UNIVERSITATEA „SAPIENTIA” DIN CLUJ-NAPOCA

FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE,

TÎRGU-MUREȘ

SPECIALIZAREA CALCULATOARE

Implementarea unei aplicații de analiză a traficului de rețea pe Raspberry PI

Coordonator științific:

Ș.l. dr. ing. Hajdu Szabolcs

Absolvent:

Bartha Álmos

2024

UNIVERSITATEA „SAPIENTIA” din CLUJ-NAPOCA
Facultatea de Științe Tehnice și Umaniste din Târgu Mureș
Specializarea: **Calculatoare**

Viza facultății:

LUCRARE DE DIPLOMĂ

Coordonator științific: **ș.l. dr. ing. Hajdu Szabolcs**	Candidat: **Bartha Álmos** Anul absolvirii: **2024**

a) Tema lucrării de licență:
Implementarea unei aplicații de analiză a traficului de rețea pe Raspberry PI

b) Problemele principale tratate:
- Studiu bibliografic privind metode de analiza datelor de trafic
- Realizarea unei aplicații pentru monitorizarea și modificarea a traficului de rețea
- Studiul efectului analizei asupra vitezei de transmisie

c) Desene obligatorii:
- Schema bloc al aplicației
- Diagrame UML privind software-ul realizat.
- Prezentarea rezultatelor obținute

d) Softuri obligatorii:
- Aplicație de monitorizare și modificare a traficului de rețea

e) Bibliografia recomandată:
- A. Jony, M. N. Islam és I. H. Sarker, „Unveiling DNS Spoofing Vulnerabilities: An Ethical Examination Within Local Area Networks," in IEEE, Cox's Bazar, Bangladesh, 2023.
- T. Lammle, CCNA: Cisco Certified Associate Study Guide, Seventh Edition, Indianapolis, Indiana: Wiley Publishing, Inc., 2011.

f) Termene obligatorii de consultații: săptămânal
g) Locul și durata practicii: Universitatea „Sapientia" din Cluj-Napoca,
Facultatea de Științe Tehnice și Umaniste din Târgu Mureș
Primit tema la data de: 31.03.2023
Termen de predare: 1.07.2024

Semnătura Director Departament	Semnătura coordonatorului
Semnătura responsabilului programului de studiu	Semnătura candidatului

Declarație

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

Localitatea,

Data: Absolvent

Semnătura……

Declarație

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

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

Localitatea,

Data: Semnătura îndrumătorului

Ide kerül a Turnitin similarity report

Implementarea unei aplicații de analiză a traficului de rețea pe Raspberry PI Extras

Scopul lucrării este crearea unui sistem încorporat care poate fi integrat între un calculator și rețeaua atașată. Sistemul trebuie să aibă capacitatea de monitorizarea și procesarea pachetele care trec între calculator și rețea și să modifice anumite pachete specificate. Unul dintre pachetele de date care este modificată este pachetul NTP, cu scopul de a sincroniza ceasul intern al calculatorului conform pachetelor modificate de sistemul integrat, în loc de ora primită de la serverele reale. Al doilea tip de pachet care este modificat este pachetul DNS, cu scopul de falsificarea răspunsului, pentru anumite nume de domenii, calculatorul să primească un răspuns falsificat modificat de sistemul integrat, care conține o adresă IPv4 selectată intern în locul răspunsului serverului.

O altă sarcină importantă a sistemului este monitorizarea traficului care trece peste. Pachetele observate în principal sunt cele TLS, FTP și HTTP, cu scopul de a extrage și afișa transparent conținutul relevant. Deoarece pachetele FTP și HTTP nu sunt criptate, informații precum parole și nume de utilizator pot fi, de asemenea, extrase.

Pentru a atinge aceste obiective, un Raspberry Pi este integrat între calculator și rețea, implementând sarcinile menționate anterior în Python. Acest program folosește biblioteca Scapy pentru a monitoriza traficul de rețea, a modifica pachetele de rețea și a afișa conținutul acestora. Transmite pachetele care sosesc la o interfață de rețea atașată către o altă interfață de rețea atașată, în timp ce le procesează și salvează câmpurile relevante.

Cuvinte Cheie: Raspberry, Python, monitorizare trafic de rețea, Scapy, DNS, NTP

SAPIENTIA ERDÉLYI MAGYAR TUDOMÁNYEGYETEM
MAROSVÁSÁRHELYI KAR
SZÁMÍTÁSTECHNIKA SZAK

Hálózati forgalom elemző alkalmazás Raspberry PI egykártyás számítógépen

Témavezető: **Végzős hallgató:**

Dr. Hajdu Szabolcs, egyetemi adjunktus **Bartha Álmos**

2024

Kivonat

A dolgozat célja egy számítógép és a hozza csatolt hálózat közé integrálható rendszer létrehozása. A rendszer képes kell legyen a számítógép és a hálózat között áthaladó csomagokat monitorizálni és feldolgozni, meghatározott csomagokat módosítani. Az egyik adat csomag amely módosításra kerül az NTP ennek módosításával az a cél, hogy a számítógép ne a valódi szerverektől kapott idő szerint szinkronizálja belső óráját, hanem az integrált rendszer által módosított csomagok szerint. A második típusú csomag, amim módosításra került a DNS ennek módosításával az a cél, hogy bizonyos domain nevek esetében a számítógép ne a szerver által küldött választ kapja meg, hanem egy az integrált rendszer által módosított hamisított válasz érkezzen, amely egy belsőleg kiválasztott IPv4 címet tartalmaz.

A rendszer másik fontos feladata az áthaladó forgalom monitorizálása. Az elsődlegesen megfigyelt csomagok a TLS, FTP és HTTP a cél, a releváns tartalom kinyerése átlátható megjelenítése. Mivel az FTP és HTTP csomagok tartalma nincs titkosítva, olyan információkat is kinyerhetőek, mint például jelszavak és felhasználónevek.

Ezen célok elérése érdekében a számítógép és a hálózat közé beépített rendszer egy Raspberry Pi, amelyen egy Pythonban implementált az előbbiekben tárgyalt feladatokat valósítaj meg. Ez a program a Scapy könyvtár segítségével képes monitorizálni a hálózati forgalmat, módosítani a hálózati csomagokat, illetve megjeleníteni azok tartalmát. Az egyik csatolt hálózati interfészre érkező csomagokat továbbítja egy másik csatolt hálózati interfészre, miközben feldolgozza és a releváns mezőket el is menti.

Kulcsszavak: Raspberry, Python, Adatforgalom monitorizálás, Scapy, DNS, NTP

Abstract

The aim of the thesis is to create a system that can be integrated between a computer and the attached network. The system should be capable of monitoring and processing packets passing between the computer and the network, and modifying specified packets. One of the data packets to be modified is the NTP packet, with the goal of having the computer synchronize its internal clock according to the modified packets from the integrated system, rather than the time received from the actual servers. The second type of packet to be modified is the DNS packet, with the goal of having the computer receive a falsified response modified by the integrated system for certain domain names, containing an internally selected IPv4 address instead of the server's response.

Another important task of the system is monitoring the passing traffic. The primarily observed packets are TLS, FTP, and HTTP packets, with the aim of extracting and transparently displaying relevant content. Since FTP and HTTP packets are not encrypted, information such as passwords and usernames can also be extracted.

To achieve these goals, a Raspberry Pi is integrated between the computer and the network, implementing the aforementioned tasks in Python. This program uses the Scapy library to monitor network traffic, modify network packets, and display their content. It forwards packets arriving at one attached network interface to another attached network interface while processing them and saving the relevant fields.

Keywords: Raspberry, Python, Traffic Monitoring, Scapy, DNS, NTP

Tartalomjegyzék

Ábrajegyzék

Táblázat jegyzék

1. Bevezető

A cél egy olyan szoftver környezet kialakítása, amely képes egy számítógép és a hozza csatlakoztatott hálózat közé beágyazódni az itt létrejövő teljes hálózati forgalmat megfigyelni, illetve részben módosítani. A rendszer egy Raspberry PI egykártyás számítógépen fut az integrált hálózati kártyán érkező csomagokat átjátssza egy másik csatolt hálózati interfészre, illetve a releváns mezőket el is menti. Az elmentett forgalom megfigyelhető a csatlakoztatott terminálon. A Raspberry PI-t be helyezzük egy számítógép és a hozza csatolt hálózat közé és elemezzük az áthaladó forgalmat.

A Raspberry PI-t a gyakorlati megvalósítás során egy számítógép és egy Linksys router eszköz közé helyezzük. A számítógépről (belső hálózat) az internet fele a csomagok a Linksys router-en keresztül haladnak. A Rapberry PI elkapja az internet fele haladó csomagokat valamint az internet felől jövőket is paraméterezés szerint módosítja ha szükséges.

A dolgozat célja kettős bizonyos hálózati forgalom módosítása, illetve kiemelt hálózati forgalom monitorizálása.

Az áthaladó csomagok közül az NTP és bizonyos DNS csomagokat módosítunk.

Az NTP csomagok módosítása során a célunk az, hogy a belső hálózat számítógépén a rendszer ne a valós szerverektől kapott idő szerint szinkronizálja az óráját, hanem az általunk módosított csomagok szerint. A számítógép belső óráját így jóval egy jövőbeni vagy a múltbeli időpontra állíthatjuk.

A DNS csomagok módosításával azt akarjuk elérni, hogy bizonyos DNS nevek esetében a számítógép ne a szerver által küldött választ kapja meg, hanem egy általunk készített választ, amely egy általunk kiválasztott IPv4 címet tartalmaz.

A DNS és NTP csomagok módosításán kívül szeretnénk a számunkra kiemelt hálózati forgalmat FTP, HTTP és TLS monitorizálni.

Az eszközt és a programokat a jövőben arra kívánjuk használni, hogy a diákok számára demonstráljuk az NTP és DNS protokoll támadhatóságát a hálózatban, MITM („Man In The Middle”) avagy közbeékelődéses támadás esetén. Továbbá betekintést nyújt abba, hogy miért is nem biztonságosak az olyan protokollok mint az FTP és HTTP protokollok. Ezenkívül felhasználható arra is, hogy gyorsan és egyszerűen megjelenítsük a hálózati csomagok számunkra lényeges tartalmát anélkül, hogy Wireshark, vagy hasonló programok által elfogott csomagokban keljen keresgélni.

2. Elméleti megalapozás és szakirodalmi tanulmány

Ezen fejezet áttekintést ad mindazon technológiákról illetve protokollokról, amelyek a gyakorlati megvalósítás során használva vagy tanulmányozva voltak. Röviden bemutatja minden technológia illetve protokoll működési elvét.

2.1. DNS protokoll

A DNS (Domain Name Sytem) [1] protokoll a tartományneveket IP-címekre oldja fel. A tartománynevek (pl. www.google.com) emberek számára könnyen értelmezhetőek és megjegyezhetőek. Egy tartománynévhez tartozhat IPv4, illetve IPv6 cím is.

A DNS-névtér irányítja a publikus DNS tartományneveket az interneten. A DNS-névtér fordított fa struktúrájú, amely minden csomópontjához a hozzá tartozó tartomány információit leíró erőforrásrekord tartozik. A tartomány neve a szülő csomópont nevét tartalmazza, ez általában egy ponttal van elválasztva a címkétől, és zónákra van osztva. Ez a struktúra teszi lehetővé, hogy a weboldalaknak egyedi nevük legyen.

A tartomány név egy több szintes hierarchikus struktúra. A tartomány nevek egy vagy több címke nevű résszel rendelkeznek, amelyek ponttal vannak elválasztva. Egy címke 63 karaktert tartalmazhat.

A DNS üzenetek kérdésekből és a rájuk érkező válaszokból állnak, amelyek az alábbi mezőket tartalmazzák.

A fejléc tartalmazza azonosítókat, zászlókat, a kérdések és válaszok számát és az RR (erőforrásrekord) mezők számát.

A zászló mező tartalmaz információkat az üzenet típusáról, illetve a lekérdezés státuszáról és arról, hogy a névkiszolgáló mérvadó vagy sem.

A kérdés mező tartalmazza a feloldandó tartomány nevet.

A válasz tartalmazza a feloldandó tartomány név IP címét.

A DNS a User Datagram Protocol-t(UDP) használja arra, hogy válaszoljon a DNS kérésekre, főként a gyorsasága miatt. Abban az esetben, ha a DNS üzenet nagyobb mint 512 byte, akkor Transmission Control Protocol-t(TCP) [1] használ, amely biztosítja az adatok integritását illetve nagyobb csomagok esetében lebontja az üzenetet kisebb csomagokra a gyorsabb célba juttatás érdekében. A kommunikáció TCP vagy UDP 53-as Porton történik. [1]

A tartománynévrendszer elosztott adatbázisra épül. Az adatbázis csomópontjait a névkiszolgálók alkotják. A névkiszolgálók válaszolnak a tartományhoz irányuló kérdésekre.

2.2. NTP protokoll

Az NTP (Network Time Protocol) vagyis hálózati idő protokoll a számítógépes hálózatok óráinak szinkronizálását végzi. Az egyezményes koordinált világidőt tartalmazza, ezredmásodperces pontossággal, kiegészítve szökőmásodpercekkel [2].

Az NTP hierarchikus módon épül fel, ahol a rétegeket stratum-nak nevezzük. A stratum 0 szinten atomórák találhatóak. Ezek nem kapcsolódnak közvetlenül az internethez.

A stratum 1 szinten levő számítógépek kacsolódnak a stratum 0 szinthez.

A stratum 2 szinten levő számítógépek kapcsolódnak NTP kéréseket küldenek, általában több, stratum 1 szintű számítógépeknek. A stratum 2-es gépek egymáshoz is kapcsolódnak, így stabilabb időinformációt biztosítanak [2].

A stratum 3 szint kiszolgálja a stratum 4 szintet, és így tovább, legfeljebb 256. szintig, amelyből a gyakorlatban legfeljebb 16 szint van kihasználva.

A kliens általában legalább három, egymástól különböző hálózaton levő szervert kérdez le. Ahhoz, hogy a kliens az óráját egy távoli szerverhez tudja szinkronizálni ki kell számítania az RTT (Round Time Trip) [2] és az offszet értékét. Az RTT idő alatt azt értjük, amíg egy csomagot elküldünk és nyugta érkezik rá.

Az RTT-t a következő módon számítjuk ki:

$$\delta = (t_3 - t_0) - (t_2 - t_1) \tag{1}$$

Ahol:

- δ -RTT értéke
- t0 - kliens időbélyeg a csomag elküldésének a pillanatában,
- t1 - szerver időbélyeg a csomag érkezésekor,
- t2 - szerver időbélyeg a válasz elküldésekor
- t3 - kliens időbélyeg a válasz érkezésekor.

Az ofszetet (θ) pedig a következő képlettel kapjuk meg:

$$\theta = \frac{[(t_1 - t_0) + (t_2 - t_3)]}{2} \tag{2}$$

A δ és θ értékek kiszűrése után statisztikai elemzés következik. A jelentős eltérést mutató értékek kiesnek, és a fennmaradó három legjobb érték alapján meghatározásra kerül az időeltérés (offset). Az óra frekvenciáját ezután fokozatosan az offset-hez igazítják, ami egy visszacsatolási hurkot hoz létre. A szinkronizáció akkor helyes, ha a bejövő és kimenő útvonalakon az idő szerver és a kliens közötti nominális eltérés szimmetrikus. Ha az útvonalakon nincs szimmetrikus nominális eltérés, akkor egy szisztematikus eltérés kerül kiszámításra, ami a kimenő és a bejövő RTT útvonal idejének különbségének fele [1].

2.3. HTTP protokoll

A HTTP (HyperText Transfer Protocol) [3] egy információátviteli protokoll. A kliens kéréseket küld míg a szerver válaszol erre épül a protokoll. A HTTP protokoll a TCP/IP hálózati réteg felett helyezkedik el az alkalmazás rétegben. Az információ vesztés elkerülésének érdekében TCP protokollt használ.

A HTTP 8 metódust használ. A metódusok az adott erőforráson végrehajtandó műveleteket határozzák meg. A metódusok a következőek:

- GET a leggyakrabban használt metódus, amely az erőforrás letöltését indítja el.
- HEAD ugyanazt adja vissza mint a GET, viszont az üzenet testet (Body) nem tartalmazza a válasz.
- POST adatot küld a szerver fele. Az elküldött adat az üzenettestben van.
- PUT segítségével töltehetünk fel erőforrásokat.
- DELETE töröl erőforrásokat.
- TRACE visszaküldi a kérést. Ennek az a célja, hogy a kliens megnézhesse, hogy a köztes gépek változtatnak-e a kérésen és ha igen, akkor mit.
- OPTIONS egy listát ad vissza amely tartalmazza azokat a metódusokat amelyeket a szerver támogat.
- CONNECT metódust legtöbbször SSL kommunikáció megvalósításánál használják. A kérés TCP/IP csatornává alakítja át. [3]

Fontos megjegyezni, hogy a HTTP protokoll használata nem biztonságos, ugyanis az adatok nincsenek titkosítva az átvitel során, ezért ajánlott a HTTPS protokoll használata. A HTTPS szintaktikailag megegyezik a HTTP-sémával, a különbség az, hogy jelzi a böngészőnek, hogy használja az SSL/TSL réteget, amely titkosítja az adatforgalmat.

2.4. MAC címek

MAC (Media Access Control) címek alapvető szerepet játszanak a hálózati kommunikációban, különösen az Ethernet hálózatokban [1].

A MAC címek egyedi azonosítók, amelyeket a hálózati eszközök hálózati interfészeinek (pl. Ethernet kártyák, Wi-Fi adapterek) azonosítására használnak. Egy MAC cím 48 bites, és általában hat 8 bites (vagy hexadecimális) részből áll.

A MAC címek hat 2 karakteres hexadecimális számként jelennek meg, például: 00:1A:2B:3C:4D:5E [4]

A MAC címek segítségével a hálózati eszközök egyedileg azonosíthatók a lokális hálózaton belül. A hálózati kapcsolók (továbbiakban hálózati switch-ek) és más hálózati eszközök a MAC címeket használják a forgalom megfelelő irányítására az adott hálózati szegmensen belül [5].

Néhány hálózati biztonsági intézkedés, például a MAC cím alapú szűrés, az adott MAC címek alapján engedélyezi vagy tiltja az eszközök hozzáférését a hálózathoz.

Az Ethernet hálózatokban az adatkapcsolati réteg (OSI modell 2. réteg) használja a MAC címeket az eszközök közötti kommunikációhoz. Az Ethernet keretek tartalmazzák a forrás és cél MAC címeket, amelyek alapján a hálózati switchek továbbítják a kereteket a megfelelő portokra.

IEEE 802.11 [4] (Wi-Fi): A Wi-Fi hálózatokban az IEEE 802.11 szabvány használja a MAC címeket az eszközök azonosítására és az adatok megfelelő célponthoz történő továbbítására.

2.5. ARP protokoll

Az ARP (Address Resolution Protocol) [1] egy hálózati protokoll, amely az IP címekhez kapcsolódó fizikai (MAC) címek meghatározására szolgál a helyi hálózatokon (LAN). Az ARP alapvető szerepet játszik az IPv4 hálózatokban, lehetővé téve az eszközök közötti kommunikációt azáltal, hogy hozzárendeli az IP címeket a megfelelő MAC címekhez. Amikor egy eszköz ismeri a hálózatban az IP címet, de nem ismeri a hozzá tartozó MAC címet, akkor ARP kérést küld a hálózatra, hogy megkapja válaszként a szükséges MAC címet.

A hálózati switch-ek a MAC címek alapján döntik el, hogy melyik porton küldjék ki a beérkező Ethernet kereteket.

2.6. UDP protokoll

Az UDP (User Datagram Protocol) [1] egy egyszerű és gyors kommunikációs protokoll, amelyet az internetes hálózatokban használnak adatcsomagok küldésére és fogadására. Az UDP a TCP/IP

protokollcsalád része, és a szolgáltatási rétegben helyezkedik el. Az UDP-t leggyakrabban olyan alkalmazásokban használják, ahol a gyors adatátvitel fontosabb, mint az adatok integritása [5].

Az UDP nem épít fel kapcsolatot a küldő és a fogadó között az adatátvitel előtt. Ehelyett egyszerűen elküldi az adatcsomagokat a célcímre anélkül, hogy biztosítaná azok kézbesítését vagy helyes sorrendben való megérkezését. Emiatt az UDP-t kapcsolat nélküli (connectionless) [1] protokollnak nevezik.

Az UDP fejléc egyszerű és kis méretű, mindössze 8 byte hosszú, ami hozza járul a gyors adatátvitelhez. Az UDP fejléc a következő mezőket tartalmazza: forrás port (16 bit), cél port (16 bit), üzenet hossza (16 bit), ellenőrző összeg (16 bit) (opcionális, a fejléc és az adat hibamentességének ellenőrzésére szolgál).

Az UDP nem garantálja az adatcsomagok kézbesítését, sorrendjét vagy integritását. Az adatcsomagok elveszhetnek, megduplázódhatnak vagy rossz sorrendben érkezhetnek meg. Az alkalmazásnak kell kezelnie az esetleges adatveszteséget vagy hibákat.

Az UDP ideális olyan alkalmazásokhoz, ahol a sebesség és az alacsony késleltetés fontosabb, mint az adatátvitel megbízhatósága. Néhány példa az UDP-t használó alkalmazásokra:

- Média sugárzás (media stream): Videó- és hangfolyamok valós idejű továbbítása.
- Online játékok: Gyors adatátvitel a játékosok közötti interakciókhoz.
- DNS lekérdezések: Gyors domain név feloldás.
- VoIP: Hangátvitel az interneten keresztül.

Az UDP egyszerűsége miatt könnyen implementálható és kevés erőforrást igényel a hálózati eszközökön és az alkalmazásokban. Ennek köszönhetően széles körben elterjedt és sokféle hálózati kommunikációs célra használják.

2.7. TCP protokoll

A TCP („Transmission Control Protocol”) [5] az egyik legfontosabb és legszélesebb körben használt kommunikációs protokoll az interneten. A TCP az adatokat megbízhatóan és sorrendben továbbítja a küldő és a fogadó között, és az internetes alkalmazások nagy részében használják.

A TCP kapcsolat-orientált protokoll, ami azt jelenti, hogy a küldő és a fogadó először felépít egy logikai kapcsolatot (TCP kapcsolat) az adatátvitel előtt. Ez a kapcsolatfelvétel háromlépcsős kézfogásnak nevezett folyamaton keresztül történik, amely lépései a következőek:

- SYN: A kliens elküld egy SYN (synchronize) csomagot a szervernek.
- SYN-ACK: A szerver válaszol egy SYN-ACK (synchronize-acknowledge) csomaggal.
- ACK: A kliens elküld egy ACK (acknowledge) csomagot, és a kapcsolat létrejön.

A TCP biztosítja, hogy az adatcsomagok helyes sorrendben és hibamentesen érkezzenek meg a célhoz. Ha egy csomag elvesz vagy hibásan érkezik, a TCP automatikusan újra küldi azt. Az adatcsomagok helyes kézbesítésének ellenőrzéséhez a TCP nyugtázást (ACK) használ.
A TCP garantálja, hogy az adatcsomagok abban a sorrendben érkezzenek meg, ahogyan azokat elküldték. Az adatcsomagokat sorszámokkal látja el, és ha egy csomag rossz sorrendben érkezik, a TCP vár, amíg az összes csomag megérkezik, majd a helyes sorrendben továbbítja azokat az alkalmazásnak.
A TCP áramlásszabályozást alkalmaz annak érdekében, hogy a küldő ne árassza el a fogadót több adattal, mint amennyit az fel tud dolgozni. Ehhez az úgynevezett "csúszó ablak" mechanizmust használja, amely dinamikusan szabályozza az elküldhető adatmennyiséget a fogadó kapacitásának megfelelően.
A TCP torlódásszabályozást is használ, hogy elkerülje a hálózati torlódásokat. Az algoritmusok, mint például a „Slow Start” és a „Congestion Avoidance” [1], segítenek optimalizálni az adatátviteli sebességet, és elkerülni a hálózat túlterhelését.
A TCP fejléc bonyolultabb és nagyobb, mint az UDP fejléc, és számos mezőt tartalmaz az adatátvitel kezeléséhez: forrás port (16 bit), cél port (16 bit), sorszám (32 bit), nyugtázási szám (32 bit), fejléc hossza (4 bit), zászlók (6 bit) - például SYN, ACK, FIN, ablak méret (16 bit), ellenőrző összeg (16 bit), fontos (urgent) mutató (16 bit), opciók (változó hosszúságú). [6]
A TCP-t széles körben használják olyan alkalmazásokban, ahol a megbízhatóság és az adat integritása kritikus fontosságú:

- Webböngészők (HTTP/HTTPS): Weboldalak és online tartalmak letöltése.
- E-mail (SMTP, POP3, IMAP): E-mailek küldése és fogadása.
- Fájlátvitel (FTP): Fájlok átvitele a hálózaton keresztül.
- Távoli hozzáférés (SSH, Telnet): Biztonságos távoli bejelentkezés és parancssori hozzáférés.

2.8. IP protokoll

Az IP („Internet Protocol”) [6] az egyik legfontosabb és legelterjedtebb hálózati protokoll, amely az adatcsomagok címzéséért és útválasztásáért felelős az interneten és más hálózatokban. Az IP két fő verziója létezik: az IPv4 és az IPv6.
Az IP protokoll a hálózati rétegben működik. Feladata az adatok csomagokra bontása, címzése és útvonalak kijelölése, hogy az adatcsomagok elérjék a céljukat. Az IP nem garantálja a csomagok megérkezését, sorrendjét vagy integritását; ezekért a magasabb szintű protokollok (pl. TCP) felelősek.

Az IPv4 32 bites címeket használ, ami összesen körülbelül 4,3 milliárd egyedi IP-címet tesz lehetővé. Az IPv4 címek négy oktettből (8 bites csoportokból) állnak, és általában pontokkal elválasztott decimális formában ábrázolják őket (pl. 192.168.1.1) [6] .

Az IPv4 fejléc 20-60 byte hosszú, és a következő mezőket tartalmazza:

- Verzió (4 bit): Az IP verziószámát jelzi (IPv4 esetén 4).
- Fejléc hossza (4 bit): Az IP fejléc hosszát adja meg.
- Szolgáltatás típusa (8 bit): Az adatcsomag prioritását és QoS (Quality of Service) információkat tartalmaz.
- Teljes hossz (16 bit): Az egész IP csomag hossza.
- Azonosító (16 bit): Az adatcsomag azonosítására szolgál.
- Flags (3 bit): Fragmentációs információkat tartalmaz.
- Fragment offset (13 bit): A csomagtöredék helyét jelzi az eredeti csomagban.
- TTL (8 bit): Az élettartam, amely csökkenti minden egyes útválasztásnál.
- Protokoll (8 bit): Az adatot tartalmazó protokollt azonosítja (pl. TCP, UDP).
- Fejléc ellenőrző összeg (16 bit): A fejléc hibamentességét ellenőrzi.
- Forrás IP cím (32 bit): Az adatok küldőjének IP címe.
- Cél IP cím (32 bit): Az adatok céljának IP címe.
- Opciók (változó hosszúságú): Opcionális fejléckiterjesztések.

Az IPv4 támogatja az adatcsomagok darabolását, ami azt jelenti, hogy a nagyobb csomagokat kisebb részeke bontja, hogy megfeleljenek az adott hálózat maximális csomagméretének (MTU) [1].

2.9. ICMP protokoll

Az ICMP („Internet Control Message Protocol”) [7] az IP (Internet Protocol) részeként definiáltak. Az ICMP célja, hogy hibajelentéseket és egyéb hálózati diagnosztikai információkat továbbítson a hálózati eszközök között. Az ICMP nélkülözhetetlen a hálózati hibaelhárításban és a hálózati működés optimalizálásában.

Az ICMP számos üzenettípust támogat, amelyek mindegyike különböző hálózati helyzetek kezelésére szolgál.

- **Echo Request** (Típus 8): Ezt az üzenetet a ping parancs küldi, hogy ellenőrizze, elérhető-e egy adott hálózati eszköz.
- **Echo Reply** (Típus 0): Ezt az üzenetet válaszként küldik az Echo Request-re, jelezve, hogy a cél eszköz elérhető.
- **Destination Unreachable** (Típus 3): Ezt az üzenetet akkor küldik, ha egy csomag nem érheti el a célját. Több al-típusa van, például: Network Unreachable (Kód 0) (a cél hálózat nem elérhe-tő), **Host Unreachable** (Kód 1) (a cél gazda nem elérhető), Port Unreachable (Kód 3) (a cél-port nem elérhető) [8].
- **Time Exceeded** (Típus 11): Ezt az üzenetet akkor küldik, ha egy csomag élettartama (TTL) lejárt, mielőtt elérte volna a célját. Ezt az üzenetet gyakran használják a traceroute parancsban.

- **Redirect** (Típus 5): Ezt az üzenetet akkor küldik, ha egy jobb útvonal létezik egy csomag céljához. Ez segít optimalizálni a hálózati útvonalakat.

Az ICMP üzenetek fejlécstruktúrája egyszerű, és a következő mezőket tartalmazza:
típus (8 bit): az ICMP üzenet típusát jelzi (pl. Echo Request, Destination Unreachable);
kód (8 bit): az üzenet típuson belüli specifikus kódot jelzi (pl. Network Unreachable, Host Unreachable);
ellenőrző összeg (16 bit): az ICMP üzenet hiba mentességének ellenőrzésére szolgál [8].
Egyéb mezők: Az üzenet típustól függően további mezőket tartalmazhat (pl. az Echo Request/Reply esetén az azonosító és szekvenciaszám) [5] [8].
A ping egy egyszerű hálózati diagnosztikai eszköz, amely ICMP Echo Request üzeneteket küld egy cél IP címre, és várja az Echo Reply üzeneteket. Ez segít meghatározni, hogy egy adott eszköz elérhető-e, illetve méri a visszaérkezési időt.
A traceroute egy olyan hálózati diagnosztikai eszköz, amely ICMP Time Exceeded üzeneteket használ az útvonal meghatározására, amelyen keresztül egy csomag eljut a céljához. A traceroute úgy működik, hogy növeli a TTL értéket minden egyes csomagban, és figyeli a Time Exceeded üzeneteket, hogy feltérképezze az útvonalat [8] [9].
Az ICMP protokollt gyakran használják hálózati támadásokhoz, például DoS (Denial of Ser-vice) támadásokhoz vagy hálózati feltérképezéshez. Emiatt sok hálózati adminisztrátor korlátozza vagy szűri az ICMP forgalmat a tűzfalak és egyéb biztonsági eszközök segítségével.

2.10. TLS protokoll

A „Transport Layer Security” [8](TLS) egy kriptográfiai protokoll, amely a számítógépes hálózatokon keresztül történő biztonságos kommunikációt biztosítja. Az e-mail, az azonnali üzenetküldés és a VoIP („Voice over IP”) [8] alkalmazásokban is széles körben alkalmazzák, de leginkább a HTTPS biztonságának megteremtésében ismert.
A TLS protokoll fő célja a biztonság nyújtása, amely magában foglalja a titkosságot, az adatok sértetlenségét és hitelességet. Ezt kriptográfiai módszerekkel, például tanúsítványok használatával éri el, két vagy több kommunikáló számítógépes alkalmazás között. A TLS a prezentációs rétegben működik, és két fő komponensből áll: a TLS rekord protokollból és a TLS kézfogási protokollból. Mivel az alkalmazások képesek TLS (vagy SSL) nélkül is kommunikálni, szükséges, hogy az ügyfél kérje a szervertől a TLS kapcsolat létrehozását. Az egyik leggyakoribb módja ennek az,

hogy másik port számot használnak a TLS kapcsolatokhoz. Például a port 80 általában a titkosítástalan HTTP forgalomhoz van fenntartva, míg a port 443 [8] a titkosított HTTPS forgalomhoz használatos. Egy másik mechanizmus a protokoll specifikus STARTTLS kérés, amely a szervertől kéri, hogy váltson TLS kapcsolatra – például az e-mail és hír protokollok használatakor.

Miután az ügyfél és a szerver megegyeznek a TLS használatában, egy kézfogási eljárással tárgyalják meg az állapotfüggő kapcsolatot. A protokollok aszimmetrikus titkosítással végzik a kézfogást [9], amely során nemcsak a titkosítási beállításokat határozzák meg, hanem egy munkamenet-specifikus megosztott kulcsot is létrehoznak, amellyel a további kommunikációt szimmetrikus titkosítással titkosítják. A kézfogás során az ügyfél és a szerver különféle paraméterekben állapodnak meg, amelyek a kapcsolat biztonságának megteremtéséhez szükségesek:

A kézfogás akkor kezdődik, amikor az ügyfél csatlakozik egy TLS-képes szerverhez, biztonságos kapcsolatot kérve, és bemutat egy listát a támogatott titkosító algoritmusokról (titkosítások és hash függvények) [8].

A listából a szerver kiválaszt egy olyan titkosító és hash függvényt, amelyet maga is támogat, és értesíti az ügyfelet a döntésről.

A szerver általában ezután egy digitális tanúsítvány formájában azonosítja magát. A tanúsítvány tartalmazza a szerver nevét, a hitelesített tanúsítványkiadót (CA), amely a tanúsítvány hitelességét igazolja, és a szerver nyilvános titkosítási kulcsát [9].

Az ügyfél ellenőrzi a tanúsítvány érvényességét, mielőtt folytatná.

A munkamenet kulcsok generálásához az ügyfél vagy:

1. egy véletlenszámot (PreMasterSecret) titkosít a szerver nyilvános kulcsával és elküldi a szervernek (amit csak a szerver tud dekódolni a saját privát kulcsával); mindkét fél ezután a véletlenszámot használja egy egyedi munkamenet kulcs létrehozásához, amellyel a további adatokat titkosítják és dekódolják a munkamenet során, vagy
2. Diffie–Hellman kulcscserét (vagy annak elliptikus görbe változatát, az ECDH-t) használ a véletlenszerű és egyedi munkamenet kulcs biztonságos generálásához, amely rendelkezik az előre tekintő titkosság tulajdonságával: ha a szerver privát kulcsa a jövőben kiderül, az nem használható a jelenlegi munkamenet dekódolására, még akkor sem, ha a munkamenetet egy harmadik fél elfogta és rögzítette.

Ezzel zárul a kézfogás, és megkezdődik a biztonságos kapcsolat, amelyet a munkamenet kulccsal titkosítanak és dekódolnak egészen a kapcsolat lezárásáig. Ha bármelyik fenti lépés sikertelen, a TLS kézfogás nem sikerül, és a kapcsolat nem jön létre.

2.11. FTP protokoll

Az FTP („File Transfer Protocol”) [9] egy régi, de még mindig széles körben használt hálózati protokoll, amelyet fájlok átvitelére terveztek egy ügyfél (client) és egy szerver (server) között egy TCP/IP hálózaton keresztül. Az FTP-t az IETF („Internet Engineering Task Force”) [9] szabványosította az RFC 959 dokumentumban. Az FTP egy kapcsolatorientált protokoll, ami azt jelenti, hogy az adatok átvitele előtt kapcsolatot kell létrehozni az ügyfél és a szerver között.

Az FTP parancsok és válaszok a vezérlő kapcsolaton keresztül haladnak, míg a tényleges adatátvitel az adatkapcsolaton keresztül történik.

A vezérlő kapcsolat egy TCP kapcsolat a következő tulajdonságokkal:

- Forrás port: Ügyfél által választott véletlenszerű port.
- Cél port: 21 (FTP vezérlő port).
- A vezérlő kapcsolaton keresztül az ügyfél és a szerver FTP parancsokat és válaszokat cserélnek.
- Az adat kapcsolat a tényleges fájlátvitelhez használt TCP kapcsolat. Az adat kapcsolat jellemzői függnek attól, hogy aktív vagy passzív módban történik az átvitel.

Aktív Mód:

- Forrás port: 20 (FTP adat port) [10].
- Cél port: Ügyfél által választott véletlenszerű port.
- Passzív Mód:
- Forrás port: Szerver által választott véletlenszerű port [10].
- Cél port: Ügyfél által választott véletlenszerű port.

Az FTP protokoll nem titkosított, ezért az átvitt adatok, beleértve a felhasználóneveket és jelszavakat, sima szövegként kerülnek továbbításra, ami biztonsági kockázatot jelent. Az FTP biztonságosabb változatai közé tartozik az FTPS („FTP Secure”) és az SFTP („SSH File Transfer Protocol”), amelyek titkosítást használnak az adatvédelem érdekében.

3. Felhasznált technológiák

3.1. Scapy/PCAP

Scapy egy erőteljes Python-alapú interaktív programozási környezet, amelyet hálózati csomagok létrehozására, manipulálására, továbbítására és elemzésére használnak. A Scapy-t Philippe Biondi [10] fejlesztette ki, és számos hálózati biztonsági szakember, kutató és IT szakember használja különféle hálózati feladatok elvégzésére.

A hálózaton közlekedő csomagok elfogásának és elemzésének több fontos felhasználási területe is létezik. A rendszergazdák felhasználhatják hálózati problémák felderítésére, biztonsági rések felderítésére. Ezenkívül hasznos a hálózat működésének a megértésére is. Fontos megjegyezni, hogy a hálózati csomagok elfogása még nem jelenti a hálózati csomagok manipulációját, csak a hálózat megfigyelését. A Scapy képes a hálózati csomagok módosítására is.

A Scapy lehetővé teszi a hálózati forgalom rögzítését („sniffing") [10], hasonlóan más hálózati elemző eszközökhöz, mint például a Wireshark. A sniff függvény használatával könnyedén elfoghatók a hálózati csomagok.

Az elfogott csomagokat a Scapy segítségével részletesen elemezhetjük. A csomagok különböző protokoll rétegekre bonthatók, és az egyes rétegek mezői elérhetők.

A Scapy lehetővé teszi csomagok létrehozását a semmiből, valamint meglévő csomagok módosítását. A Scapy támogatja a különféle hálózati protokollok rétegiet, mint például Ethernet, IP, TCP, UDP, ICMP stb. Ezeket a rétegeket osztályokként definiálják, amelyek lehetővé teszik a csomagok könnyű létrehozását és manipulálását.

A Scapy a pcap („Packet Capture") [10] könyvtárakat használja a csomagok elfogására és elemzésére. A pcap könyvtárak biztosítják az alacsony szintű hozzáférést a hálózati interfészekhez.

Mivel a Scapy Pythonban van írva, a felhasználók a teljes Python ökoszisztémát kihasználhatják, beleértve a széleskörű könyvtárakat és modulokat, amelyek segítenek az adatok feldolgozásában és vizualizálásában.

A „pcap" („Packet Capture") egy szabványos alkalmazásprogramozási felület (API) hálózati csomagok elfogásához és feldolgozásához. A „pcap" könyvtárak lehetővé teszik a felhasználók számára, hogy alacsony szintű hozzáférést kapjanak a hálózati interfészekhez, és csomagokat rögzítsünk vagy generáljunk különféle hálózati elemzési és biztonsági célokra.

A „pcap” lehetővé teszi a felhasználók számára, hogy kiválasszanak egy hálózati interfészt, amelyen a csomagokat elfogják. Az interfészek lehetnek Ethernet, Wi-Fi vagy más típusú hálózati kapcsolatok. A „pcap” alacsony szintű hozzáférést biztosít a hálózati interfészekhez, gyakran közvetlenül a kernel szintjén dolgozva. Ez lehetővé teszi a csomagok valós idejű elfogását és generálását.
A „pcap” csomagokat képes rögzíteni valós időben, vagy fájlból is olvashatja őket. Az elfogott csomagok a „pcap loop” vagy „pcap dispatch” függvények segítségével kerülnek feldolgozásra.
Minden elfogott csomagot egy „callback” [10] függvény dolgoz fel, amelyet a felhasználó definiál. Ez a függvény felelős a csomag adatainak értékeléséért és feldolgozásáért.
A pcap API két fő implementációja a libpcap (Linux és Unix-alapú rendszerekhez) és a WinPcap (Windows rendszerekhez). Mindkét implementáció biztosítja az alapvető funkcionalitást, de a WinPcap kiegészítő funkciókat is kínál Windows környezetben.

3.2. NetfilterQueue

A NetfilterQueue közvetlenül a Linux kernel szintjén működik, és képes hálózati csomagokat elfogni, mielőtt azok elérnék a felhasználói térben futó alkalmazásokat.
Ezáltal lehetőséget biztosít nagyon alacsony szintű csomagkezelésre.
A NetfilterQueue integrálódik az IPTables szabályrendszerrel, amely lehetővé teszi a hálózati csomagok irányítását és szűrését a kernel szintjén. Ez sokkal rugalmasabb és hatékonyabb csomagszűrést és manipulációt tesz lehetővé, mint amit a Scapy egyedül kínál.
A NetfilterQueue és a Scapy együtt használva kiegészítik egymást, és együttesen erőteljes megoldást kínálnak hálózati csomagok kezelésére.
A NetfilterQueue segítségével alacsony szinten elfoghatjuk a csomagokat a kernel szintjén, majd ezeket a csomagokat továbbadhatjuk a Scapy-nak részletes elemzés és manipuláció céljából.
A Netfilter a Linux kernel része, és a hálózati csomagok szűrésére, NAT-olására és egyéb hálózati műveletek végrehajtására szolgál. Az IPTables eszközzel konfigurálható, amely lehetővé teszi a szabályok létrehozását és kezelését.
A Netfilter különböző helyeken "hook” [11]-okat (kampókat) biztosít a hálózati stack-ben. Ezek a hook-ok lehetővé teszik, hogy a csomagok bizonyos pontokon keresztülmenjenek, ahol a Netfilter modulok beavatkozhatnak. A fő hook pontok:

- NF_INET_PRE_ROUTING: A csomagok beérkezése után, de még a routing előtt.

- NF_INET_LOCAL_IN: A csomagok a helyi gépre érkeznek be.
- NF_INET_FORWARD: A csomagok a gépen keresztülhaladnak, de nem a helyi gépre vannak címezve.
- NF_INET_LOCAL_OUT: A csomagok a helyi gépről kerülnek kiküldésre.
- NF_INET_POST_ROUTING: A csomagok elhagyják a gépet, miután a routing megtörtént. [11]

Az IPTables egy eszköz a Netfilter konfigurálására és kezelésére. Segítségével a felhasználók szabályokat hozhatnak létre a csomagok kezelésére különböző láncokban (chains) és táblákban (tables). Az IPTables általános táblái:

- filter: A csomagok szűrésére szolgál.
- nat: A csomagok NAT-olására (pl. forrás NAT, cél NAT).
- mangle: A csomagok módosítására (pl. TOS mező beállítása).
- raw: A csomagokhoz kapcsolódó nyers kezelésekre. [11]

A szabályok meghatározzák, hogy egy adott csomag hogyan legyen kezelve a láncon belül. A szabályok különböző feltételek alapján szűrhetőek és különböző célokat (targets) szolgálnak. Az IPTables szabályokban használt leggyakoribb célok:

- ACCEPT:Elfogadja a csomagot, és további feldolgozás nélkül engedi tovább.
- DROP:Eldobja a csomagot, mintha az sosem érkezett volna meg.
- REJECT:Elutasítja a csomagot, és értesítést küld a küldőnek.
- LOG:Naplózza a csomagot, majd továbbítja a lánc következő szabályához.
- MASQUERADE:NAT-olás (forrás NAT) alkalmazása, általában dinamikus IP-címekhez.
- DNAT:Cél NAT alkalmazása, a cél IP-cím megváltoztatása.
- SNAT:Forrás NAT alkalmazása, a forrás IP-cím megváltoztatása. [11]

Az IPTables láncok sorozata, amelyekben a szabályok végrehajtásra kerülnek. Minden tábla külön láncokat tartalmaz. Az alapértelmezett láncok:

- INPUT: A helyi gépre érkező csomagok.
- OUTPUT: A helyi gépről kimenő csomagok.
- FORWARD: az IPTables rendszerben az átmenő forgalmat kezeli, vagyis azokat a csomagokat, amelyek a helyi gépen csak áthaladnak, de nem a helyi gépre vannak címezve. Ez a lánc különösen fontos routerek és átjárók (gateways) esetén, ahol a forgalom nagy része áthalad a rendszeren.
- PREROUTING: A routing előtt a csomagokra alkalmazandó szabályok.
- POSTROUTING: A routing után a csomagokra alkalmazandó szabályok. [11]

3.3. Ebtables

Az ebtables egy olyan parancssori eszköz, amely lehetővé teszi a hálózati forgalom szabályozását az Ethernet szinten (OSI modell 2. réteg) [12]. Az ebtables különösen hasznos, ha hidalt (bridged) hálózatokon szeretnénk szűrni a forgalmat, mert az iptables nem képes elkapni a hálózati hidon áthaladó csomagokat.

3.3.1 Keret szűrési célok

Egy tűzfalszabály meghatározza egy Ethernet keret kritériumait és egy keret feldolgozási specifikációt, amit csomagcélnak nevezünk. Amikor egy keret megfelel egy szabálynak, akkor a következő műveletet a kernel a célpont alapján hajtja végre. A célpont lehet az alábbi értékek egyike: ACCEPT, DROP, CONTINUE, RETURN, [11] egy 'kiterjesztés' vagy ugrás egy felhasználó által definiált láncra.

Az ACCEPT azt jelenti, hogy a keret átmehet. A DROP azt jelenti, hogy a keretet el kell dobni. Azonban a BROUTING láncban az ACCEPT és DROP célpontok különböző jelentéssel bírnak. A CONTINUE azt jelenti, hogy a következő szabályt kell ellenőrizni. Ez hasznos lehet például arra, hogy megtudjuk, hány keret halad át egy adott ponton a láncban, hogy naplózzuk ezeket a kereteket vagy hogy több célpontot alkalmazzunk egy keretre. A RETURN azt jelenti, hogy leállítjuk az adott lánc bejárását és folytatjuk az előző (hívó) lánc következő szabályánál.

3.3.1. Táblák

A Linux kernelben három ebtables tábla található. Ezek nevei: filter, nat és broute [12]. Ezen három tábla közül a filter tábla az alapértelmezett, amelyen a parancs működik.

A filter az alapértelmezett tábla, és három beépített láncot tartalmaz: INPUT (a hídnak szánt keretekhez, a MAC célcím szintjén), OUTPUT (helyben generált vagy (b)route-olt keretekhez) és FORWARD (a híd által továbbított keretekhez).

A NAT elsősorban a MAC címek megváltoztatására használható, és három beépített láncot tartalmaz: PREROUTING (a keretek módosításához, amint beérkeznek), OUTPUT (helyben generált vagy (b)route-olt keretek módosításához, mielőtt hidaltak lennének) és POSTROUTING (a keretek módosításához, mielőtt kimennének). Egy kis megjegyzés a PREROUTING és POSTROUTING láncok elnevezéséhez: pontosabb lenne őket PREFORWARDING és POSTFORWARDING [12]-nak nevezni, de azok számára, akik az iptables világból érkeznek az ebtables-hez, könnyebb ugyanazokat a neveket használni. Megjegyzendő, hogy ha nem tetszik az alapértelmezett név, megváltoztathatja.

A DROP valójában azt jelenti, hogy a keretet útvonalra kell állítani, míg az ACCEPT azt jelenti, hogy a keretet hidalni kell. A BROUTING lánc nagyon korán bejárásra kerül. Azonban csak a híd porton keresztül beérkező keretek esetén járják be, amelyek átirányítási állapotban vannak. Ezeket a kereteket általában hidalni kellene, de itt dönthetünk másképp. Az irányítási célpont itt nagyon hasznos.

3.3.2. Ebtables parancssori argumentumok

Az ebtables '-t table' parancssori argumentuma után a további argumentumokat több csoportba lehet osztani. Ezek a csoportok a parancsok, vegyes parancsok, szabály specifikációk, egyezés kiterjesztések, figyelő kiterjesztések és cél kiterjesztések.

Parancsok

Az ebtables parancs argumentumok meghatározzák a táblán elvégzendő műveleteket a -t argumentummal megadott tábla esetén. Ha nem használjuk a -t argumentumot egy tábla nevének megadására, akkor a parancsok az alapértelmezett filter táblára vonatkoznak. Egyszerre csak egy parancs használható a parancssorban, kivéve, ha a -L és -Z parancsokat kombináljuk, a -N és -P parancsokat kombináljuk, vagy ha „--atomic-file” [12]-t használunk.

3.4. iPerf

Az iPerf egy nyílt forráskódú, hálózati teljesítmény mérésére szolgáló eszköz, amelyet főként sávszélesség és adatátvitel tesztelésére használnak. A program különféle hálózati protokollokkal és technológiákkal kompatibilis, beleértve a TCP és UDP protokollokat, valamint IPv4 és IPv6 hálózattokat.

Az iPerf lehetővé teszi a hálózati kapcsolat maximális sávszélességének meghatározását. Ezzel a funkcióval megállapítható, hogy egy adott hálózati kapcsolaton mennyi adatot milyen sávszéleséggel lehet átvinni egy meghatározott idő alatt.

Az iPerf használatával mérhető a hálózat késleltetése, „jitter” [13], és adatvesztés, ami segít a hálózati teljesítmény részletesebb elemzésében.

Hálózati problémák diagnosztizálására is használható, mivel lehetővé teszi a hálózati forgalom generálását és elemzését különböző körülmények között.

Az iPerf támogatja mind a TCP (Transmission Control Protocol), mind a UDP (User Datagram Protocol) protokollokat [13].

Az iPerf parancssori eszköz, amely egyszerű parancsokkal vezérelhető, így könnyen beállítható és használható. Az iPerf több operációs rendszeren is fut, beleértve a Linux, Windows és macOS rendszereket.

Az iPerf teszt során az ügyfél egy előre meghatározott mennyiségű adatot küld az iPerf szervernek, amely méri az adatátvitel időtartamát. A teszt végezhető csak feltöltési, csak letöltési, vagy kétirányú (feltöltési és letöltési) módban. Ezen információk alapján az iPerf kiszámítja a hálózat teljesítménymutatóit, amelyeket a hálózati mérnökök elemezhetnek és értelmezhetnek.

3.5. Ismert hasonló alkalmazások

1.*Wireshark*: Ez egy népszerű hálózati protokoll analizátor, amely lehetővé teszi a hálózati forgalom rögzítését és elemzését. Telepíthető és futtatható Raspberry Pi 4-en is.

2.*tcpdump*: Ez egy parancssori eszköz a hálózati forgalom rögzítésére és elemzésére. Telepíthető a Raspberry Pi 4-re, és lehetővé teszi a csomagok részletes elemzését.

3.*Tshark*: Ez a Wireshark parancssori változata, amely lehetővé teszi a csomagok rögzítését és elemzését parancssori módban. Hasznos lehet, ha nincs grafikus felület elérhető.

4.*Bro*(*ZEEK*): Egy hálózati forgalom elemző rendszer, amely lehetővé teszi a csomagok mélyreható elemzését és értelmezését. Nagyobb projektekben hasznos lehet.

5. *Suricata*: Hasonlóan a Snorthoz, a Suricata is egy hálózati fenyegetésmegelőző rendszer, amely képes csomagok rögzítésére és elemzésére a hálózaton.

Az előbbiekben felsorolt alkalmazások közül egyik sem képes a hálózati csomagok módosítására a legnépszerűbb alkalmazás a Wireshark amely az áthaladó forgalom minden bájtját képes analizálni.

Fiddler segítségével lehet módosítani a HTTP és HTTPS csomagokat.

CommView for WiFi egy másik alkalmazás, amely segítségével módosítani lehet a WiFi hálózaton elkapott csomagokat.

A *A. Jony, M. N. Islam* és *I. H. Sarker* átal írt dolgozatban „Unveiling DNS Spoofing Vulnerabilities: An Ethical Examination Within Local Area Networks” [14] megvalósítottak egy olyan rendszert amelyben a támadó egy hamis DHCP szervert telepít a helyi hálózatba, és erőszakosan átirányítja a felhasználók forgalmát egy rosszindulatú vagy hamis alapértelmezett átjáróhoz. Másodszor, egy Python (Scapy) alapú DNS hamisító motort telepítenek a hamis DHCP szerverre, amely meghamisítja a DNS szerver válaszait. A Scapy megváltoztatja a legitim weboldal IP címét egy rosszindulatú vagy adathalász weboldal IP címére, amelyet a támadó kezel,

és úgy néz ki, mint a valódi weboldal. Ennek eredményeként a kompromittált felhasználót azonos kinézetű adathalász weboldalra irányítják át, és adathalász támadás áldozatává válik.

Aanchal Malhotra, Isaac E. Cohen, Erik Brakke, and Sharon Goldberg által írt dolgozatban „Attacking the Network Time Protocol” [15] az NTP protokoll támadhatóságát tárgyalják, és sikerült befolyásolniuk a számítógép belső óráját, a hálózaton elkapott NTP csomagok módosításával.
Ezek ismeretében a dolgozatban én is a Scapy könyvtárat használtam az adatforgalom monitorizálására illetve módosítására.

4. A rendszer specifikációi és architektúrája

4.1. Hardver

Miután több különböző forrást is megnéztem, arra a következtetésre jutottam, hogy egy Raspberry PI 2 kártya valószínűleg nem lenne elég preformáns a projekt megvalósításához.

A Raspberry PI 2 kártya processzora, illetve RAM memória mérete szűk keresztmetszetett jelentethet. Míg egy Rasbperry PI 2 kártya olcsóbb lenne, mint egy Rasbperry PI 4, nem biztos, hogy az eszköz megfelelő lenne az integrált hálózati kártyán érkező csomagok átjátszására egy másik csatolt hálózati interfészre és közben el is menteni azokat.

Érdemes megemlítenem azt is, hogy a hardver követelmény nagyban függ a hálózattól, amelyben használni szeretnénk, például a hálózati átvitel sávszélességétől.

A Raspberry Pi 4 model B általános jellemzői

- Processzor: Broadcom 2711;
- Processzor architektúrája: Cortex-A72, 64 bites;
- Processzor frekvencia: 1.5 GHz;
- Fizikai magok száma: 4;
- RAM memória: 4 GB LPDDR4 SDRAM;
- Csatlakoztathatóság: WiFi: 2.4 GHz és 5 GHz IEEE 802.11 b/g/n/ac;
- Ethernet: Gigabit;
- 2 x USB 2.0; 2 x USB 3.0;
- MicroSD kártyahely az operációs rendszer futtatásához és adatok tárolásához;
- USB Type C táplálás: 5V, 3A;
- Működési hőmérséklet: 0 - 50 Celsius fok;
- Videó és audió: 2 x micro HDMI csatlakozó; [16]

A Rasbperry PI tápforrása egy külső akkumulátor, pontosabban egy „hama Supreme 20HD” újratölthető akkumulátor, amely képes a Raspberry számára szükséges 5 Volt feszültség és 3 Amper áramerősség biztosítására USB Type C csatlakozón keresztül.

4.2. Operációs rendszer

Raspberry Pi OS (korábban Raspbian): Ez a Raspberry Pi hivatalos operációs rendszere, és a legtöbb támogatott eszköz és szoftver elérhető rá. A libpcap telepítése és használata viszonylag egyszerű, és a készített programokhoz könnyen hozzáférhet a Raspberry Pi OS csomagkezelőjével. A Raspberry Pi OS a Debian alapú Linux disztribúcióra épül, így a Debian Linux előnyeit kínálja, és a Raspberry Pi számára optimalizált.

A rendszer alapértelmezett grafikus felhasználói felülete a Pixel (Pi Improved Xwindows Environment Lightweight) nevet viseli, amely könnyen használható és energiatakarékos.

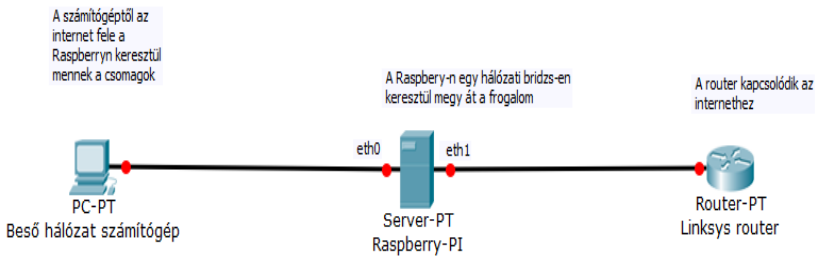
A rendszer használja a Debian csomagkezelő rendszerét, amely lehetővé teszi könnyű telepítést és frissítést. A csomagkezelő rendszer segítségével könnyedén telepíthetők és frissíthetők a szükséges szoftverek és könyvtárak, például a libpcap is. A Raspberry Pi OS folyamatosan frissül, így mindig az a legújabb kernel verziót tartalmazza, ami fontos a hardver támogatás és a biztonság szempontjából.

A rendszer specifikusan a Raspberry Pi számára lett tervezve, így kiválóan támogatja a Raspberry Pi eszközeit és hardver funkcióit. A Raspberry Pi OS alacsony erőforrás igénnyel rendelkezik, így kiválóan alkalmas a Raspberry Pi 4-es kis teljesítményű számítógéphez.

4.3. A rendszer tömbvázlata

A Rasbperry PI-t bekötjük a belső hálózat számítógépe és a router közé. A csomagok a belső hálózatról a Raspberry-n keresztül jutnak el a router felé. A router felől érkező csomagok a belső hálózat felé szintén áthaladnak a Raspberry-n.

A Raspberry az “eth1” illetve “eth0” ethernet interfész segítségével kapcsolódik a routerhez és a belső hálózathoz.



ábra 1 A rendszer tömbvázlata

4.4. Szoftver specifikációk

A program képes kell legyen elkapni és módosítani különböző hálózati csomagokat. Egyszerre akár két hálózati interfészen is dolgoznia kell, mivel nem csak a belső hálózatról kimenő csomagokat, de a router felöli csomagokat is el kell kapnia, és bizonyos esetekben módosítania is.

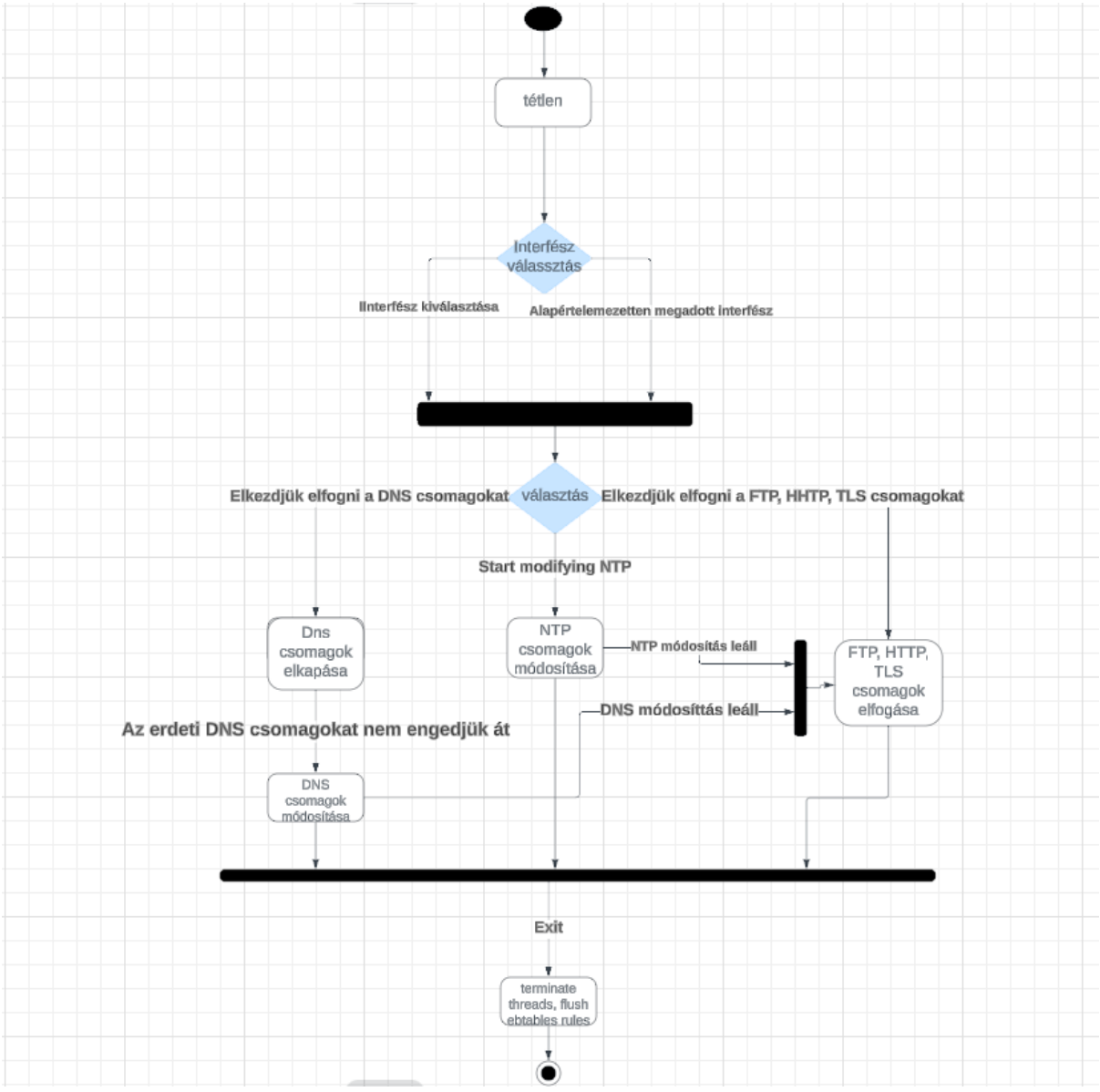
1. A DNS módosítás esetében a célunk az, hogy a DNS válasz IPv4 címét megváltoztassuk, és ezt a hamisított csomagot tovább küldjük a belső hálózat számítógépe fele.

2. Az NTP csomagok módosítása esetében azt szeretnénk elérni, hogy a belső hálózat számítógépének a rendszer idejét megváltoztassuk egy általunk meghatározott időpontra.
3. A rendszer grafikus felhasználói felülete egyszerű kell legyen dizájn szempontból, és erőforrás igénye kicsi. A gombok jól láthatóak, a feliratuk egyértelműen tükrözze a szerepüket.
4. A felhasználónak legyen lehetősége kiválasztani azokat a hálózati interfészeket, amelyeken a csomagokat megszeretné jeleníteni, illetve módosítani.
5. A felhasználónak legyen lehetősége bármikor leállítani a csomagok módosítását, illetve újra elindítani azt.
6. Amikor a felhasználó kilép az alkalmazásból, a hálózat álljon vissza abba az állapotba, amelyben az indítás elött volt.
7. Az elkapott csomagok kiíratásakor csak a szükséges adatokat írassuk ki a képernyőre.
8. A belső hálózat számítógépén az adatforgalom sebessége a lehető legkisebb mértékben csökkenjen.
9. Azok a hálózati csomagok, amelyeken nem módosítunk, feldolgozás nélkül át kell engedni. Azok a hálózati csomagok, amelyeket csak kiíratunk de nem módosítunk, késedelem nélkül átengedhetjük a Rasbpberry PI számítógépen.

5. Tervezés és megvalósítás

Az alábbi ábrán látható a rendszer állapotdiagramja.

A program elindítása után nem kezdődik rögtön a csomagok elfogása, illetve módosítása. A felhasználónak először kikell választania, hogy mely hálózati interfészeken akarja elkapni a csomagokat, ha nem választ interfészt, de mégis megpróbálja elindítani akkor a Rasbperry „eth0” illetve „eth1” interfészeit használja alapértelmezetten.



ábra 2 A rendszer állapotdiagramja

5.1. Hálózati Híd (Bridge) Konfiguráció:

A Raspberry két hálózati interfész között átjátssza a csomagokat és közben le is menti azokat.

Ahhoz, hogy ezt elérjük a Raspberry PI rendszerben az alábbi beállítást kell megvalósítani.

1. *ip link add name br0 type bridge* Ez a parancs létrehozza a br0 nevű hálózati bridzset a típusa bridge alapján. A bridge egy olyan eszköz, amely képes egyesíteni két vagy több hálózati interfészt.
2. *ip link set dev br0 up* Ez a parancs engedélyezi és aktiválja a létrehozott br0 hálózati bridzset (up állapotba helyezi). Így a bridzset használni lehet a hálózati kapcsolatokhoz.
3. *ip link set eth0 up* Ez a parancs engedélyezi és aktiválja az eth0 nevű Ethernet interfészt (up állapotba helyezi), amelyet később csatlakoztatni fogunk a hálózati bridzshez.
4. *ip link set eth0 master br0* Ez a parancs csatlakoztatja az eth0 Ethernet interfészt a br0 hálózati bridzshez, így az eth0 interfész a bridzsen keresztül lesz elérhető.
5. *ip link set eth1 up* Ez a parancs engedélyezi és aktiválja az eth1 nevű Ethernet interfészt (up állapotba helyezi), amelyet szintén csatlakoztatni fogunk a hálózati bridzshez.
6. *ip link set eth1 master br0* Ez a parancs csatlakoztatja az eth1 Ethernet interfészt a br0 hálózati bridzshez, így az eth1 interfész is elérhetővé válik a bridzsen keresztül.
7. *bridge link* Ez a parancs listázza az összes bridzshez csatlakoztatott interfészt. Ezzel ellenőrizheted, hogy az eth0 és eth1 valóban csatlakozik-e a br0 bridzshez.
8. *net.ipv4.ip_forward=1* Ez a parancs bekapcsolja az IPv4 csomagok továbbítását a Raspberry PI kernelben. Ez szükséges ahhoz, hogy a Raspberry PI hálózati forgalmat tudjon átjátszani az egyik interfészről a másikra.

Ezen parancsok végrehajtásával a Raspberry Pi hidat hoz létre, amely lehetővé teszi az áthaladó hálózati csomagok olvasását, és engedélyezi azok továbbítását az egyes interfészek között a megfelelő konfigurációval.

5.2. DNS csomagok módosítása.

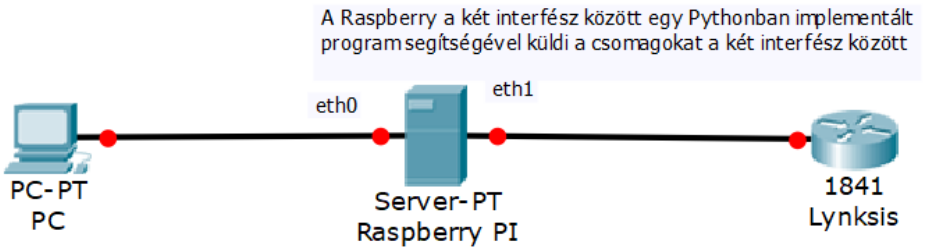
A cél az, hogy a hídon áthaladó hálózati csomagok közül elkapjam a DNS csomagokat, és módosítsam ezeket. Pontosabban a DNS válaszokat szeretném módosítani, úgy, hogy a válaszban szereplő IPv4 IP címet kicserélem egy általam választott címre. A DNS szerver megkapja a kéréseket módostás nélkül a vissza érkező legitim válaszban módosítom az IP címet nem egy teljesen új csomagot generálok.

5.3. Csomagok elfogása hálózati híd használata nélkül

Hálózati híd használata helyet megpróbáltam más módszerrel átjátszani a hálózati forgalmat, a Raspberry PI két interfésze között.

Az ötlet az, hogy a két interfész közötti adatátvitelt teljes mértékben Python kóddal (Scapy segítségével oldjam meg). Ez azért jó, mert így a Raspberry PI jelenléte észrevehetetlen lenne a hálózaton, és könnyen tudnám befolyásolni azt, hogy mely csomagokat engedjük tovább menni.

A kartya1 és kartya2 függvények a két különböző interfészhez tartozó szálakat definiálják. Ezek folyamatosan figyelik az adott interfészen érkező Ethernet kereteket, és ezeket módosítják és továbbítják.



ábra 3 Hálózati forgalom híd nélkül

Az „if __name__ == "__main__"” részben létrehozzuk és elindítjuk a két szálat (t1 és t2), majd megvárjuk, amíg mindkettő befejeződik (join). Ez biztosítja, hogy a program ne álljon le azonnal a szálak indítása után.

Ahhoz, hogy ezt elérjük tudnunk kell a belső hálózat gépének a hálózati interfészének a MAC címét, valamint a külső hálózat (Link sys) hálózati interfészének a MAC címét.

Az eth0mac és eth1mac változók értékét a „get_if_hwaddr” függvény segítségével állítjuk be. Ezek a változók a megadott interfész MAC-címeit tárolják.

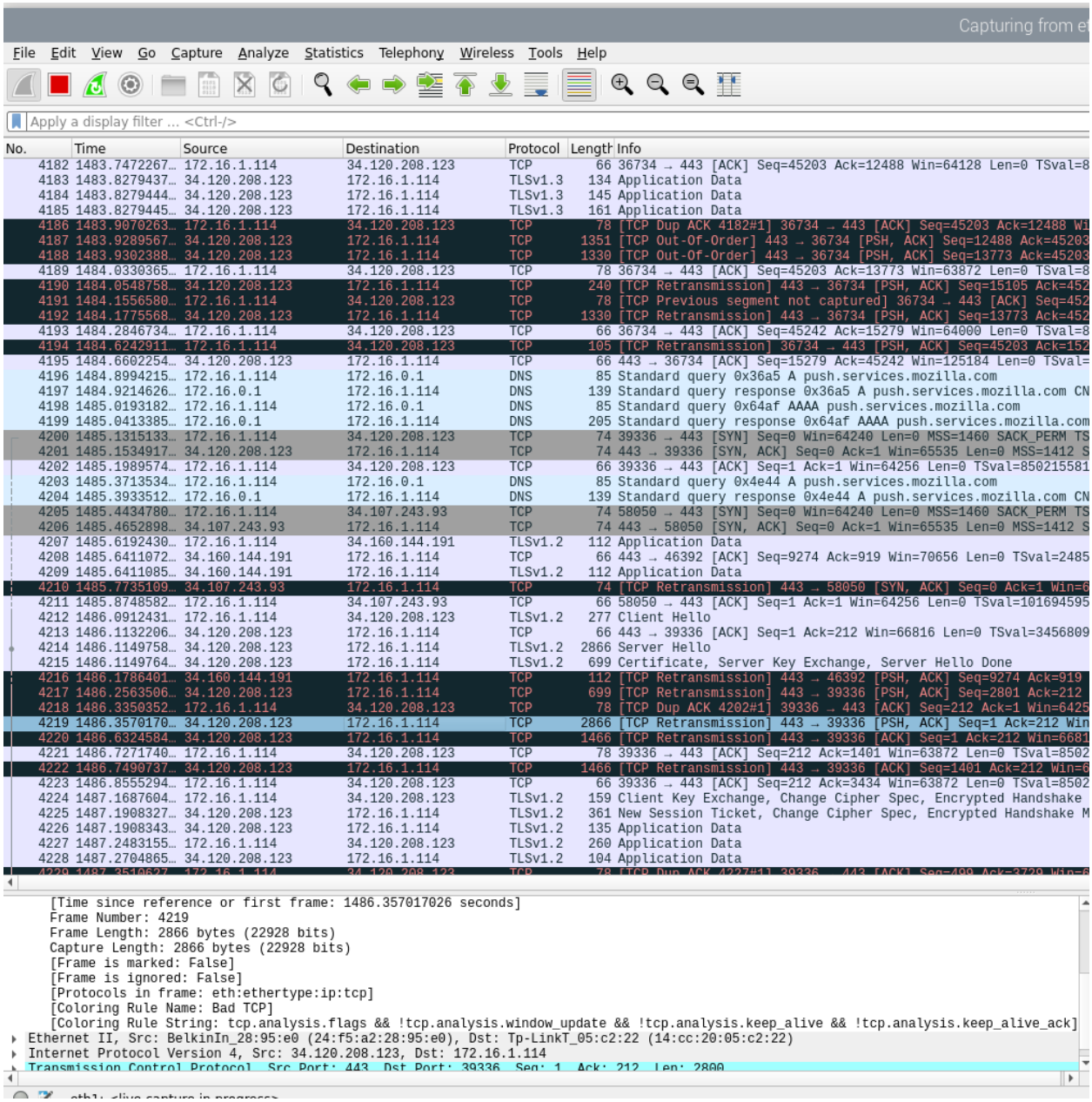
A „get_if_hwaddr("eth0")” függvényhívás az "eth0" nevű hálózati interfészhez tartozó MAC-címet kéri le.

Ezeket a MAC-címeket később használjuk fel a hálózati csomagok forrás- és cél-címének módosításához a „sendp” függvényben. A módosított csomagok elküldésekor ezek a MAC-címek fogják jelenteni a csomag forrás- és cél-címét az Ethernet keret szintjén.

Amikor például a lynksys router felöli hálózati interfészt figyeljük a Raspberry PI lapon, olyankor az adat szóró (broadcast) csomagokat módosítás nélkül átengedjük anélkül, hogy módosítanánk a cél MAC címen. Máskülönben az összes csomag cél MAC címét megváltoztatjuk a belső hálózat gépének hálózati interfész MAC címére. E mellett még a forrás MAC címet mindig meg kell változtatni A Raspberry PI azon hálózati interfészének a MAC címére, amely a belső hálózathoz csatlakozik. Hasonlóan járunk el a másik interfész esetén is.

A probléma ezzel a megoldással az volt, hogy az összes hálózati csomag módosítása jelentősen lelassította az adatforgalmat a belső hálózat számítógépén. Emellett még több abnormális-hibás csomag is létrejött, amelyek tartalma meghaladta az MTU-t, ezért végül ez a megoldás nem bizonyult sem elég hatékonynak, sem elég megbízhatónak, úgyhogy végül új megoldás után kellett néznem.

Az alábbi ábrán láthatóak példa azokra a csomagokra, amelyeket Wireshark segítségével sikerült elfognom. Ha megvizsgáljuk a csomag „Frame length” mezejét azt vehetjük észre, hogy az jóval meghaladta a Rasbperry 1500 byte-os maximális MTU-ját. Ezeknek az abnormális csomagoknak a létrejöttére nem sikerült pontos magyarázatot találnom.



ábra 4 hibás csomagok Wireshark-ban

5.4. DNS csomagok blokkolása ebtables-el

Annak érdekében, hogy az eredeti DNS csomagok ne tudjanak eljutni a célhoz, ebtables szabályokat használtam.

A „*ebtables -A FORWARD -p IPv4 --ip-protocol udp --ip-dport 53 -j DROP*” és az „*ebtables -A FORWARD -p IPv4 --ip-protocol udp --ip-sport 53 -j DROP*” parancs segítségével blokkolhatjuk a DNS csomagokat az áthidalt hálózaton. Amíg a fenti parancsok segítségével nem állítjuk be az ebtables szabályokat az nem tudjuk módosítani a DNS csomagokat, viszont kiírathatjuk azokat.

Mivel a Scapy segítségével hamarabb elfoghatjuk a hálózati csomagokat, mintsem azokat eldobjuk az ebtables szabályok segítségével, megtehetjük azt, hogy csak azokat a DNS csomagokat engedjük át, amelyeket mi szeretnénk.

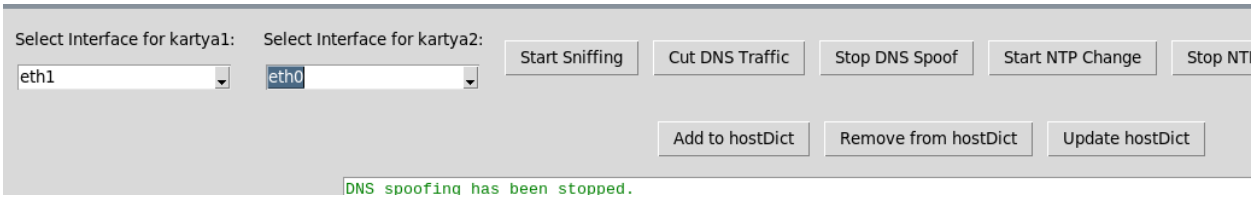
Elindítunk két szálat, egy-egy szál mindkét hálózati interfész számára (meg a főszál UI számára). Azon az interfészen, amelyen a DNS válaszok érkeznek (linksys felől), módosítások nélkül átengedjük a DNS csomagokat a belső hálózat felőli interfész fele. Ezen kívül elküldjük a szükséges információkat a UI számára.

Amikor a belső hálózat interfészét figyeljük általában DNS kéréseket kapunk el. Ha a DNS kérésben olyan neveket kapunk el, amelyek tartalmazzák a szótárban lévő neveket, akkor az eredeti kérést tovább küldjük, a választ elfogjuk módosítjuk az IP mezőt, és majd azt küldjük vissza.

```
Ether / IP / UDP / DNS Qry b'google.com.'
Ether / IP / UDP / DNS Ans 216.58.212.174
Ether / IP / UDP / DNS Qry b'joco.ro.'
MODIFIED: Ether / IP / UDP / DNS Ans 8.8.8.8
Ether / IP / UDP / DNS Ans 8.8.8.8
Ether / IP / UDP / NTP v4, client
MODIFIED: Ether / IP / UDP / NTP v4, server
Ether / IP / UDP / DNS Ans 162.55.5.235
Ether / IP / UDP / DNS Qry b'connectivity-check.ubuntu.com.'
Ether / IP / UDP / DNS Ans 2620:2d:4000:1::2a
```

ábra 5 DNS csomag módosítás

A szótárat lokálisan tároljuk egy JSON fájlban. A szótár tartalmát a felhasználó módosíthatja. Hozzáadhat új DNS név és cím párokat, módosíthatja a már meglévő elemeket, valamint törölhet az elemek közül.



ábra 6 DNS szótár módosító elemek

Ha a név nem szerepel a szótárban akkor elküldjük a DNS csomagot módosítások nélkül.

Az „*eth = Ether(src=packet[Ether].dst, dst=packet[Ether].src)*” az Ethernet keret forrás és cél MAC címét megcseréli.

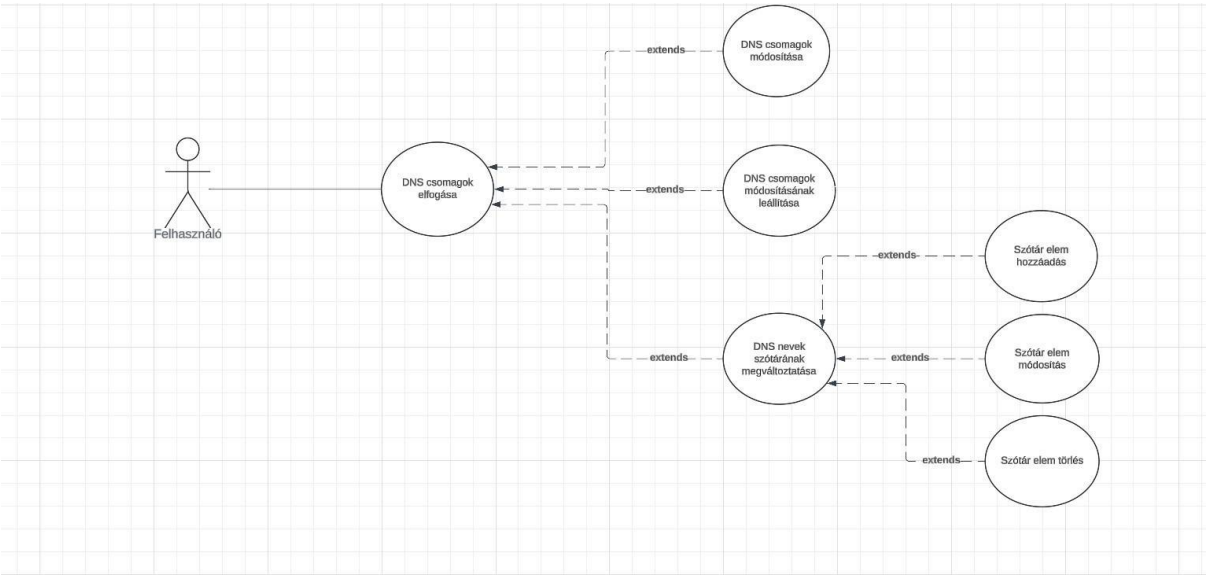
Az „*ip = IP(src=packet[IP].dst, dst=packet[IP].src)*” az IP csomag forrás és cél IP címét megcseréli.

Az „udp = UDP(dport=packet[UDP].sport, sport=packet[UDP].dport)” az UDP csomag forrás és cél portját megcseréli.

A DNS válasz, a DNS kérés csomag azonosítóját (id), kérését (qd) tartalmazza. Az aa (authoritative answer) mezőt igazra (1) állítjuk, jelezve, hogy a válasz hiteles forrásból származik. Az ar (válasz/answer) mezőbe bekerül a szótárban található IP cím. A szótár működésének kódrészlete megtalálható a függelékben.

Az Ethernet, IP, UDP és DNS rétegeket összefűzzük egy csomaggá, és elküldjük a belső hálózat interfészen keresztül. A módosított csomag információit egy sorba (packet_queue) helyezzük.

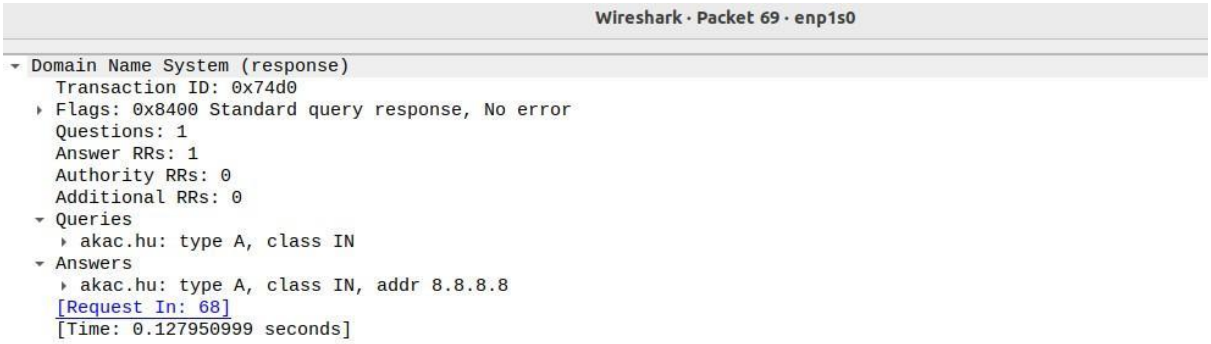
Alább látható a DNS módosításnak a “use case” diagramja.



ábra 7 DNS módosítás „use case” diagram

Ez a megoldás sokkal hatékonyabb, mint az összes hálózati csomag módosítása és elküldése Scapy-vel.

A belső hálózat számítógépén elfogott csomagban látszik, hogy a DNS válasz IP cím értéke az „akac.hu” weboldal esetében 8.8.8.8.



ábra 8 Wireshark DNS csomag

A “dig @8.8.8.8 -t a akac.hu” parancs kiadásával terminálból szintén láthatjuk, hogy a weboldal IP címe 8.8.8.8.

```
diak@313x8:~$ dig @8.8.8.8 -t a akac.hu

; <<>> DiG 9.18.18-0ubuntu0.22.04.2-Ubuntu <<>> @8.8.8.8 -t a akac.hu
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 29904
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;akac.hu.                       IN      A

;; ANSWER SECTION:
akac.hu.                600     IN      A       8.8.8.8

;; Query time: 127 msec
;; SERVER: 8.8.8.8#53(8.8.8.8) (UDP)
;; WHEN: Thu Jun 20 14:22:15 EEST 2024
;; MSG SIZE  rcvd: 48
```

ábra 9 dig parancs

5.5. DNS csomagok módosítása Scapy segítségével

A DNS csomagok módosítására és újra küldésére a Scapy Python könyvtárat használtam.

Az alábbi ábrán látható Python kóddal figyeljük a hálózati forgalmat, észleli a DNS kéréseket, módosítja azokat, majd visszaküldjük a hálózatra a módosított csomagokat.

A csomagokat a Raspberry PI eth0 hálózati interfészén kapjuk el, amelyhez a belső hálózat számítógépe csatlakozik. Az elfogott csomagokat megszűrjük úgy, hogy csak az 53 port UDP és TCP csomagokat kapjuk el. Minden elfogott csomagra meghívjuk a „dns_packet_callback” függvényt, amely a módosításokat és a csomagok elküldését végzi.
A DNS kérések neveit "ogs.google.com"-ra változtatjuk. Ha DNS válaszokat kapunk el, azokat pedig úgy módosítjuk, hogy azok mindig az '8.8.8.8' IP-címet adják vissza. Ezt a „packet[DNS].an.rdata” mező megváltoztatásával érjük el.
Az ellenőrző összegek és csomag hossz újra kalkulálása biztosítja, hogy a módosított csomagok érvényesek legyenek. Azzal, hogy töröljük a „cheksum” és „len” értékeket a csomag elküldése előtt a Scapy újra számolja azokat, a csomag tartalma alapján. Az új értékek így már figyelembe veszik a változtatásokat is. Enélkül az elküldött csomagokat „malformed” csomagokként észlelné, és nem venné figyelembe a válaszokat.

Ezzel a megközelítéssel az a probléma, a DNS csomagokat valójában nem „fogjuk el”, hanem csak megnézzük, megváltoztatjuk és egy új, megváltozott csomagot elküldjük, de az eredeti csomagot nem állítjuk meg. Mivel az eredeti DNS csomag elmegy a módosított csomag előtt, nincs rá esély, hogy a meghamisított csomagot figyelembe vegyék. A Scapy-vel nem lehetséges a hálózati csomagok megállítása, eldobása, erre más eszközt is kell alkalmaznom.

5.6. DNS csomagok módosítása Scapy és Netfilterqueue segítségével

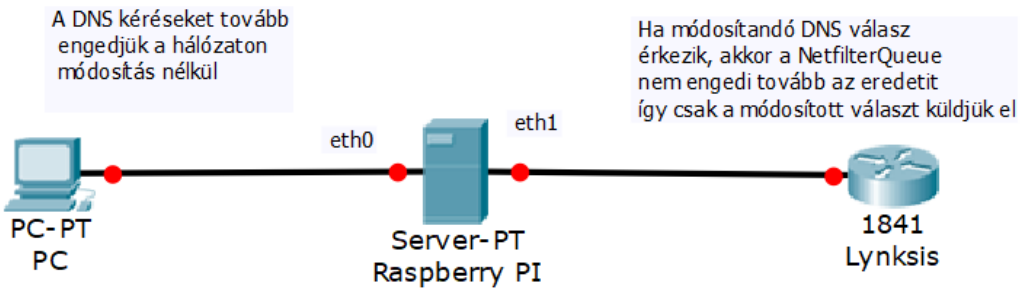
Egy harmadik próbálkozás figyeli a hálózati forgalmat, észleli a DNS válaszokat, és szükség esetén módosítja a DNS válasz IP-címét az előre meghatározott értékekre. Az Iptables szabályok beállításával a csomagokat átirányítja az NFQUEUE-ba, ahol a NetfilterQueue kezeli a csomagokat. A Scapy segítségével módosítom a csomagokat, majd visszaállítom őket a hálózatba.
A „callBack” metódus a DNS válasz csomagok elfogását és módosítását végzi.
A callBack metódus a következő lépéseket hajtja végre:

- Elkapja a hálózati csomagot és Scapy formátumra alakítja.
- Ellenőrzi, hogy a csomag DNS válasz csomag-e.
- Ha igen, akkor ellenőrzi, hogy a DNS kérés név szerepel-e a megadott szótárban (hostDict).
- Ha a név szerepel a szótárban, akkor módosítja a DNS válasz IP-címét a megadott új értékre.
- Törli az IP és UDP rétegek hossz és ellenőrzőösszeg mezőit, hogy Scapy újrakalkulálja azokat.
- Visszaállítja a módosított csomagot nyers formátumba és elfogadja azt.

A „packet.get_payload()” a NetfilterQueue által kapott nyers csomagot adja vissza.

Az „IP(packet.get_payload())” a nyers csomagot Scapy IP csomaggá alakítja. Ezáltal hozzáférhetünk az IP réteghez és annak alrétegeihez (például UDP és DNS).

Az „__init__” metódus a DnsSnoof osztály inicializálásáért felelős. Ez a metódus beállítja azokat az alapvető paramétereket és objektumokat, amelyek szükségesek a DNS csomagok módosításához és kezeléséhez.



ábra 10 DNS csomagok módosítása Netfilterqueue segítségével

A „__call__” metódus a DnsSnoof osztályt hívja meg függvényként. Ez a metódus felelős a DNS csomagok elfogásának és kezelésének elindításáért.

Az „os.system(f'iptables -I FORWARD -j NFQUEUE --queue-num {self.queueNum}')” meghívja az iptables parancsot a megfelelő szabályok beállítására. Az „ -I FORWARD -j NFQUEUE --queue-num {self.queueNum}” paraméterek azt jelentik, hogy az összes FORWARD irányba tartó csomagot az NFQUEUE-be irányítjuk át, ahol a sor száma a queueNum érték lesz.

A „self.queue.bind(self.queueNum, self.callBack)” összekapcsolja a queueNum sor számot a callBack metódussal, amely az átadott csomagokat fogja kezelni.

A „self.queue.run()” elindítja a NetfilterQueue futását, ami a csomagok elfogását és kezelését teszi lehetővé.

Az „except KeyboardInterrupt” elkapja a felhasználó által küldött KeyboardInterrupt kivételt, ami akkor dobódik, ha a felhasználó megszakítja a folyamatot (pl. Ctrl+C lenyomásával).

Az „os.system(f'iptables -D FORWARD -j NFQUEUE --queue-num {self.queueNum}')” meghívja az iptables parancsot a korábban beállított szabályok törlésére.

A „log.info("[!] iptable rule flushed")” naplóba írja a "[!] iptable rule flushed" szöveget, ezzel jelzi a szabályok törlését.

A „set_payload” és a „return packet.accept()” sorok (6. Ábra) a NetfilterQueue-hoz tartoznak, és a módosított csomag visszaadását és elfogadását végzik

A „packet.set_payload(bytes(scapyPacket))” beállítja a csomag tartalmát a módosított scapyPacket-re. A NetfilterQueue-ban a csomagokat byte formátumban kell visszaállítani, miután Scapy-vel módosítottuk őket.

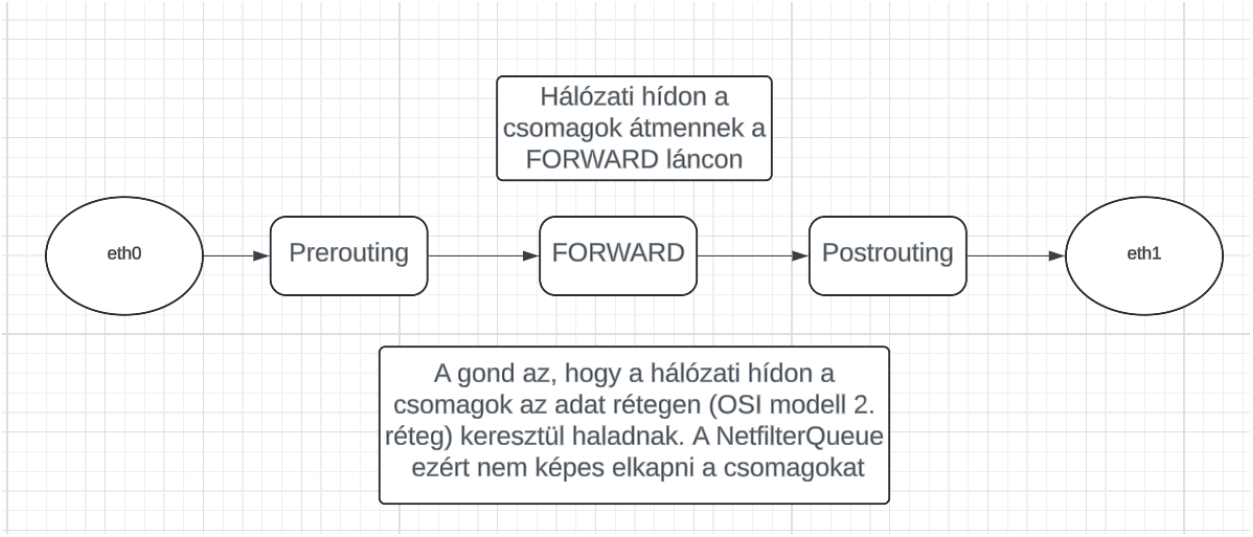
A „return packet.accept()” sor visszaadja a módosított csomagot a NetfilterQueue-nak, és elfogadja azt. Ezután a csomag továbbításra kerül a hálózatban. A NetfilterQueue ezt követően a következő csomagot fogja feldolgozni, ha van ilyen. Ez a sor biztosítja, hogy a módosított csomagok tovább haladhassanak a hálózaton.

A „hostDict” változó egy szótárat tartalmaz, amely párosítja a DNS neveket azokkal az IP-címekkel, amelyekre módosítani kívánod őket. Itt a kulcsok byte stringek (b"google.com", b"facebook.com."), az értékek pedig stringek ("8.8.8.8"), amelyek az adott domainekhez tartozó új IP-címeket jelölik.

Ez a szótár például azt mondja, hogy ha valaki a "joco.ro" vagy az "akac.hu" domain nevet keresi fel, a válaszban megadott IP-cím helyett mindkét esetben a "8.8.8.8" IP-címet fogja kapni.

A DNS válaszok módosítását csak akkor végezzük, ha a DNS csomagban olyan név szerepel, amely benne van a létrehozott szótárban.

A scapyPacket[DNS].an = DNSRR(rrname=queryName, rdata=self.hostDict[queryName])” sorban létrehozunk egy új DNS válasz rekordot (DNSRR objektumot), amely a kérdezett domain nevet (rrname) és az azt megfeleltetett új IP-címet (rdata) tartalmazza a hostDict-ből. Ez a sor a DNS válasz rekord (an vagy "answer") részének módosítását jelenti a Scapy scapyPacket objektumán belül. Így a válasz az új IP-címet fogja tartalmazni a kérdezett domain név helyett.



ábra 11 Hálózati csomagok útvonala a hídon keresztül

A probléma ezzel a próbálkozással az volt, hogy a NetfilterQueue csak olyan csomagokat képes elfogni, amely érvényesek az iptables szabályok, viszont a hídon (bridge) átmenő csomagokra ebtables szabályok érvényesek, így ezzel a módszerrel sem értem el a kívánt eredményt.

5.7. NTP csomagok módosítása.

A cél az, hogy elfogjam, módosítsam majd a módosított NTP csomagokat tovább küldjem. Az NTP kliens csomagot módosítások nélkül tovább engedem a szerver fele. A választ viszont módosítom, így a belső hálózat gépén azt kell lássuk, hogy az nem a helyes időt mutatja, hanem az általam megadottat.

5.7.1. NTP csomagok módosításának megvalósítása

Az előzőkéből tanulva a DNS csomagok módosításához hasonlóan, itt is ebtables szabályok segítségével megállítom az NTP csomagok forgalmát a hálózati bridzsen.

A „*ebtables -A OUTPUT -p IPv4 --ip-proto udp --ip-dport 123 -j DROP*” parancs blokkolja a kimenő UDP csomagokat, amelyek cél portja 123, megakadályozva ezzel az NTP kérések küldését a hálózatról.

- “ebtables”: Az Ethernet keretek szűrésére használt parancs.
- “-A OUTPUT”: Az OUTPUT lánchoz (kimenő forgalom) adunk hozzá egy szabályt.
- “-p IPv4”: A protokoll IPv4-et határozza meg.
- “--ip-proto udp”: Az IP protokoll típusa UDP.
- “--ip-dport 123”: A cél UDP port 123 (NTP).
- “-j DROP”: Az ilyen csomagokat eldobja.

Majd az “ *ebtables -A OUTPUT -p IPv4 --ip-proto udp --ip-sport 123 -j DROP*” szintén blokkolom az NTP csomagokat, de ezúttal azokat szűröm ki, amelyek forrás (sport) portja 123. Ezek azok a csomagok amelyek az NTP szervertől jönnek válaszként a kliens kérésére.

Ezek a szabályok biztosítják, hogy a rendszer ne tudjon NTP szolgáltatást használni az idő szinkronizálására, sem kérés, sem válasz formájában.

Ezután Scapy segítségével elfogom az NTP csomagokat.

A “pkt = sniff(iface="eth0", filter="udp and dst port 123", store=1, count=1)” halgtózik az eth0 ethernet interfészen. Ezek a csomagok a belső hálózat felöl jönnek, így várhatóan NTP kliens kéréseket tartalmaznak. Ezeket módosítások nélkül tovább küldöm az eth1-es interfészre.

A “pkt = sniff(iface="eth1", filter="udp and src port 123", store=1, count=1)” halgatózik az eth1 ethernet interfészen és elkapja a szervertől jövő válaszokat. Pontosabban egyszerre egy csomagot kap el, amely azután feldolgozásra kerül.

Két ethernet interfész között átjátsszuk az NTP csomagokat.

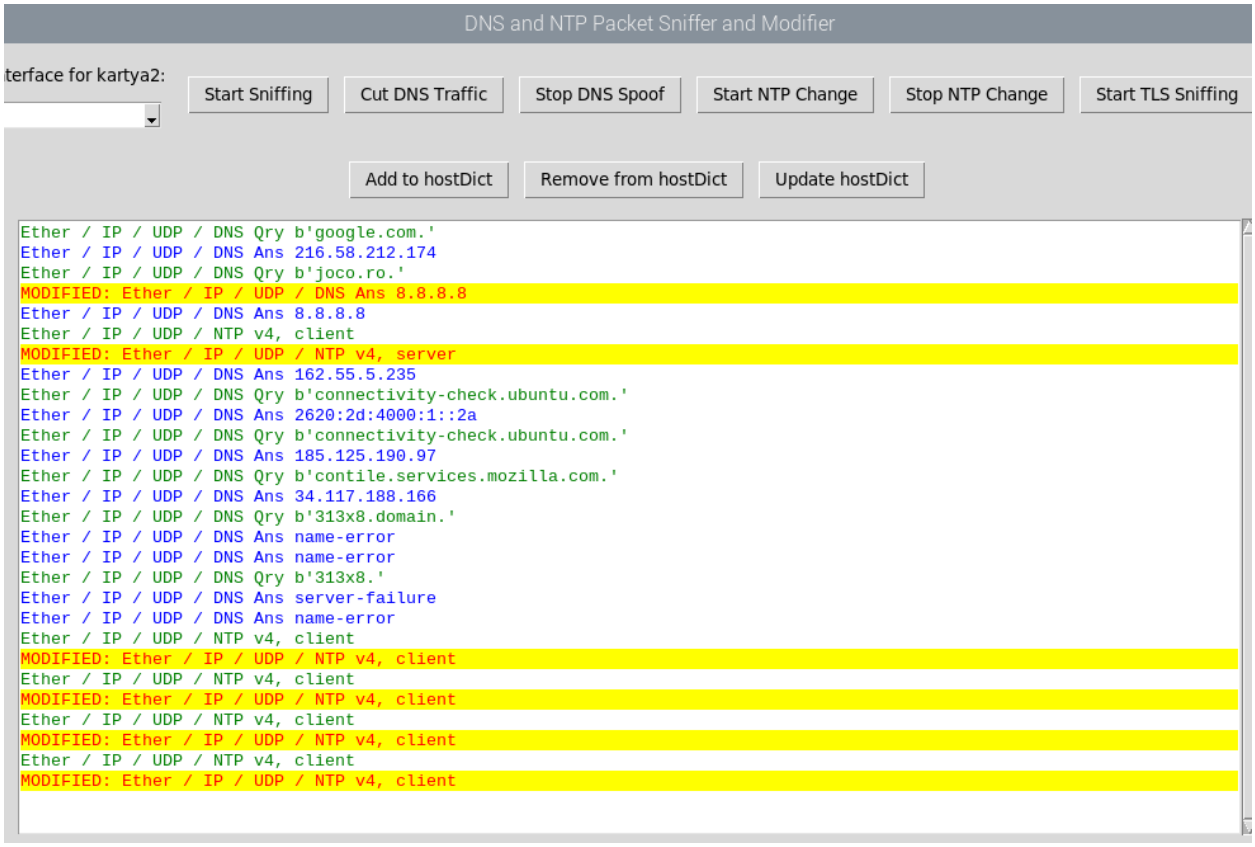
Az „if packet.haslayer(NTP)” feltétel ellenőrzi, hogy a csomag tartalmaz-e NTP réteget.

A packet[NTP].recv és packet[NTP].sent mezőket új értékekre állítja. Receive Timestamp (recv): Az időbélyeg, amikor a szerver megkapta a kérést. Transmit Timestamp (sent): Az időbélyeg, amikor a szerver elküldte a válaszát. A belső hálózat számítógépe ezeket a mezőket (és ezeken kívül még két mezőt ref, orig) fog felhasználni arra, hogy megállapítsa az időt.

Az IP és UDP rétegek len és chksum mezőit törli, hogy a Scapy újra számolja azokat a módosítás után.

Végül a módosított csomagokat tovább küldöm az eth0 interfész fele.

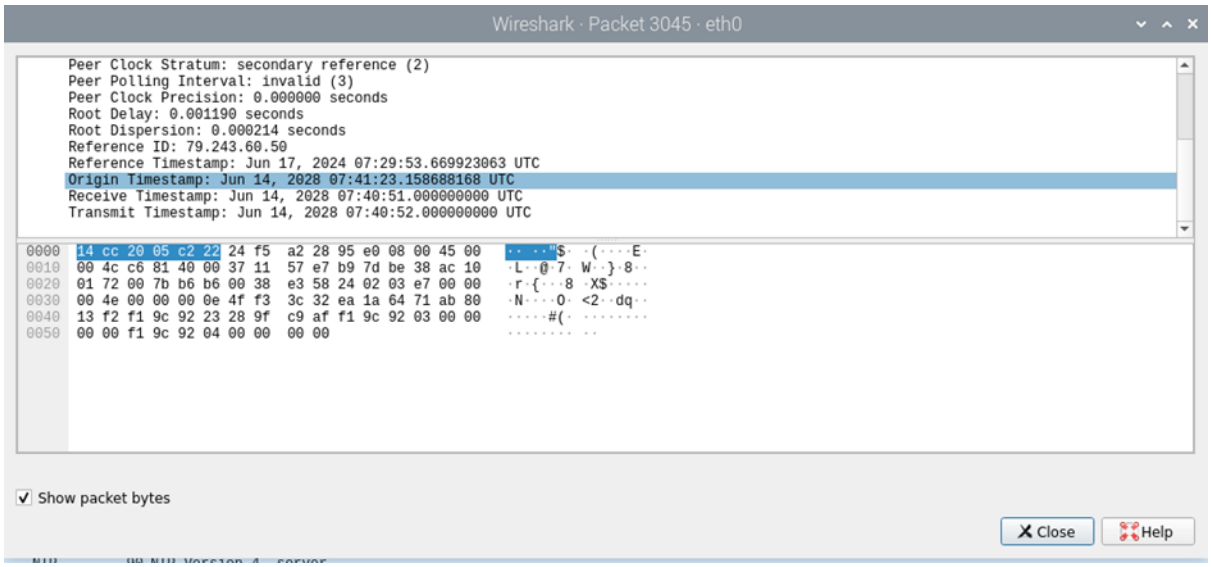
A belső hálózat gépén az időt 2028 Június 14. 10 óra 40 percre akartam beállítani. Ez az idő jóval a jövőben van, így amellett, hogy a számítógép órája átáll azt is észrevehetjük, hogy a weboldalak SSL tanúsítványai is lejártak a számítógép szemszögéből.



ábra 12 NTP csomagok módosítása

5.7.2. NTP csomagok módosítása eredményei

A Wiresahark segítségével elfogott NTP csomagokban jól látszanak a módosított mezők értékei.



ábra 13 módosított NTP csomag Wiresharkban

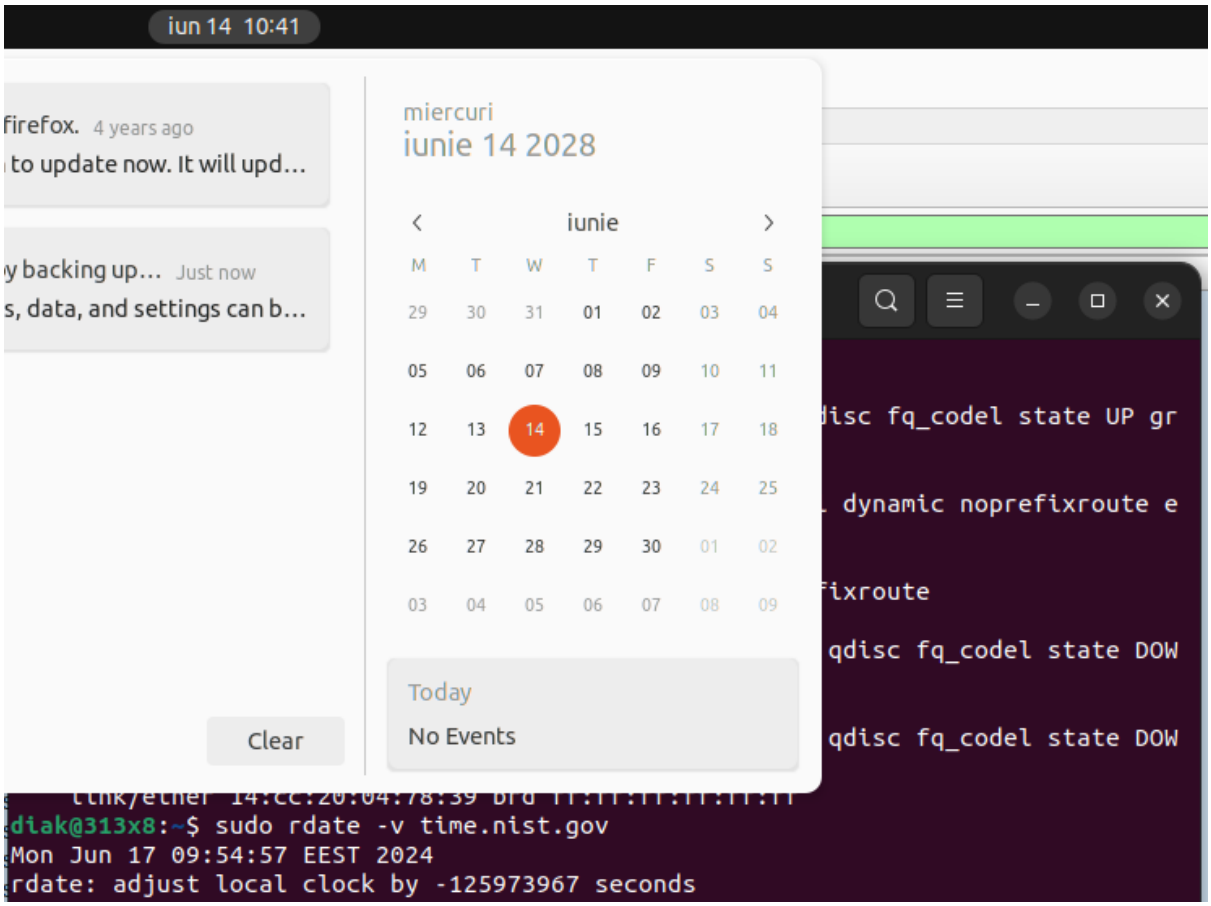
Ezek valóban azok az értékek, amelyeket látni szeretnénk a módosítások után.

Az rdate parancs az Ubuntu operációs rendszerben egy olyan eszköz, amely lehetővé teszi a rendszeridő szinkronizálását egy távoli időszolgáltató szerverrel az RFC 868 Time Protocol használatával. Mivel az rdate nem NTP protokollt használ ezért ez probléma nélkül letudja kérni a helyes időt a szerverektől.

Az rdate emellett az is megmutatja, hogy a rendszer ideje hány másodperccel tér el a valós időtől.

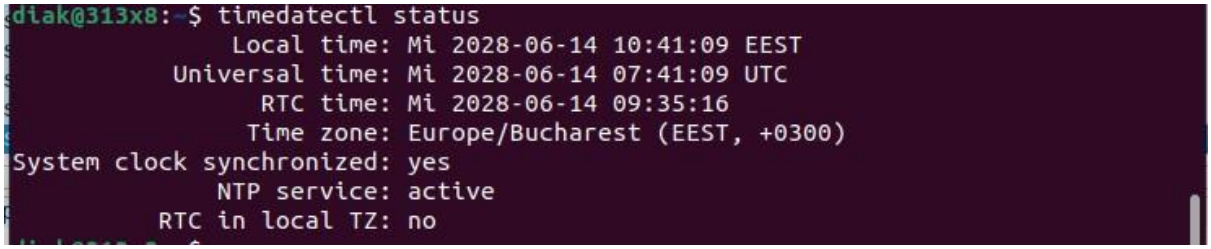
A belső hálózat gépe az idő szinkronizálására „systemd-timesyncd” használ, amely minden újraindításkor és utána meghatározott időközönként kéri le a szervertől az időt.

Az alábbi ábrán látszik a rendszer ideje, amelyet a „systemd-timesyncd” alított be és az „rdate” által lekért idő egymás mellett.



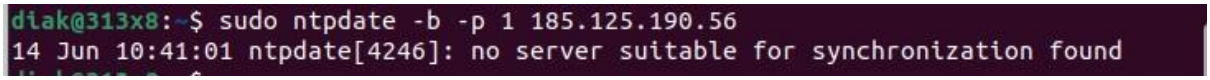
ábra 14 Rendszer idő és rdate által lekért idő

A timedatectl parancs segítségével is lekérhető a rendszer ideje amelyet a „systemd-timesyncd” állított be.



ábra 15 timedatectl lekérés

Az ntpdate parancs lehetővé teszi a rendszeridő szinkronizálását NTP (Network Time Protocol) szerverekkel. Telepítéséhez az alábbi parancsokat kell futtatni sudo apt-get update, sudo apt-get install ntpdate. Ntpdate segítségével lekérjük az időt a megadott szervertől. Az ntpdate nem fogadja el a szerverünket egy megbízható szerverként.



ábra 16 ntpdate lekérés

Ebben az esetben is az általunk megadott időt kapjuk, viszont fontos megjegyezni, hogy ezt az időt ebben az esetben nem a szervertől jövő NTP csomag alapján számítja ki. Mivel a szervert nem fogadja el valós/megbízható NTP szerverként ezért nem szinkronizálja az időt. Azért jelenik meg a módosított idő, mert a rendszer idejét jeleníti meg ebben az esetben.

Látható, hogy annak ellenére, hogy a rendszer ideje megváltozott az ntpdate mégsem fogadja el a szervert, ha módosítottuk az NTP csomagok mezőit.

Ez valószínűleg azért van így, mert egyszerre túl nagy lépéssel a rendszer idejét. Ezzel átlépjük az úgynevezett „pánik küszöböt” [15].

5.8. TLS, HTTP, FTP csomagok elfogása (forgalom monitorizálás)

A TLS csomagok esetében a cél nem az, hogy ezeket módosítsam, hanem csak bizonyos mezők értékeinek megjelenítése.

Mivel az eszközt elsősorban hálózatok labor órák során, demonstráció céljából fogják használni, a cél nem az, hogy olyan részletességgel jelenítsük meg az adatokat, mint például a Wireshark,

hanem csak azt a részét a csomagnak, amelyek relevánsak egy labor vagy hálózati demo bemutatásakor.

A megadott interfészen elindítjuk a Scapy „sniff” függvényét. Ezzel elkaphatjuk a hálózati hídon áthaladó összes csomagot. A csomagok közül ebben az esetben csak azok érdekelnek minket, amelyek tartalmaznak TLS protokollt.

A TLS rétegben sem érdekel minket az összes csomag. Azokra vagyunk kíváncsiak, amelyek a TLS kézfogás „handshake” részei.

Tudjuk azt, hogy azok a csomagok tartalmaznak a kézfogásról információt, amelyek „handshake” típusúak, viszont amikor a „type” mezőt kiolvassuk a csomagból nem egy szót, hanem egy szám értéket kapunk vissza. Ezt a számértéket a Scapy „show()” metódusa képes feloldani, viszont egyetlen mezőre nem hívható ez meg. Emiatt képtelenek vagyunk ezt a szám értéket felhasználni annak érdekében, hogy kiszűrjük a kívánt csomagokat.

Miután sikerült a kívánt csomagok megszűrése következik a csomag mezőinek a feldolgozása.

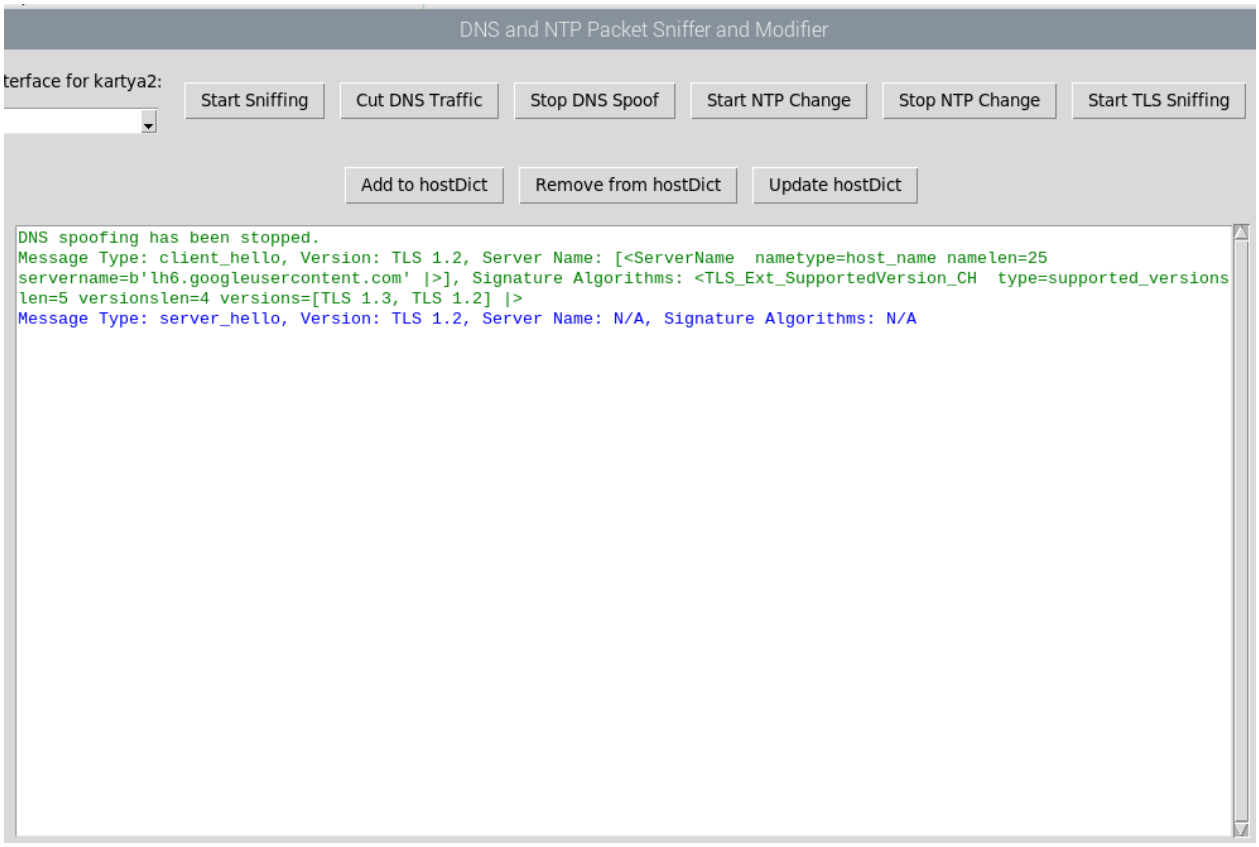
Első lépésben üres „N/A” -vel inicializáljuk azokat a string változókat, amelyek a keresett mezők string értékeit fogják tárolni.

A mezők közül kell nekünk a „msgtype”, amely információt tartalmaz az üzenet típusáról, ez jelzi, például azt hogy az üzenet egy „client-hello” vagy „server-hello” egy kézfogás során.

Kiszedjük még a szerver nevét, amellyel a kézfogás történik, a kliens által támogatott aláíró algoritmusokat, valamint a TLS verziót is.

Eközben megtörténik a „servername” és a „sig_algs” mezők string-é alakítása, valamint a „version” és a „msgtype” szám értékeinek megfeleltetése, felhasználva a megfelelő „tls_version” valamint „tls_message_type” szótárakat. Ezekben a szótárakban tároljuk el a szám értékeknek megfelelő szavakat.

Végül a mezők string értékeit összefűzzük, és beletesszük a „tls_queue” sorba, amely majd a feldolgozásra kerül és a tartalma megjelenik a felhasználói felületen.

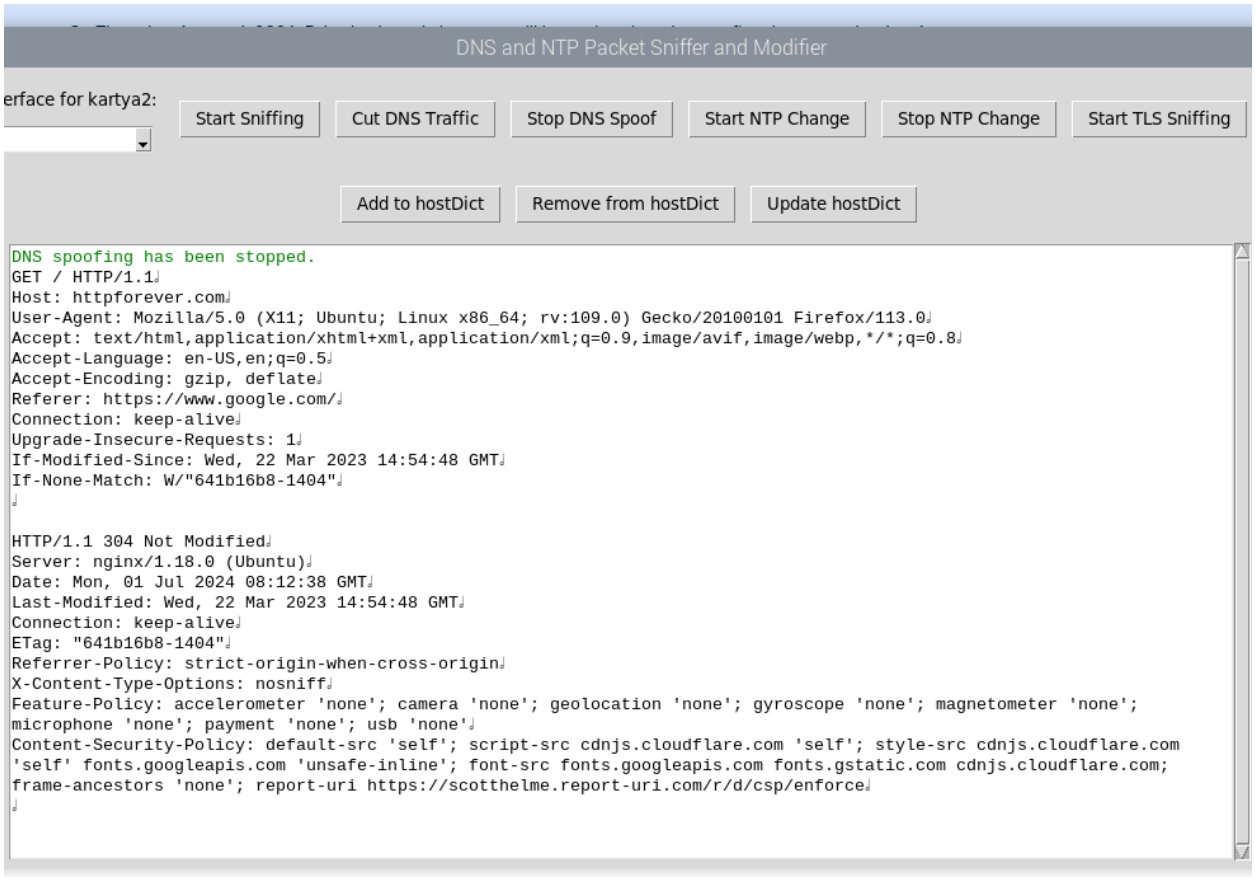


ábra 17 TLS csomagok megjelenítése

A TLS csomagokhoz hasonlóan a http és FTP csomagok esetében is szeretnénk megjeleníteni bizonyos információkat.

Az FTP csomagokból kiíratjuk a szerverhez kapcsolódásához szükséges felhasználónevet és jelszót, mivel ezek nincsenek titkosítva.

A HTTP csomagok esetében megjelenítjük a HTTP fejléc tartalmát.



ábra 18 HTTP csomagok megjelenítése

6. Mérések következtetések

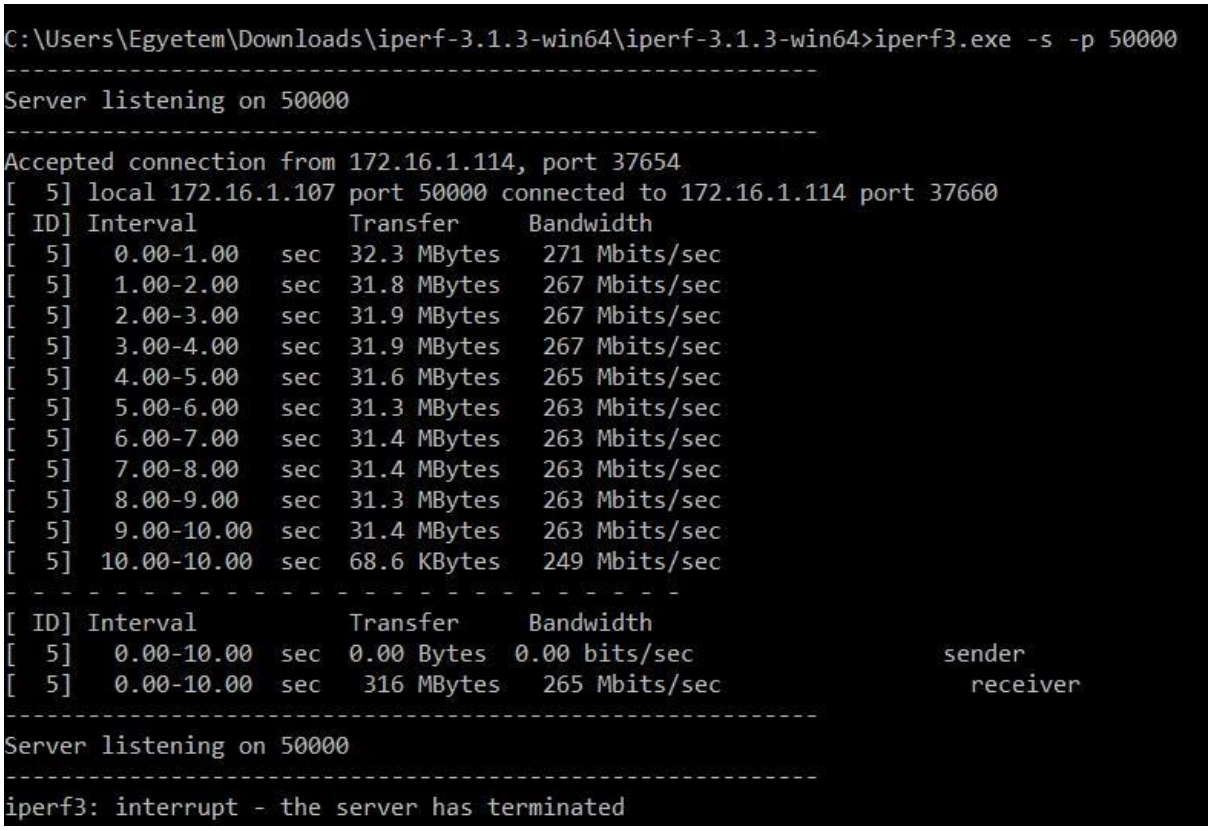
Hálózati híd nélkül átjátszani a csomagokat a Rasbperry PI két interfésze között nem bizonyult hatékony megoldásnak.

Ahogy az alábbi, iperf3-al elvégzett, mérésen is látszik átlagban 33.7 Kbits/sec volt a sávszélessége a hálózatnak.

```
C:\Users\Egyetem\Downloads\iperf-3.1.3-win64\iperf-3.1.3-win64>iperf3.exe -s -p 50000
-----------------------------------------------------------
Server listening on 50000
-----------------------------------------------------------
Accepted connection from 172.16.1.114, port 53780
[  5] local 172.16.1.107 port 50000 connected to 172.16.1.114 port 53792
[ ID] Interval           Transfer     Bandwidth
[  5]   0.00-1.01   sec  0.00 Bytes  0.00 bits/sec
[  5]   1.01-2.02   sec  0.00 Bytes  0.00 bits/sec
[  5]   2.02-3.02   sec  0.00 Bytes  0.00 bits/sec
[  5]   3.02-4.01   sec  4.28 KBytes  35.1 Kbits/sec
[  5]   4.01-5.00   sec  7.13 KBytes  59.0 Kbits/sec
[  5]   5.00-6.01   sec  0.00 Bytes  0.00 bits/sec
[  5]   6.01-7.01   sec  9.98 KBytes  82.0 Kbits/sec
[  5]   7.01-8.00   sec  7.13 KBytes  58.7 Kbits/sec
[  5]   8.00-9.00   sec  5.70 KBytes  46.5 Kbits/sec
[  5]   9.00-10.01  sec  5.70 KBytes  46.7 Kbits/sec
[  5]  10.01-10.39  sec  2.85 KBytes  61.3 Kbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth
[  5]   0.00-10.39  sec  0.00 Bytes  0.00 bits/sec                  sender
[  5]   0.00-10.39  sec  42.8 KBytes  33.7 Kbits/sec                  receiver
-----------------------------------------------------------
Server listening on 50000
```

ábra 19 iPerf mérés átviteli sebesség

Ennek oka az, hogy az összes hálózati csomagot (azokat is amelyeket nem akarunk kiíratni, vagy módosítani) a Scapy „sendp” függvényével kell elküldenünk. Ráadásul még meg is kell módosítanunk mindegyiket, hogy a helyes MAC címek kerüljenek a cél és a forrás MAC címek helyére, és emiatt a csomagok „cheksum” és „len” mezeit is újra kell számolni minden küldésnél. Összehasonlításképp a bridzsen elvégzett iperf3 mérések majdnem 10x-es növekedést mutatnak a sávszélességben.



ábra 20 iPerf mérés átviteli sebesség

Hasonló eredményeket értünk el abban az esetben is, amikor a csomagok módosítását és kiírását végeztük a hálózaton.

táblázat 1 iPerf mérési eredmények

Sorszám	Mérés leírása	Mérés eredménye (Kbits/s)
1	iperf mérés hálózati bridzs használatakor	33.7
2	iperf mérés minden csomag átjátszásával	265000

Figyelembe kell venni, hogy annak ellenére, hogy az iperf3 mérések nem mutatnak lényeges különbséget a sávszélességben a módosítások ideje alatt, a csomagok módosítása, felépítése, és „sendp” függvénnyel való elküldése behoz egy észrevehető késleltetést az érintett csomagok esetében. Annak érdekében, hogy ezt a késleltetést kimutassuk és mérni tudjuk, egy másik eszközre van szükségünk.

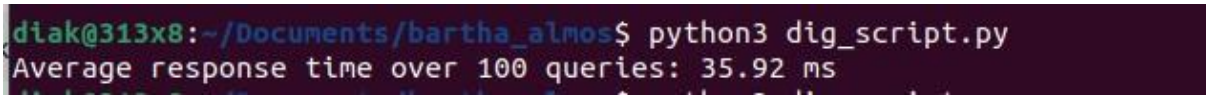
A „dig” parancs segítségével megnézhetjük a DNS szerver válasz idejét. Ezt felhasználva már mérhetjük azt, hogy mennyivel lassabban érkeznek a módosított csomagok.

Az alábbi ábrán láthatjuk, hogy a „Querry time” sorban megjelenik a szerver válasz ideje ms-ban.

Ez viszont csak egyetlen DNS kérésnek az idejét jelenti. Az eltérések ugyan azzal a konfigurációval ugyan azt a szervert kérdezve is jelentősek lehetnek. Esetekként akár +- 20 ms eltérés is előfordulhat ugyanazt a mérést végezve.

ábra 21 DNS server válaszidő mérése „dig” paranccsal

Annak érdekében, hogy reprezentatívabb mérési eredményeket kapjunk, szükségünk van egy olyan eszközre, amely nem csak egyetlen mérést végez, és a mérések végén átlagot számól. Ennek érdekében egy olyan Python scriptet implementáltam, amely egymást követően több „dig” parancsot futtat, kimenti a válasz időket és végül kiszámítja az átlag válasz időt.



ábra 22 Mérési eredmények Python

Annak érdekében, hogy a csomagok küldését egyik interfésztől a másikra felgyorsítsuk, különböző módszerekkel kísérleteztünk. A kódban minimális változtatásokkal (a „sendp” függvény kicserélésével) próbáltunk javítani a válaszidőket.

Először a „sendpfast” függvény segítségével próbáltuk csökkenteni a válaszidőt, viszont ennek az ellenkezőjét értük el (Lásd táblázat 2.).

Ezt következő ígéretesnek tűnő próbálkozás az volt, hogy az elküldendő csomagnak a 2. réteg béli socket-jét külön zárjuk (a Sacapy automatikusan zárja a socket-et minden küldés után), és felhasználjuk ugyanazt a socket-et minden csomag küldésénél. Ezzel a módszerrel sikerült felgyorsítani a csomagok küldését.

táblázat 2 dig script eredmények

Sorszám	Mérés leírása	Mérés eredménye (ms)
1	átlag DNS válaszidő módosítások nélkül	35.59
2	átlag DNS válaszidő sendp-vel küldve	133.2
3	átlag DNS válaszidő sendpfast-el küldve	295
4	átlag DNS válaszidő socket újrahasználatával	93

7. Jövőbeli tervek, továbbfejlesztés

Az alkalmazás grafikus felhasználói felületén lehetne javítani. Az elfogott csomagok mezeit részletesebben kiíratni.

Mást hálózati csomagok, például ARP csomagok módosítása és kiírása. Meglehetne valósítani egy ARP támadást úgy, hogy a belsőhálózatról küldött csomagokat átirányítjuk egy általunk választott szerverre.

Több mező módosítására lehetőség. A kiíratott mezők kiválasztására lehetőség. Például a felhasználó kiválasztja, hogy egy TLS csomagból melyik mezők jelenjenek meg.

Az NTP módosítást tovább lehetne fejleszteni úgy, hogy az „ntpdate” -et is sikerüljön átverni a hamisított csomagokkal.

A DNS csomagok módosítása esetében a DNS neveket és IPv4 címeket tartalmazó szótár módosításakor, a már meglévő elemek között kereshessünk név, vagy IPv4 cím szerint.

Ki bővíteni a DNS módosítást úgy, hogy nem csak IPv4 címekre, hanem IPv6 címekre is működjen.

A DNS csomagok esetében a lekérdezések válasz idejét még jobban lecsökkenteni.

Megpróbálni felgyorsítani a hálózati csomagok feldolgozását, úgy hogy C/C++ -ban Libpcap könyvtárat használunk.

8. Következtetések

A rendszer képes módosítani DNS és NTP csomagokat. A csomagok módosítása során az iperf-el készült mérések azt mutatják, hogy a hálózati forgalom nem lassul le, viszont a DNS csomagok esetében mégis lelassul a DNS csomagok forgalma a hálózaton. A TLS, http, és FTP csomagok esetében sikerült kiírni a kívánt mezőket.

Habár a Scapy egy nagyon sokoldalú és hasznos eszköz, hálózati csomagok módosítására, a hatékonyság szempontjából nem ideális. C/C++ nyelvben a Libpcap könyvtárat használva talán jobb eredményeket lehetne elérni.

Hálózati híd használata nélkül, a csomagok küldése két hálózati interfész között túlságosan is lelassítja a hálózatot.

9. Irodalom jegyzék

[1] T. Lammle, CCNA: Cisco Certified Associate Study Guide, Seventh Edition, Indianapolis, Indiana: Wiley Publishing, Inc., 2011.

[2] „Network Time Protocol,” Network Time Foundation, [Online]. Available: https://www.ntp.org/. [Hozzáférés dátuma: 20 5 2024].

[3] „HTTP,” mdn, [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP. [Hozzáférés dátuma: 15 5 2024].

[4] K. Yasar, „MAC address (media access control address),” TechTarget, 12 2022. [Online]. Available: https://www.techtarget.com/searchnetworking/definition/MAC-address. [Hozzáférés dátuma: 6 6 2024].

[5] E. Wesley, „Transmission Control Protocol (TCP),” Internet Engineering Task Force (IETF), 8 2022. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc9293. [Hozzáférés dátuma: 12 5 2024].

[6] „IPv4 and IPv6 address formats,” IBM, 2 3 2021. [Online]. Available: https://www.ibm.com/docs/en/ts3500-tape-library?topic=functionality-ipv4-ipv6-address-formats. [Hozzáférés dátuma: 2 6 2021].

[7] J. Postel, „ INTERNET CONTROL MESSAGE PROTOCOL,” 9 1981. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc792. [Hozzáférés dátuma: 5 6 2024].

[8] „SSL/TLS in Detail,” Microsoft, 10 8 2009. [Online]. Available: https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc785811(v=ws.10). [Hozzáférés dátuma: 10 6 2024].

[9] O. Weis, „FTP port – how to forward and open access,” 27 10 2023. [Online]. Available: https://mac.eltima.com/ftp-port.html. [Hozzáférés dátuma: 10 4 2024].

[10] P. Biondi, „Scapy Documentation,” 22 12 2022. [Online]. Available: https://scapy.readthedocs.io/_/downloads/en/latest/pdf/. [Hozzáférés dátuma: 11 4 2024].

[11] PyPI, 1 3 2023. [Online]. Available: https://pypi.org/project/NetfilterQueue/. [Hozzáférés dátuma: 15 4 2024].

[12] „ebtables(8) - Linux man page,” [Online]. Available: https://linux.die.net/man/8/ebtables. [Hozzáférés dátuma: 1 6 2024].

[13] „iPerf,” [Online]. Available: https://iperf.fr/. [Hozzáférés dátuma: 12 3 2024].

[14] A. Jony, M. N. Islam és I. H. Sarker, „Unveiling DNS Spoofing Vulnerabilities: An Ethical Examination Within Local Area Networks,” in *IEEE*, Cox's Bazar, Bangladesh, 2023.

[15] I. E. C. a. E. B. S. G. Aanchal Malhotra, „Cryptology ePrint Archive,” 2015. [Online]. Available: https://eprint.iacr.org/2015/1020. [Hozzáférés dátuma: 19 5 2024].

[16] „Raspberry Pi 4 Tech Specs,” [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/. [Hozzáférés dátuma: 28 8 2023].

[17] B. Lutkevich, „TechTarget,” 8 2021. [Online]. Available: https://www.techtarget.com/searchnetworking/definition/domain-name-system.

[18] D. J. W. ANDREW S. TANENBAUM, COMPUTER NETWORKS (5th Edition), PRENTICE HALL, 2010.

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

Vizat decan
Conf. dr. ing. Domokos József

Vizat director departament
Ș.l. dr. ing Szabó László Zsolt