében is minimálisan nőtt a precízió (precision) értéke, viszont az érzékenység (recall) paraméter látványosan visszaesett (5-3. táblázat), amely arra utal, hogy megnőtt a hamis negatívok aránya, tehát több lett az a VPN forgalom, amit sima forgalomként definiáltak a modellek.

5.2.4 PLS jellemzők kialakítása

A SEF hátránya, hogy nehezen tudja megkülönböztetni a VPN forgalmat a simán titkosított forgalmaktól, mivel ezeknél is ugyanúgy magas entrópia értékek keletkeznek a titkosítás miatt. Ennek elkerülése érdekében érdemes olyan jellemzőket is létrehozni, amelyek ezek megkülönböztetésében segítik a tanulási modelleket, mint esetünkben a PLS-t.

Hasonló algoritmussal dönti el, hogy az első hány darab csomagnak a csomaghosszát használja jellemzőkként, mint az előző fejezetben. Az első háromtól az

első 11 csomagig kipróbálja egy Random Forest osztályozás segítségével, hogy melyik esetben kapja a legjobb pontossági értéket.

Miután az algoritmus meghatározta, hogy melyik esetben a legpontosabb a modell, azután jellemzőkké is alakítja őket, majd futtatja a megfelelő modelleket.

modell	*accuracy*	*precision*	*recall*	*f1-score*
Decision Tree	0.999914	0.968721	0.941162	0.954524
Random Forest	1	1	1	1
Neural Network	0.999743	0.949886	0.764692	0.833269

5-4. táblázat: PLS jellemzőkkel tanított gépi tanulási modellek klasszifikációs eredményei [44]

A 5-4. táblázatot nézve az összes modell szinte minden paramétert tekintve jól tudta osztályozni az adatfolyamokat, még annak ellenére is, hogy csak a PLS jellemzők voltak elérhetőek a számukra. A Random Forest modell bizonyult a legjobbnak, amely teljes mértékben hiba nélkül tudott osztályozni.

5.2.5 SEF és PLS jellemzőkkel való gépi tanulás

Utolsó lépésként az maradt, hogy a kialakított jellemzők felhasználásával betanítsam a választott modelleket, amelyeknek tudom vizsgálni a hatékonyságát. Jól látható a 5-5. táblázat alapján, hogy mindegyik modell- magas százalékban sikeresen el tudta különíteni a megadott jellemzőket kombinálva a VPN forgalmakat a hétköznapi forgalmaktól. A Random Forest modell bizonyult a legjobbnak, amely továbbra is hiba nélkül osztályozott.

modell	*accuracy*	*precision*	*recall*	*f1-score*
Decision Tree	0.999914	0.94443	0.97056	0.957121
Random Forest	1	1	1	1
Neural Network	0.999829	0.933276	0.882324	0.906207

5-5. táblázat: SEF és PLS jellemzőkkel tanított gépi tanulási modellek klasszifikációs eredményei [44]

5.3 Tunneling Protocol Characteristics

Ennek a cikknek [34] a feldolgozásához is szükség volt egy egyedi NFPlugin kialakítására, hogy az említett jellemzőket létre tudja hozni. A plugin feladata összefoglalva, hogy minden egyes adatfolyam végén statisztikai jellemzőket alakítson ki az adatcsomagok mérete, illetve a csomagok közötti érkezési idők (Packet Inter-Arrival Time: PIAT) alapján. Az összes felhasznált és kiszámított jellemzőt a 5-6. táblázat tartalmazza.

JELLEMZŐ TÍPUSA	STATISZTIKAI JELLEMZŐK
Halmozott összegek	csomagszám, csomag tartalom mérete, csomag tartalom aránya
Csomag tartalmának mérete	maximum, középérték, nem nulla minimum
Csomagok közötti érkezési idők	maximum, középérték, nem nulla minimum
Standard entrópia	csomag tartalom méret, csomag érkezési idők

5-6. táblázat: Tunneling Protocol Characteristics jellemzők [34]

Ami még fontos része a pluginnak az a standard entrópia számoló függvény, amely a 5-3. egyenlet alapján számolja ki az adatcsomag mérethez, illetve a PIAT-hoz az entrópia értékeket.

$$H_N(X) = -\frac{\sum_{i=1}^{N} p(x_i) log_2 p(x_i)}{log_2 N}$$

5-3. egyenlet: Standard entrópia [34]

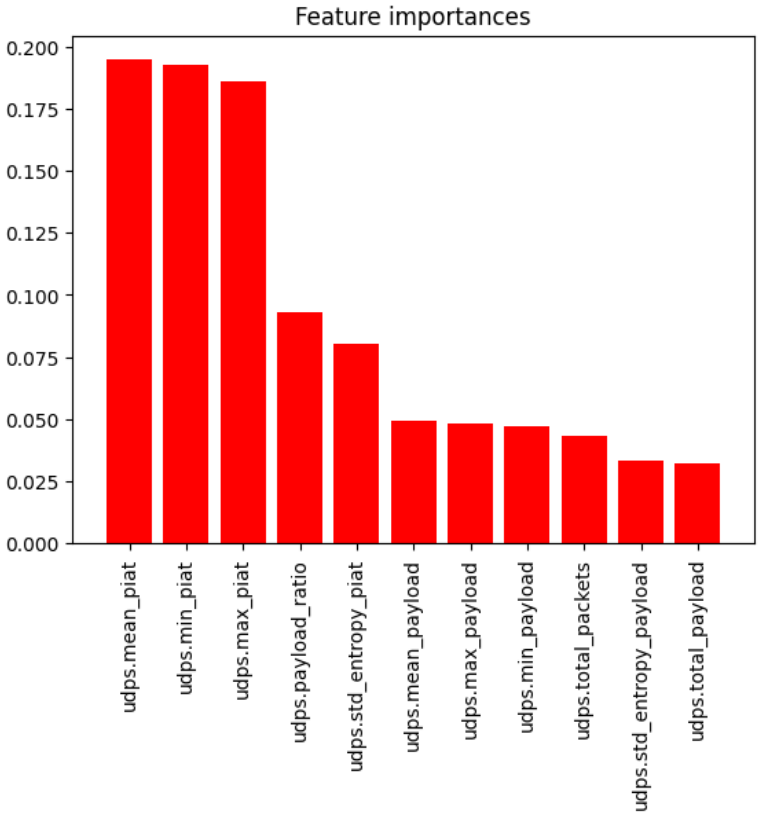
A standard entrópiát kiszámoló függvény pszeudokódja a következő:

```
Függvény standard_entrópia(adat):
     Ha az adat lista mérete == 0:
          Kilépés None értékkel.
     Egyébként, ha az adat lista mérete == 1:
          Kilépés 0 értékkel.
     Elágazás vége.

     Az egyedi értékek számosságának kiszámítása az adat listából.

     Az egyes egyedi értékekre a valószínűség kiszámítása, a számosság
     értéke osztva az adat lista elemeinek számával.

     A mintavételi entrópia kiszámítása a 5-3. egyenlet szerint, ahol p(x_i)
     a valószínűség értékének felel meg.

     Visszatérés az entrópia értékével.
Függvény vége.
```

Az NFStreamer-t a következő beállításokkal futtattam:

```
NFStreamer(source='PCAP fájl neve',
           decode_tunnels=False,
           n_dissections=20,
           splt_analysis=32,
           statistical_analysis=True,
           accounting_mode=1,
           n_meters=0,
           idle_timeout=15,
           active_timeout=180000,
           udps=PayloadFeatures(length=32)
          ).to_csv('temp.csv')
```

Az NFStreamer beállításai nagyrészt ugyanazok, mint a SEF+PLS esetében, viszont ebben az esetben a cikk szerinti első 32 csomaggal számolja ki a statisztikai jellemzőket. Valamint az „idle_timeout" értéke 15, tehát ha 15 másodpercig nem érkezik meg a következő csomag, akkor passzivitás miatt vége az adatfolyamnak és a következő csomag már egy új adatfolyamot fog elkezdeni.



5-3. ábra: TPC jellemzőinek Extra-Trees szerinti fontossági sorrendje [44]

Extra-Trees modell segítségével megvizsgálva a jellemzők fontossági értékeit, azt látjuk a 5-3. ábra alapján, hogy a PIAT jellemzők majdnem négyszer annyira fontosak, mint a csomag tartalmát feldolgozó egyéb jellemzők.

modell	*accuracy*	*precision*	*recall*	*f1-score*
Decision Tree	0.999861	0.973404	0.99993	0.986304
Random Forest	1	1	1	1
Neural Network	0.999916	0.999958	0.983146	0.991408

5-7. táblázat: TPC jellemzőkkel tanított gépi tanulási modellek klasszifikációs eredményei [44]

Végezetül a gépi tanulási modelleket a létrehozott jellemzők alapján futtatva a 5-7. táblázatban látható eredményeket kaptam. Mind a három modell nagyon jó eredményt ért el minden tekintetben, a legjobbnak itt is a Random Forest bizonyult, amely tökéletes klasszifikációt végzett.

5.4 NFStream statisztikai jellemzői

Az általam létrehozott megoldásban arra a kérdésre kerestem a választ, hogy mennyire tudnának jól teljesíteni azok a modellek, amelyek csupán az NFStream statisztikai jellemzői alapján lennének betanítva. Ha csak annyit csinálnánk, hogy feldolgozzuk a PCAP fájlokat NFStream-el és az adatbázis lényegtelen oszlopai eldobása után vannak betanítva a modellek akkor természetesen nagyon egyszerűen el tudják különíteni a forgalmakat, mivel a VPN-re az a jellemző és ez a használt adatbázisra is igaz, hogy csupán egy nagyon hosszú adatfolyam áll rendelkezésre. Mivel a VPN csatornázva kommunikál, ezért az NFStream nem tudja úgy elkülöníteni a csatornán belüli forgalmakat, ahogy azt a sima forgalmak esetén képes.

Az előző két megoldás jó potenciális felhasználása, hogy már az első N csomag alapján el tudják különíteni a forgalmakat, így téve alkalmassá élő forgalomdetektálásra a modelleket. Ahhoz, hogy erre az én megoldásom is képes legyen, szükség van itt is egy NFPlugin létrehozására, amely ez esetben csupán csak annyit fog csinálni, hogy ha a beállításokban megadott N csomagot eléri a vizsgált adatfolyam, akkor azt elvágja és a következő csomag már új adatfolyamba fog bekerülni. Az adatok feldolgozása itt még nem ér véget, mivel nekünk nem kell az így létrejött összes folyam, hanem minden felaprított folyamnak csak az első darabja.

Elvégeztem kétféle beállítással is a megoldást, egyszer 32 csomag és egyszer 40 csomag hosszúságú adatfolyamokat vágva. Így is hozzájárulva, ahhoz, hogy a legjobban össze lehessen hasonlítani az előző két megoldással.

Az NFStreamer beállításai 32 csomag esetében a következőek:

```
NFStreamer(source='PCAP fájl neve',
           decode_tunnels=False,
           n_dissections=20,
           statistical_analysis=True,
           accounting_mode=1,
           n_meters=1,
           idle_timeout=120,
           active_timeout=180000,
           splt_analysis=32,
           udps=FlowSlicer(limit=32)
          ).to_csv('temp.csv')
```

Az „n_meters” paraméterrel azt lehet szabályozni, hogy hány darab párhuzamos szálon fusson a feldolgozás. Ezt egyre állítottam az alapértelmezett nulláról, mivel így garantáljuk, hogy jó sorrendben lesznek az egyes PCAP fájlok csomagjai feldolgozva. Hátránya, természetesen, hogy ezáltal lassabb a feldolgozás. A FlowSlicer az NFPlugin, amely végzi az adatfolyamok vágását. A többi paraméter hasonló az eddigiekhez.

A VPN forgalmak esetében könnyebb dolgunk van kiszűrni az eredeti folyamok első darabját, mivel a FlowSlicer NFPlugin egyedi -1-es értékkel címkézte meg az „expiration_id” oszlopban azokat a folyamokat, amelyeket ő vágott el.

Így a következő algoritmussal ki tudjuk szűrni a szükséges VPN folyamokat:

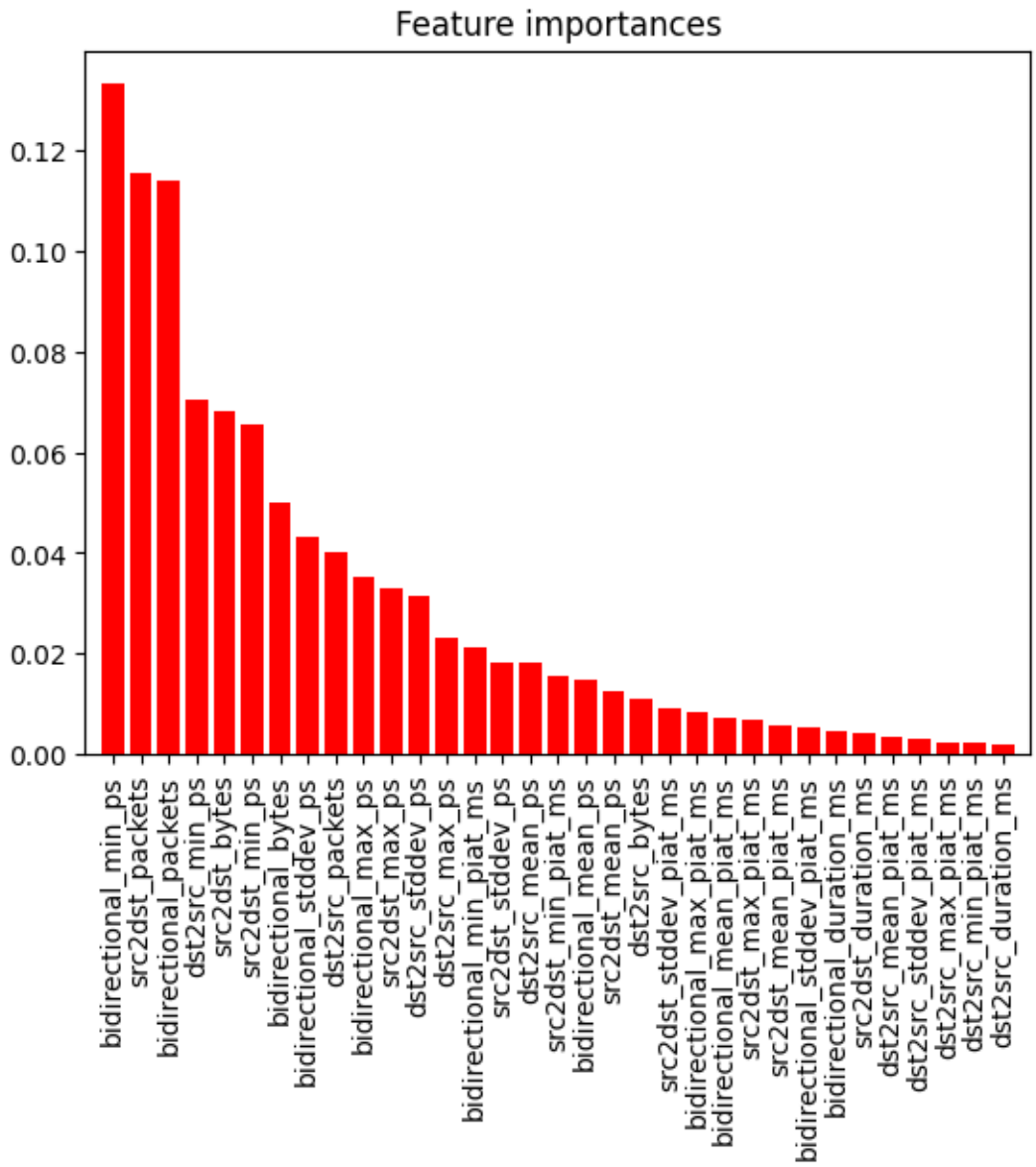
```
Iterálj végig az összes VPN adatfolyamon a másodiktól kezdve:

     Ha az ezt megelőző adatfolyam „expiration_id” értéke == -1:
          Jelöld meg az aktuális adatfolyamot.
     Elágazás vége.

Ciklus vége.

Válaszd ki az első és a többi megjelölt adatfolyamot.
```

A nem VPN forgalmak kiszűrése már egy fokkal nehezebb, mivel itt nem feltétlen egymás után követik az egy adott adatfolyamhoz tartozó csomagok egymást. Ez esetben általában több kommunikáció zajlik egymással párhuzamosan, ezért a forrás és cél portszámok, valamint időbeli rendezés alapján szűrjük a forgalmakat.

A használt algoritmus pszeudokódja a következő:

```
Függvény port_tuple_letrehozasa(folyam):
     Hozz létre egy listát, amelybe helyezd bele a megadott folyam forrás
     és cél portjait.

     Rendezd a listát sorrendbe.

     Alakítsd át a listát egy n-essé, majd vedd fel a folyam egy új oszlop
     paramétereként „port_tuple” néven.

     Add vissza a megváltoztatott folyamot.
Függvény vége.

Iterálj végig az összes nem VPN folyamon:
     Hívd meg minden egyes folyamra a port_tuple_letrehozasa(folyam)
     függvényt.
Ciklus vége.

Rendezd a nem VPN folyamokat első sorban az új „port_tuple” paraméter
alapján, majd pedig azon belül időrendben.

Iterálj végig az összes nem VPN adatfolyamon a másodiktól kezdve:
     Vond ki az aktuális folyam kezdetének idejéből az előző csomag végének
     idejét, hogy megkapd a két folyam között eltelt időkülönbséget.

     Ha a „port_tuple” paraméter a mostani folyamban nem egyezik az előző
     folyamban lévővel,
     Vagy ha az időkülönbség meghaladja a 15 másodpercet,
     Vagy ha az időkülönbség értéke < 0:
          Jelöld meg az aktuális adatfolyamot.
     Elágazás vége.
Ciklus vége.

Válaszd ki az első és a többi megjelölt adatfolyamot.
```

Miután a szűrés megtörtént minden adatfolyamra, már csak el kell dobálni az NFStream által létrehozott, de nem szükséges oszlopokat és a maradék jellemzőkkel betanítani a modelleket.



5-4. ábra: NFStream statisztikai jellemzőinek Extra-Trees szerinti fontossági sorrendje [44]

Extra-Trees modell segítségével megvizsgálva a maradék jellemzők fontossági értékeit, a 5-4. ábra alapján látható, hogy főként a csomagméretre és csomagszámra vonatkozó jellemzők dominálnak, azon belül is a kétirányú minimális csomagméret, a forrás-cél irányú csomagszám és a kétirányú csomagszám. Érdekes, hogy míg TPC esetében inkább a PIAT jellemzők voltak a meghatározóak, addig ebben az esetben ezek már sokkal kevésbé fontosak, mint más csomagméretre és csomagszámra vonatkozó jellemzők.

implementáció	*modell*	*accuracy*	*precision*	*recall*	*f1-score*
32 csomag	Decision Tree	0.999972	0.993421	0.999986	0.996682
	Random Forest	0.999972	0.999986	0.993333	0.996637
	Neural Network	1	1	1	1
40 csomag	Decision Tree	0.999943	0.993407	0.993407	0.993407
	Random Forest	1	1	1	1
	Neural Network	0.999943	0.999972	0.986842	0.993319

5-8. táblázat: NFStream jellemzőkkel tanított gépi tanulási modellek klasszifikációs eredményei [44]

A 5-8. táblázatba foglalt klasszifikációs eredmények alapján az látszik, hogy mind 32, mind 40 csomag esetében az összes modell sikeresen rátanult a jellemzőkre. 32 csomag esetében a Neurális háló, míg 40 csomagnál a Random Forest modellnek sikerült 100%-os teljesítményt nyújtania és egyben a legjobb modelleknek lenniük.

5.5 Összehasonlító elemzés

Az előzőleg már tárgyalt három gépi tanulásos megoldást hasonlítom össze, pontosság, adatfüggőség és számítási komplexitás szerint, melynek célja, hogy azonosítsam az egyes módszerek erősségeit és gyengeségeit. Az eredmények összesítését a 5-9. táblázat tartalmazza.

5.5.1 Pontosság

Mind a három megoldás esetében volt legalább egy modell, amely 100%-os pontossággal tudta azonosítani egy adott forgalomról, hogy VPN-e, ezért létjogosultsága és alkalmazhatósága biztos, hogy mindegyiknek van. A SEF+PLS, TPC és a 40 csomagos NFStream eljárások esetében a Véletlen erdő modell és a 32 csomagos NFStream-nél a Neurális háló volt képes erre.

Ha az egyes modellek pontosságát vetjük össze, tehát például, hogy a három Döntési fa modell közül melyik volt a leghatékonyabb, akkor kicsit más a helyzet. Míg a Döntési fa és Véletlen erdő modelleket összehasonlítva igazából nem érdemes

különbséget keresni, mivel mindegyik nagyjából egyformán jól teljesített, addig a Neurális háló esetében a SEF+PLS eljárásnak már számottevően rosszabbak a precíziós és érzékenységi értékei, a másik kettő eljáráshoz képest.

A SEF+PLS módszer esetében az a kérdés is felmerül, hogyha már csak a PLS jellemzőknél is ilyen jól teljesítenek a modellek, akkor van-e értelme még sok plusz számítási kapacitásért cserébe valamennyivel jobb eredményeket kapni. A Döntési fa és Véletlen erdő modellek esetében nem biztos, mivel Döntési fa esetében csupán néhány százalékos javulást tudunk elérni, viszont a Neurális háló esetében már lehetséges, hiszen ott akár több mint 10%-os érzékenységi javulást lehet megfigyelni.

5.5.2 Adatfüggőség

A TPC és NFStream eljárások esetében hasonlóak az adatigények, mivel az adatfolyamokból számított statisztikai jellemzőkre alapoznak főként, amelyekre akár a csomagok fejléc adatai is elegek lehetnek. A SEF+PLS módszer ebben a tekintetben is más, mert itt szükség van a Sample Entropy Fingerprint számolásánál a csomagok tartalmára is, ami nem biztos, hogy elérhető információ és jelentősen növeli az adathalmaz méretét.

Utólagos vizsgálatra alkalmas bármelyik módszer, ezt végeztem el a szakdolgozat során is, tehát ha van egy már korábbi adatforgalom, akkor bármelyik megoldással jó eredmények érhetőek el. Elméletben továbbá akár élő VPN detektálásra is potenciálisan megfelelőek lehetnek a fent kidolgozott eljárások, mivel egyiknek sincs szüksége a teljes folyamra, hanem elég csupán nekik az első N csomag feldolgozása, hogy osztályozásra legyenek képesek.

5.5.3 Számítási komplexitás

A SEF+PLS módszer számítási komplexitására legnagyobb hatással a SEF algoritmus van, amely O(n·m·w), ahol ’n’ az adatfolyamok száma, ’m’ a csomagok száma az adatfolyamon belül és ’w’ az, hogy az ablakolás során hány ablaknak számítja ki az entrópiáját. A TPC és NFStream eljárások számítási komplexitásának meghatározó része az adatfeldolgozás során történik. A komplexitást O(x·y) -ként tudjuk felírni, ahol ’x’ a PCAP fájlok számát, és ’y’ az egy fájlban lévő csomagok számát mutatja. Így, mivel az adatfeldolgozás számítási komplexitása a SEF+PLS megoldásban is megjelenik

ugyanígy, és még ennél nagyobb komplexitású rész is van, ezért ez tekinthető ebből a szempontból a gyengébb eljárásnak a három közül.

implementáció	*modell*	*accuracy*	*precision*	*recall*	*f1-score*	*számítási komplexitás*
SEF algoritmus	-	0.999138	0.669072	0.905497	0.738539	O(n·m·w)
SEF	DT	0.999344	0.68169	0.735094	0.704964	O(n·d·logn)
	RF	0.999458	0.6873	0.588164	0.619864	O(t·n·m·logm)
	NN	0.999515	0.499757	0.5	0.499879	O(e·n·w)
PLS	DT	0.999914	0.968721	0.941162	0.954524	O(n·d·logn)
	RF	1	1	1	1	O(t·n·m·logm)
	NN	0.999743	0.949886	0.764692	0.833269	O(e·n·w)
SEF+PLS	DT	0.999914	0.94443	0.97056	0.957121	O(n·d·logn)
	RF	1	1	1	1	O(t·n·m·logm)
	NN	0.999829	0.933276	0.882324	0.906207	O(e·n·w)
TPC	DT	0.999861	0.973404	0.99993	0.986304	O(n·d·logn)
	RF	1	1	1	1	O(t·n·m·logm)
	NN	0.999916	0.999958	0.983146	0.991408	O(e·n·w)
NFStream (32)	DT	0.999972	0.993421	0.999986	0.996682	O(n·d·logn)
	RF	0.999972	0.999986	0.993333	0.996637	O(t·n·m·logm)
	NN	1	1	1	1	O(e·n·w)
NFStream (40)	DT	0.999943	0.993407	0.993407	0.993407	O(n·d·logn)
	RF	1	1	1	1	O(t·n·m·logm)
	NN	0.999943	0.999972	0.986842	0.993319	O(e·n·w)

5-9. táblázat: Összehasonlító táblázat

6 Összegzés

A szakdolgozat során bemutattam a VPN technológiát, annak széles körű alkalmazási lehetőségeit, valamint a különböző protokoll fajtáinak működését. Továbbá részletesen elemeztem a VPN észlelési technikákat, beleértve az IP elemzést, a mély csomagvizsgálatot és a gépi tanulási módszereket, miközben bemutattam a kapcsolódó korábbi kutatások legfontosabb eredményeit. Az adathalmaz tulajdonságainak feltárása, valamint a három különböző VPN észlelési eljárás részletes bemutatása lehetővé tette az eredmények összehasonlítását, amely során megállapítottam az egyes módszerek előnyeit és korlátait.

A vizsgált három VPN forgalomosztályozási megoldás mindegyike képes volt bizonyítani alkalmazhatóságát, hiszen legalább egy modelljük 100%-os pontosságot ért el a VPN forgalom detektálásában. Ez azt mutatja, hogy az eljárások alapvetően jól működnek és relevánsak. Az összehasonlítás ugyanakkor rávilágított arra, hogy a könnyű alkalmazhatóság és a kisebb számítási igények miatt a TPC és NFStream eljárások praktikusabb alternatívát jelenthetnek, míg a SEF+PLS inkább olyan helyzetekben lehet hasznos, ahol a számítási erőforrások nem jelentenek korlátozást. Az eredmények tehát fontos iránymutatást nyújthatnak a további kutatások és gyakorlati alkalmazások számára.

A dolgozat még több szempontból fejlesztési lehetőséget kínál, hiszen az adatbázisról nem tudni, hogy milyen VPN protokoll vagy protokollok felhasználásával lett létrehozva. A lehetőségek egyike az, hogy olyan adatbázison is értékelve legyenek a használt módszerek, amelyben többféle ismert VPN protokoll is megtalálható. Szintén az adatbázis gyengesége, hogy a sima hálózati forgalmak darabszámához képest, nagyon kevés darabszámú VPN forgalom található meg benne, melyek arányai javítandóak. Az NFStream esetében az adatok feldolgozásának idején még szinte biztosan lehetne javítani, hogy ne utólag kelljen kiszűrni az elvágott adatfolyamokat, hanem már a feldolgozás során eleve ne kerüljenek ezek bele az adathalmazba.

Egy lehetséges továbbfejlesztési lehetősége a munkának, hogyha már a VPN forgalmat ilyen jól el tudják különíteni a modellek, akkor a VPN-en belüli különböző típusú forgalmak klasszifikációjára is potenciálisan képesek lehetnek akár. Egy másik út, hogy a különböző típusú VPN protokollok osztályozása lehetséges-e ezekkel vagy hasonló modellekkel. Még további fejlesztési lehetőség, hogy implementáljuk őket élő VPN forgalom detektálására. Végül létre lehetne hozni egy jobb minőségű, részletesebb, akár publikusan is elérhető adathalmazt, amelyen még jobb modelleket lehet felépíteni.

Ezen lehetőségek mentén szeretném a jövőben továbbfejleszteni a munkámat, mivel egyre aktuálisabb kutatási terület a VPN forgalmak észlelése, a dolgozat írása közben megszavazott új Magyarországi törvény [46] miatt is, ami az internetszolgáltatókra hárítaná a feladatot, hogy kiskorúak elől elérhetetlenné tegyenek bizonyos felnőtt tartalmú oldalakat.

A reprodukálhatóság fényében a munkámhoz tartozó összes digitális forrás a [44] tárhelyen érhető el.

Köszönetnyilvánítás

Köszönöm szépen konzulensemnek, Dr. Pekár Adriánnak a szakdolgozat elkészítésében nyújtott kimagasló és rengeteg segítségét, illetve mindazon személyeknek is köszönöm, akik akár közvetlen, akár közvetve segítették ennek a dolgozatnak a létrejöttét!

Irodalomjegyzék

[1] Vox Media: *WhatsApp fined $267 million for breaching EU privacy law* https://www.theverge.com/2021/9/2/22653747/whatsapp-fine-amount-europe-gdpr-privacy (2024. dec.)

[2] Cable News Network: *Google hit with lawsuit alleging it stole data from millions of users to train its AI tools* https://edition.cnn.com/2023/07/11/tech/google-ai-lawsuit/index.html (2024. dec.)

[3] Zhang, Z., Zhang, Y. Q., Chu, X., & Li, B.: *An overview of virtual private network (VPN): IP VPN and optical VPN*, Photonic network communications, Vol. 7, 2004, pp. 213-225, 10.1023/B:PNET.0000026887.35638.ce

[4] Clayton, Richard, Steven J. Murdoch, and Robert NM Watson: *Ignoring the great firewall of china*, International workshop on privacy enhancing technologies, Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 20-30, 10.1007/11957454_2

[5] Khan, James.: *A Study in Protocol Obfuscation Techniques and their Effectiveness*. GRIN Verlag, 2018, 10.13140/RG.2.2.14167.32165

[6] Akinsanya, M. O., Ekechi, C. C., & Okeke, C. D.: *Virtual private networks (vpn): a conceptual review of security protocols and their application in modern networks*, Engineering Science & Technology Journal, Vol. 5(4), 2024, pp. 1452-1472, 10.51594/estj.v5i4.1076

[7] Nord Security: *What is a VPN?*, https://nordvpn.com/what-is-a-vpn/ (2024. dec.)

[8] Crist, E. F., & Keijser, J. J.: *Mastering OpenVPN*, ISBN 978-1-78355-313-6, Packt Publishing Ltd., 2015

[9] Abdulazeez, A., Salim, B., Zeebaree, D., & Doghramachi, D.: *Comparison of VPN Protocols at Network Layer Focusing on Wire Guard Protocol*, International Journal of Interactive Mobile Technologies (iJIM), Vol. 14(18), 2020, pp. 157–177, 10.3991/ijim.v14i18.16507

[10] Hamzeh, K., Pall, G., Verthein, W., Taarud, J., Little, W., and G. Zorn: *Point-to-Point Tunneling Protocol (PPTP)*, RFC 2637, 1999, 10.17487/RFC2637

[11] Schneier, B., & Mudge.: *Cryptanalysis of Microsoft's point-to-point tunneling protocol (PPTP)*, 5th ACM Conference on Computer and Communications Security, 1998, pp. 132-141

[12] Ferguson, N., & Schneier, B.: *A cryptographic evaluation of IPsec*, 1999

[13] Satapathy, A., & Livingston, J.: *A Comprehensive Survey on SSL/TLS and their Vulnerabilities*, International Journal of Computer Applications, Vol. 153(5), 2016, pp. 31-38

[14] Kotuliak, I., Rybár, P., & Trúchly, P.: *Performance comparison of IPsec and TLS based VPN technologies*, 9th International Conference on Emerging eLearning Technologies and Applications (ICETA), 2011, pp. 217-221, 10.1109/ICETA.2011.6112567

[15] Skendzic, A., and B. Kovacic.: *Open source system OpenVPN in a function of Virtual Private Network*, IOP Conference Series: Materials Science and Engineering, Vol. 200, No. 1. IOP Publishing, 2017, 10.1088/1757-899X/200/1/012065

[16] Donenfeld, J. A.: *WireGuard: Next Generation Kernel Network Tunnel*, NDSS, 2017, pp. 1-12

[17] Dan, O., Parikh, V., & Davison, B. D.: *IP geolocation through reverse DNS*, ACM Transactions on Internet Technology (TOIT), Vol. 22(1), 2021, pp. 1-29, 10.1145/3457611

[18] El-Maghraby, R. T., Abd Elazim, N. M., & Bahaa-Eldin, A. M.: *A survey on deep packet inspection*, 12th International Conference on Computer Engineering and Systems (ICCES), 2017, pp. 188-197, 10.1109/ICCES.2017.8275301

[19] Deri, L., Martinelli, M., Bujlow, T., & Cardigliano, A.: *ndpi: Open-source high-speed deep packet inspection*, International Wireless Communications and Mobile Computing Conference (IWCMC), 2014, pp. 617-622, 10.1109/IWCMC.2014.6906427

[20] Mahesh, B.: *Machine learning algorithms-a review*, International Journal of Science and Research (IJSR), Vol. 9(1), 2020, pp. 381-386

[21] Wang, C., Zhou, X., You, F., & Chen, H.: *Design of P2P traffic identification based on DPI and DFI*, International Symposium on Computer Network and Multimedia Technology, 2009, pp. 1-4, 10.1109/CNMT.2009.5374577

[22] Dharmapurikar, S., Krishnamurthy, P., Sproull, T., & Lockwood, J.: *Deep packet inspection using parallel bloom filters*, Symposium on High Performance Interconnects, 2003, pp. 44-51, 10.1109/CONECT.2003.1231477

[23] Kumar, S., Dharmapurikar, S., Yu, F., Crowley, P., & Turner, J.: *Algorithms to accelerate multiple regular expressions matching for deep packet inspection*, ACM SIGCOMM computer communication review, Vol. 36(4), 2006, pp. 339-350 10.1145/1151659.1159952

[24] Liu, Y., Li, W., & Li, Y.: *Network traffic classification using k-means clustering*, Second international multi-symposiums on computer and computational sciences (IMSCCS), 2007, pp. 360-365, 10.1109/IMSCCS.2007.52

[25] Soysal, M., & Schmidt, E. G.: *Machine learning algorithms for accurate flow-based network traffic classification: Evaluation and comparison*, Performance Evaluation, Vol. 67(6), 2010, pp. 451-467, 10.1016/j.peva.2010.01.001

[26] Miller, S., Curran, K., & Lunney, T.: *Detection of virtual private network traffic using machine learning*, International Journal of Wireless Networks and Broadband Technologies (IJWNBT), Vol. 9(2), 2020, pp. 60-80, 10.4018/IJWNBT.2020070104

[27] Miller, S., Curran, K., & Lunney, T.: *Multilayer perceptron neural network for detection of encrypted VPN network traffic*, International conference on cyber situational awareness, data analytics and assessment (Cyber SA), 2018, pp. 1-8, 10.1109/CyberSA.2018.8551395

[28] Draper-Gil, G., Lashkari, A. H., Mamun, M. S. I., & Ghorbani, A. A.: *Characterization of encrypted and vpn traffic using time-related*, Proceedings of the 2nd international conference on information systems security and privacy (ICISSP), 2016, pp. 407-414

[29] Aouini, Z., & Pekar, A.: *NFStream: A flexible network data analysis framework*, Computer Networks, Vol. 204, 2022, 10.1016/j.comnet.2021.108719

[30] Chovanec, M., Hasin, M., Havrilla, M., & Chovancová, E.: *Detection of HTTP DDoS Attacks Using NFStream and TensorFlow*, Applied Sciences, Vol. 13(11), 2023, 10.3390/app13116671

[31] Wang, C., Yin, J., Li, Z., Xu, H., Zhang, Z., & Liu, Q.: *Identifying VPN Servers through Graph-Represented Behaviors*, Proceedings of the ACM on Web Conference 2024, 2024, pp. 1790-1799, 10.1145/3589334.3645552

[32] Shapira, T., & Shavitt, Y.: Flowpic: *Encrypted internet traffic classification is as easy as image recognition*, IEEE INFOCOM 2019-IEEE conference on computer communications workshops (INFOCOM WKSHPS), 2019, pp. 680-687, 10.1109/INFCOMW.2019.8845315

[33] Gao, P., Li, G., Shi, Y., & Wang, Y.: *VPN traffic classification based on payload length sequence*, International Conference on Networking and Network Applications (NaNA), 2020, pp. 241-247, 10.1109/NaNA51271.2020.00048

[34] Li, Y., Wang, F., & Chen, S.: *VPN Traffic Identification Based on Tunneling Protocol Characteristics*, IEEE 5th International Conference on Computer and Communication Engineering Technology (CCET), 2022, pp. 150-156, 10.1109/CCET55412.2022.9906397

[35] GitHub: *NFStream*, https://github.com/nfstream/nfstream (2024. dec.)

[36] University of New Brunswick: *VPN-nonVPN dataset (ISCXVPN2016)*, https://www.unb.ca/cic/datasets/vpn.html (2024. dec.)

[37] Jorgensen, S., Holodnak, J., Dempsey, J., de Souza, K., Raghunath, A., Rivet, V., ... & Wollaber, A.: *Extensible machine learning for encrypted network traffic application labeling via uncertainty quantification*, IEEE Transactions on Artificial Intelligence, Vol. 5(1), 2023, pp. 420-433, 10.1109/TAI.2023.3244168

[38] Lincoln Laboratory, Massachusetts Institute of Technology: *VPN/Non-VPN Network Application Traffic Dataset (VNAT)*, https://www.ll.mit.edu/r-d/datasets/vpnnonvpn-network-application-traffic-dataset-vnat (2024. dec.)

[39] Kaggle: *CIC-Darknet2020*, https://www.kaggle.com/datasets/dhoogla/cicdarknet2020 (2024. dec.)

[40] University of New Brunswick: *Tor-nonTor dataset (ISCXTor2016)*, https://www.unb.ca/cic/datasets/tor.html (2024. dec.)

[41] CERN: *Encrypted VPN Dataset*, https://doi.org/10.5281/zenodo.7292908 (2024. dec.)

[42] NFStream: *APIs Documentation*, https://www.nfstream.org/docs/api (2024. dec.)

[43] Luo, S., Seideman, J. D., & Dietrich, S.: *Fingerprinting cryptographic protocols with key exchange using an entropy measure*, IEEE Security and Privacy Workshops (SPW), 2018, pp. 170-179, 10.1109/SPW.2018.00032

[44] GitHub: *VPNdetection*, https://github.com/FlowFrontiers/VPNdetection (2024. dec.)

[45] Scikit-learn: *PCA*, https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html (2024. dec.)

[46] Wolters Kluwer N.V.: *2024. évi XLIX. törvény* https://mkogy.jogtar.hu/jogszabaly?docid=A2400049.TV (2024. dec.)