3. wireshark TCP kapcsolat elemzés feladat

A következő gyakorlati példán keresztül megnézzük, hogy a valóságban, hogy néz ki egy egyszerű adatátvitel TCP segítségével.

Adatgyűjtés

A példához gyűjtsünk magunk számára adatokat. Első lépésként töltsük le az alábbi linkről az ott található szöveges állományt (ez az Alice csodaországban c. mű txt formátumban.)

http://gaia.cs.umass.edu/wireshark-labs/alice.txt

Ezután nyissuk meg az alábbi weboldalt a böngészőnkben:

http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html

Következő lépésként indítsuk el a Wiresharkot. Majd, miközben a háttérben fut a forgalommérés, a böngészőben a „*Fájl kiválasztása*" gombra kattintva válasszuk ki az előbb letöltött txt fájlt, majd kattintsunk az alul található „*Upload alice.txt*" gombra. Ha megkaptuk a visszajelző oldalt, akkor leállíthatjuk a wireshark mérést és bezárhatjuk a fenti oldalt. Ezután már csak a Wiresharkban kell dolgoznunk.

Szűrés

Hacsak nem kapcsoltunk ki minden, hálózatot használó programot a gépünkön, akkor valószínűleg sok más egyéb csomag is bekerült a listába. Először szűrjük ki a számunkra lényeges csomagokat az alábbi szűrőkifejezéssel:

```
tcp
```

Így csak a TCP protokollt használó csomagok maradnak bent a listában. Ezután, hogy biztos a jó streamet vizsgáljuk, keressük meg a fogadó ip címét, és szűrjünk erre is. Mivel tudjuk, hogy mi a fogadó domain neve (http://gaia.cs.umass.edu), valamint a 3. héten láttuk, hogy lehet rákeresni hosztnévre, könnyedén megtalálhatjuk a szükséges IP címet. Egyszerre lehet két szűrőt is használni a programozásból ismert ÉS művelettel:

```
tcp && ip.addr == 128.119.245.12
```

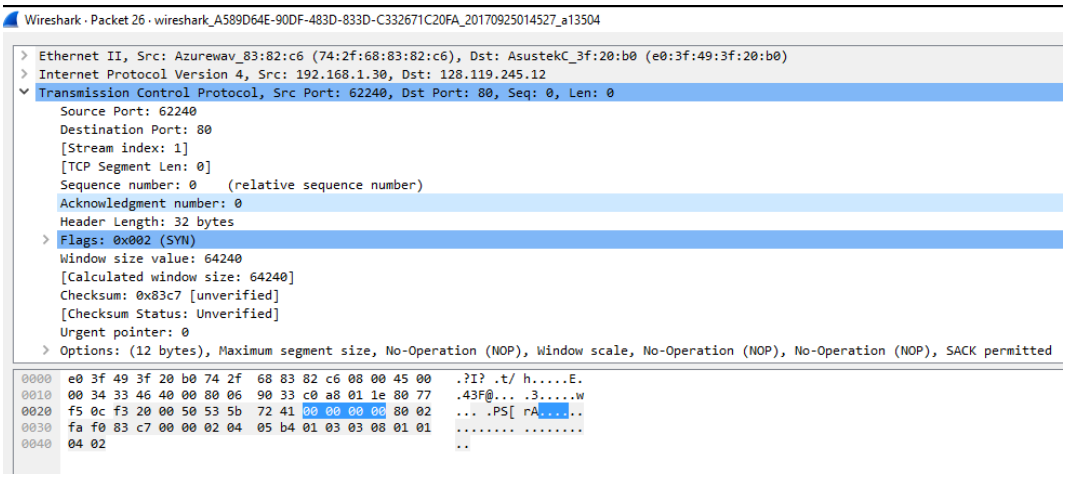
Ekkor csak olyan szegmenseket látunk a listában, amik TCP protokollt használtak és vagy a megadott IP címre mentek, vagy onnan jöttek. Kezdhetjük a TCP folyam vizsgálatát!

TCP kapcsolatot felépítő csomagok

Keressük meg az első három szegmenst a listában. A csomag info mezőjéből is láthatjuk, hogy ez a three-way handshake három szegmense.

26	10.180035	192.168.1.30	128.119.245.12	TCP	66 62240 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
27	10.309429	128.119.245.12	192.168.1.30	TCP	66 80 → 62240 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1452 SACK_PERM=1 WS=128
28	10.309559	192.168.1.30	128.119.245.12	TCP	54 62240 → 80 [ACK] Seq=1 Ack=1 Win=66560 Len=0

Ha megnyitjuk az első csomagot, megbizonyosodhatunk róla, hogy tényleg adat nélküli a szegmens, csak a fejrész található benne.



Vegyük szemügyre a szegmens főbb adatait!

Portszám megállapítása

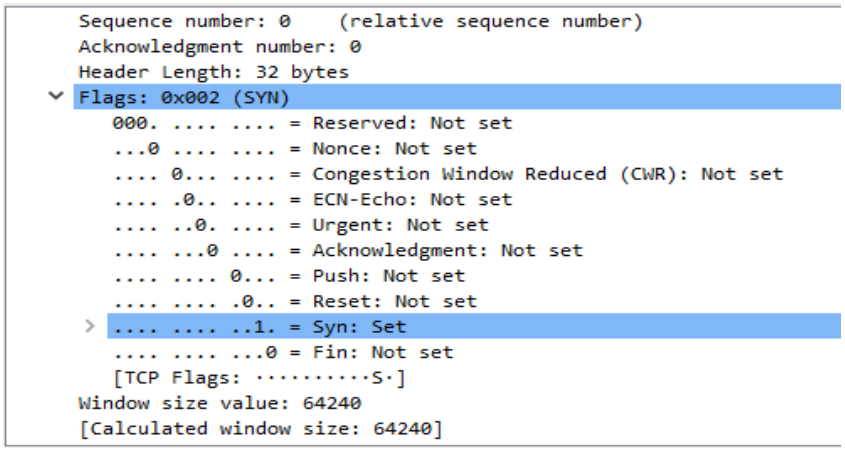
Mivel a mi gépünk kezdeményezte a kapcsolatot a Source port jelenti az én gépemen futó folyamat socketjének az azonosítóját, a Destination port pedig a távoli folyamat portja lesz (a 80 a már korábban látott http protokollnak a portja).

Header főbb részeinek kiolvasása

Látható még a képen a Sequence number mező, ami jelzi a küldő (tehát a gépünk) kezdeti sequence numberét, az Acknowledgement number pedig szintén nulla. Azonban, ha jobban megnézzük, itt az „ACK" flag 0, tehát ezzel jelzi a szegmens, hogy akármi is van az „Acknowledgement number" mezőben, az nem valós. A header length mező mutatja, hogy a fejrész jelen esetben 32 bájt. A számunkra érdekes mező jelenleg össze van csukva, ezért nyissuk le a „Flags" részt.

Kapcsolatot felépítő Flagek

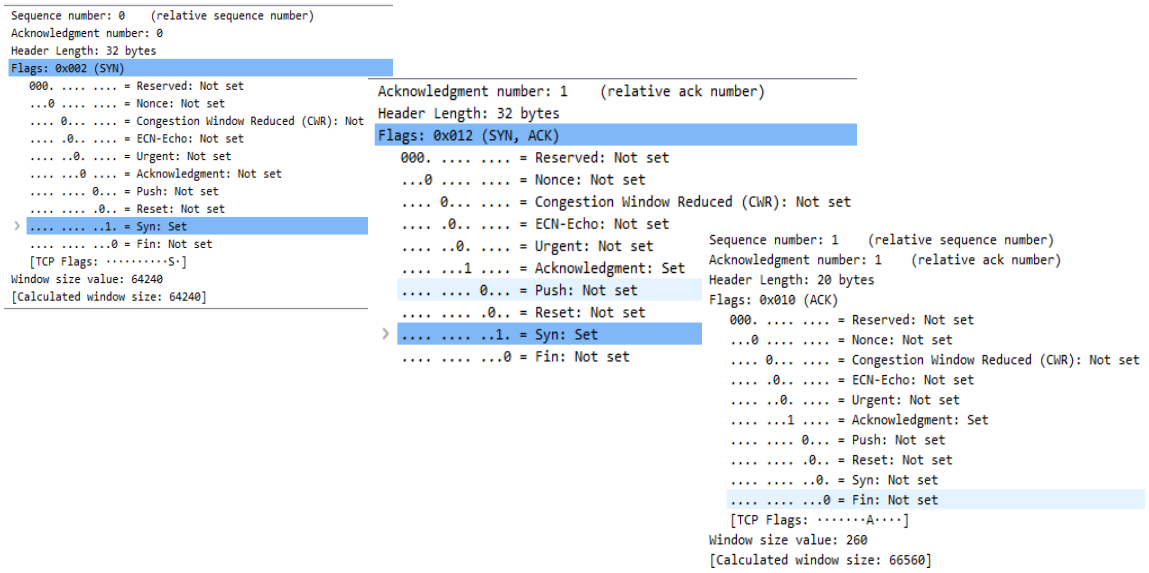
Itt láthatjuk a számunkra érdekes három Flaget: ACK (értéke 0), SYN (értéke 1) és FIN (értéke 0).



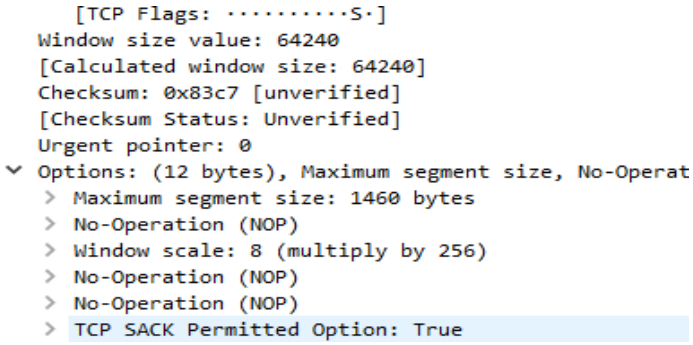
Three-way handshake csomagok

Az egymás alatti első három csomagot megvizsgálva láthatjuk a korábban említett három lépéses kapcsolatfelvételt. Amik megfigyelhetők a szegmenseken:

- A source és destination portszám változik, attól függően, hogy kliens vagy a szerver szegmenséről van szó
- A sequence number és az acknowledgement number eggyel nő
- A beállított flagek változnak: előbb SYN, aztán SYN+ACK, végül ACK



A SYN és SYNACK csomagokból még egy fontos dolgot megtudhatunk: ekkor egyeznek meg a felek (kliens és a szerver) arról, hogy mekkora a maximális szegmens méret, amit küldeni szabad. Ezt az options mezőben találjuk, azon belül pedig a *Maximum segment size* mező melleti érték lesz az. Jelen esetben az én gépem 1460-at állít be. (Fontos: a szerver és a kliens MSS értéke eltérhet!)



Adatcsomag vizsgálata

Miután felépült a kapcsolat a két fél között, megkezdődhet a továbbítás. Korábban láttuk, hogy először a kliens elküld egyszerre több szegmenst, amire aztán megérkeznek a nyugtázó válaszok. Egy adatszegmenst kiválasztva az alábbi adatokat tudhatjuk meg belőle.

```
Transmission Control Protocol, Src Port: 62240, Dst Port: 80, Seq: 1, Ack: 1, Len: 1452
    Source Port: 62240
    Destination Port: 80
    [Stream index: 1]
    [TCP Segment Len: 1452]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 1453    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    Header Length: 20 bytes
  Flags: 0x010 (ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
      .... .... ..0. = Syn: Not set
      .... .... ...0 = Fin: Not set
      [TCP Flags: ·····A····]
    Window size value: 260
    [Calculated window size: 66560]
    [Window size scaling factor: 256]
    Checksum: 0xb5c7 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  [SEQ/ACK analysis]
    [Reassembled PDU in frame: 201]
    TCP segment data (1452 bytes)
```

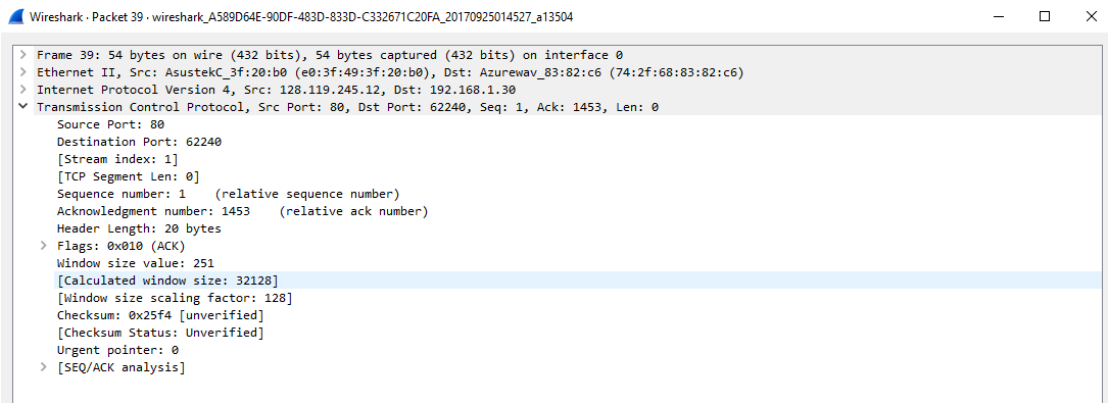
A portszámokat már ismerjük, ugyan azok, mint a kapcsolat felépítésekor. A Sequence number jelen esetben 1, ezzel jelöljük, hogy a küldeni kívánt adat első bájtjától indulunk. (Korábban szóba került, hogy a TCP egy véletlen számot generál kezdő szegmensszámként, hogy kisebb legyen az ütközés a későbbiek során. A Wireshark az egyszerűség miatt mindig átalakítja nekünk relatív számmá, azaz az első szegmens 0, és a többit ehhez viszonyítja.)

Az Acknowledgement number 1, ami azt jelenti, hogy fogadtuk a szerver 0 Sequence numberrel rendelkező szegmensét (ez volt a SYNACK szegmens) és jöhet az 1-es jelzésű. Mivel az ACK flag 1-re van állítva, így ez egy valós nyugta.

Végül a lista alján látható a TCP segment data, ami az adatot tartalmazza, összesen 1452 bájtot (ennyit határozott meg a szerver, mint MSS). Ha rákattintunk erre a sorra, alul kijelölődik a szegmensből az adatrész, látható hexadecimális és szöveges alakban is.

Szegmens nyugta

A kommunikáció két irányú is lehetne, de jelen esetben csak mi küldünk a szervernek adatot és ő nyugtáz nekünk, így a szerver felől érkező nyugtát fogjukmegnézni.



Ez az első adatszegmensre (amit az előbb néztünk) érkező nyugta. Ezt onnan látjuk, hogy az Acknowledgement number 1453 értéket veszi fel. Honnan jön ez? Az adatszegmens, amit az előbb néztünk, az 1. bájttól kezdődött és 1452 bájtot küldött, azaz a teljes küldendő adatfolyam 1-1452 bájtját. A TCP mindig azt nyugtázza, hogy mi az a bájtsorszám, amitől kezdve jöhet a folytatás – ez a következő, azaz 1453. Mivel ez egy nyugtázó szegmens, így ebben nincs adatmező a végén.

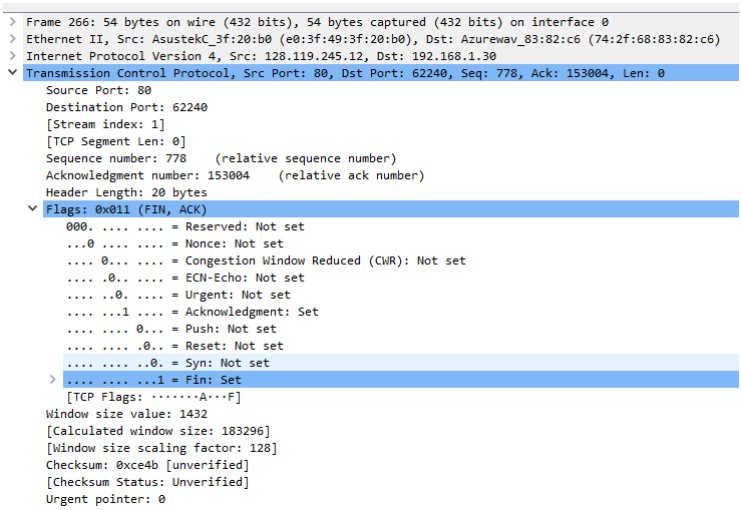
Újraküldés

Ha valami probléma van átvitel közben, akkor retransmission fog történni (azaz újraküld a küldő egy csomagot). Az ilyen eseteket az alábbi szűrőkkel választhatjuk ki:

```
tcp.analysis.retransmission || tcp.analysis.fast_retransmission
```

Kapcsolat lezárása

Ha végeztünk a feltöltéssel, a szerver elküldi nekünk a korábban látott visszajelző oldalt (amelyben leírja, hogy sikeres a feltöltés). Amint a szerver megkapja a nyugtát tőlünk erre a weboldalra vonatkozóan, bontja a kapcsolatot. Ezt az elméleti részben említett FINACK csomaggal teszi meg, ami így néz ki:



Ennek eddig csak egy mezője lehet ismeretlen: a Flag mező FIN nevű jelzője, amivel jelzi a szerverünk a kapcsolat végét. Erre a kliens már csak egy ACK üzenetet küld, nyugtázva a kapcsolat bontását. Mivel most a kommunikáció csak egyirányú volt, így nincs szükség arra, hogy mi is visszaküldjünk egy FIN szegmenst.

TCP stream összeállítása

Ha kíváncsiak vagyunk a teljes átküldött adatra, akkor kattintsunk jobb gombbal az egyik TCP szegmensre, és ott válasszuk ki a *Follow -> TCP Stream* menüpontot. Ekkor a felugró ablakban megjelenik összeállítva a teljes TCP folyam.

UDP csomagok vizsgálata

Mivel az UDP csomag sokkal egyszerűbb szerkezetű, illetve nincs szükség ilyen összetett folyamatra a küldéshez, az UDP csomagok vizsgálata legyen önálló munka. Nagy valószínűséggel mérés közben sikerült elkapni pár UDP szegmenst is. (Ha esetleg mégsem, akkor emlékezzünk vissza, hogy a múlt heti DNS protokoll is UDP-t használja a szállítási rétegben.) Az UDP szegmenseket az udp szűrővel tudjuk kiválogatni. Válasszunk ki egy tetszőleges szegmenst és nézzük meg, milyen részekből áll!

Feladat kérdések

1. Mi a kliens és a szerver portszáma? Mi a szerver IP címe?
2. Hogy tudod eldönteni, hogy egy szegmens a three-way handshakeben SYN, ACK vagy SYNACK szerepet tölt be?
3. Találtál-e valahol retransmissiont? Ha igen, melyik sequence numberrel rendelkező szegmenst kellett újraküldeni?
4. Válassz ki egy tetszőleges adatszegmenst, majd keresd meg az erre érkező nyugtát. Honnan tudtad, hogy melyik tartozik hozzá?
5. Válassz ki egy tetszőleges UDP csomagot. Mekkora a csomag mérete?
6. A hármas kézfogásról kérek egy képernyő fotót amit a pdf-be illesz be!