Wireshark használata, HTTP protokoll

Bordé Sándor

Tartalomjegyzék

Wireshark

A Wireshark segítségével elkaphatjuk és elemezhetjük a hálózaton közlekedő csomagokat.

Kiknek és miben segíthet a program?

- **rendszergazdák**nak a hálózati problémák felderítésében
- **hálózat-biztonsági szakemberek**nek a biztonsági rések felderítése
- **fejlesztők**nek a protokoll implementációk tesztelésénél, debuggolásnál
- **hallgatók**nak a hálózatok működésének megértésében (pl. ez a kurzus)

A program néhány főbb jellemzője:

- ingyenesen elérhető (wireshark.org), ugyan itt tutorial is található
- Linuxos és Windowsos verzió is van belőle
- elkapja egy hálózati interfészre érkező adatcsomagokat, ezekről részletes információt szolgáltat
- korlátozható az elfogni kívánt, illetve elfogás után a megjelenített csomagok köre
- a csomagok kereshetők több módon
- a forgalmi adatok elmenthetők és betölthetők

És végül: mire nem jó a Wireshark?

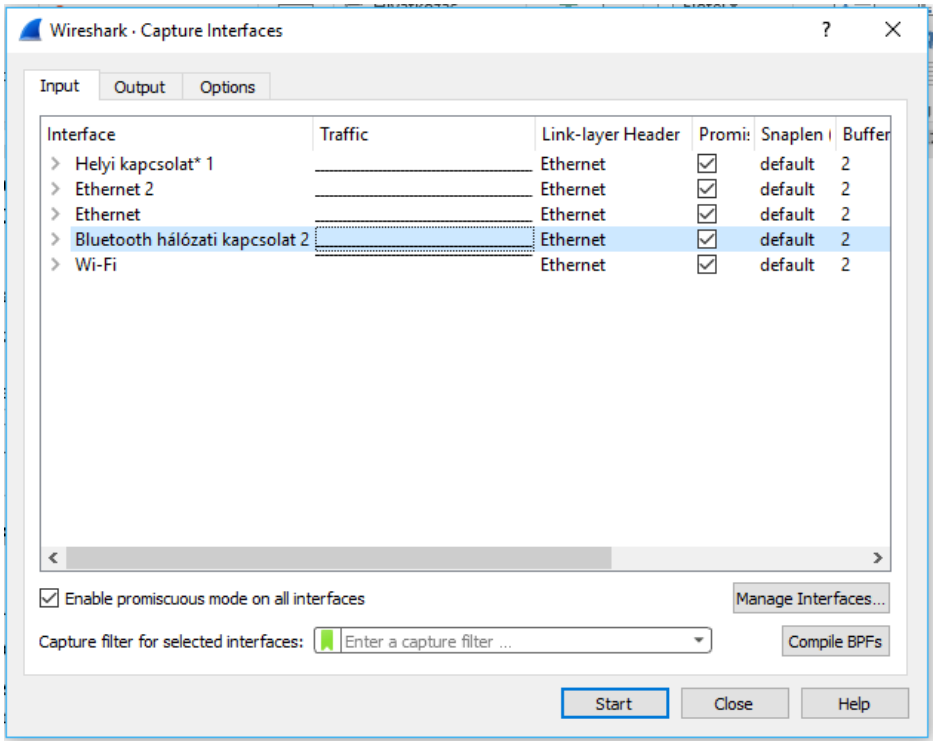
- Nem akadályozza meg, illetve nem figyelmeztet külső behatolás esetén.
    - Ennek ellenére, felhasználhatók a "különös" dolgok felderítésére.
- Nem lehet vele manipulálni a hálózatot, hanem csak mérni, megfigyelni lehet azt.

A program használata

A program elindításakor láthatjuk az elérhető hálózati interfészeket, amiknek a forgalmát figyelhetjük. Hogy melyik interfészre van szükségünk, az IP címe alapján tudjuk eldönteni (az 1. ábrán a negyedik kell nekünk).

1. ábra Interfész választó felület

Ha megfelelnek az alapértelmezett beállítások, akkor kétszer az interfészre (vagy egyszer a menüsorban a kék háromszögre – cápauszony) kattintva azonnal indíthatjuk a mérést. Az „Capture options" gombra kattintva (menüsoron a kis fogaskerék) beállíthatjuk a mérés paramétereit. Itt most csak egy-két érdekesebb beállítást fogunk megnézni, de a teljes leírás megtalálható a hivatalos tutorialban.

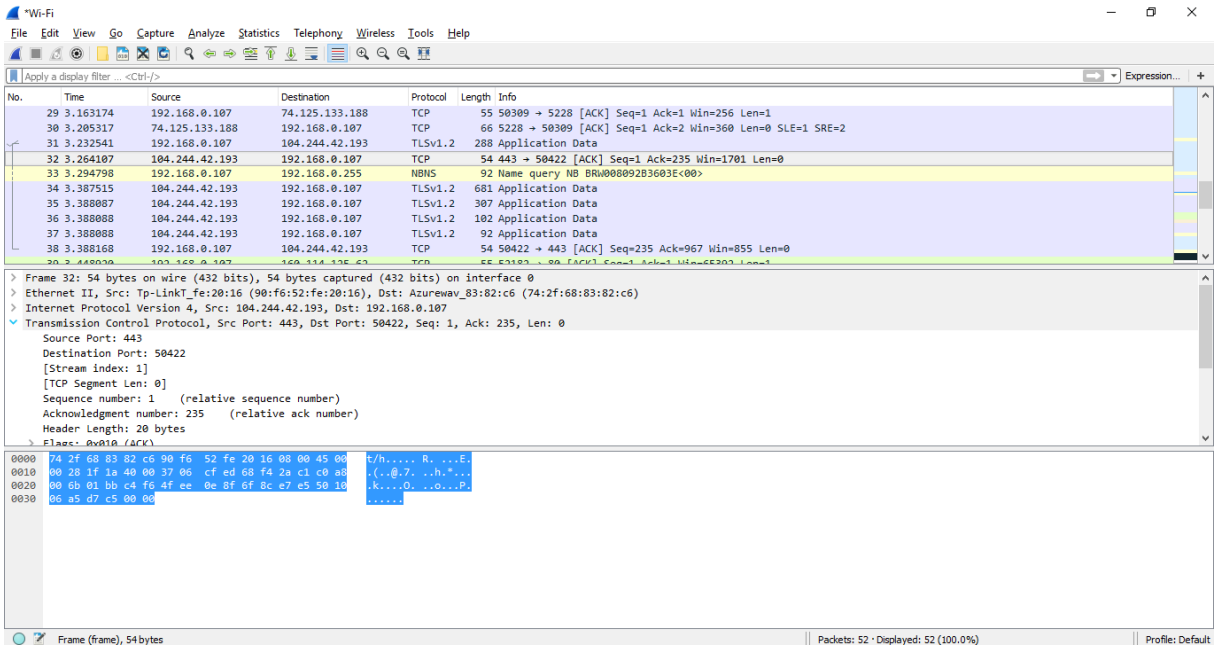


Az első ilyen beállítási lehetőség a „*Capture packets in promiscuous mode*” jelölőnégyzet. Alaphelyzetben a program csak a saját számítógépünknek címzett csomagokat fogja el. Ha bekapcsoljuk ezt a módot (tehát kipipáljuk a jelölőnégyzetet), akkor minden, a hálózati adapteren átfolyó csomagot elkapunk, nem csak azt, ami nekünk jön.

A „*Capture filter for selected interfaces*” felirat melletti sorba adhatunk meg elfogási szűrőt.

Munka az elfogott csomagokkal

Beállítás után a „Start” gombra kattintva elindul a forgalom figyelése. A listában valós időben jelennek meg az elkapott csomagok.



2. ábra Csomagok listája

A listában látható a csomagok legfőbb adatai: az elkapás ideje, sorszáma (ezzel tudunk rájuk hivatkozni), feladó és fogadó IP címe, a protokoll típusa és egyéb információ. Ha rákattintunk egy csomagra, alul megjelennek a részletes információi (dupla kattintás után átkerül új ablakba).

Bizonyos helyekre (fejléc, csomag a listában, részletes nézet) jobb egérgombbal kattintva helyi felugró menüt hozhatunk elő. A helyi menükben található menüpontok részletes leírásait a http://www.wireshark.org/docs/wsug_html_chunked/ChWorkDisplayPopUpSection.html oldalon olvashatjátok. Ezek közül néhányat emelek ki (de a többi is hasznos).

Fejlécre kattintva:

- **Sort Ascending/Sort Descending**: rendezi a csomagokat az adott mező szerint növekvő/csökkenő sorrendbe

A listában egy csomagra kattintva:

- **Apply as Filter:** a kiválasztott csomag alapján szűrőt hoz létre és azt alkalmazza a listára
- **Follow TCP Stream:** megjeleníti egy csomópont pár közötti TCP forgalmat

A részletes nézeten kattintva:

- **Wiki Protocol Page:** megnyitja a böngészőben az adott protokoll leírását
- **Filter Field Reference:** az adott protokoll szűrőjének referenciáját nyitja meg a böngészőben

Megjelenítési szűrők (Display filters)

Az elfogott és kilistázott csomagokat tovább szűrhetjük. A szűrőfeltételnek nem megfelelő csomagok nem tűnnek el a listából, csak nem lesznek láthatók. Szűrhetünk egy adott mező meglétére, mező értékére, protokollra...

Néhány példa szűrőkre:

- egy adott IP címre/ről jövő csomagok

  ```
  ip.addr==192.168.0.1
  ```

- A 25-ös (SMTP) port csomagjait jelenítsük csak meg

  ```
  tcp.port eq 25
  ```

- Csak a 10.0.0.5 címről érkező csomagokat mutassuk meg

  ```
  ip.src==10.0.0.5
  ```

További példák: http://wiki.wireshark.org/DisplayFilters

Szűrőprimitívek: http://www.wireshark.org/docs/dfref/

Szűrőkifejezések létrehozása, tárolása

Ha még nem vagyunk gyakorlottak a szűrőkifejezések létrehozásában, vagy egy adott protokollra vonatkozó primitívekben, akkor segítségünkre lehet a „*Filter Expression*” dialógusablak. A csomaglistánk felett lévő, „*Expression...*” gombra kattintva kapunk

egy listát, ahol protokollok szerint rendezve megtaláljuk az összes primitívet és relációt. Ezek segítségével könnyen összeállíthatjuk a saját szűrőkifejezésünket. Ha nevet is adunk neki, akkor később újra felhasználhatjuk.

Csomagok keresése

Lehetőségünk van egy adott csomag megkeresésére. Erre az „*Edit*” menü „*Find packet...*” menüpont (vagy a kis „üres” nagyítóikon az eszköztáron) szolgál. Kereshetünk szűrő alapján, byte szekvenciára vagy szövegrészre.

Csomagok megjelölése, ignorálása

A csomagok listájában megjelölhetünk, ignorálhatunk egyes csomagokat. Ezt úgy tehetjük meg, hogy a kívánt csomagra jobb gombbal kattintunk, és ott a „*Mark packet*” (jelölés) vagy „*Ignore packet*” (ignorálás) menüt választjuk.

Megjelöléskor fekete háttérszínt kap a csomag, így később könnyebb lesz megtalálni.

Ignoráláskor fehér hátterre és szürke betűszínre vált a csomag. Az ignorált csomagok nem kerülnek mentésre, tehát a program bezárása után ez elveszik.

Adatok mentése, betöltése

Lehetőségünk van korábban elfogott adatok betöltésére, illetve az aktuális forgalom elmentésére (ekkor az ignorált csomagok nem mentődnek). Össze is fűzhetünk több fájlt (például, mikor különböző interfészről gyűjtünk adatokat), ezt a „*File*” menü „*Merge*” menüpontjával tehetjük meg.

Egyszerűbb mód, ha a kívánt fájlokat egyszerre ráhúzzuk a munkaterületre.

Mivel a program segítségével a hálózaton közlekedő csomagokat lehet elfogni és megvizsgálni, érdemes megismerkedni az ehhez kapcsolódó fogalmakkal. Az 5-rétegű Internet protokoll stack-ről (TCP/IP) és az OSI modellről már volt szó, úgyhogy itt most csak ismétlésként szerepel. A protokollokról itt olvashatunk bővebben! Ahogy az korábban is meg lett fogalmazva, a réteges felépítést fentről-lefelé haladva vizsgáljuk.

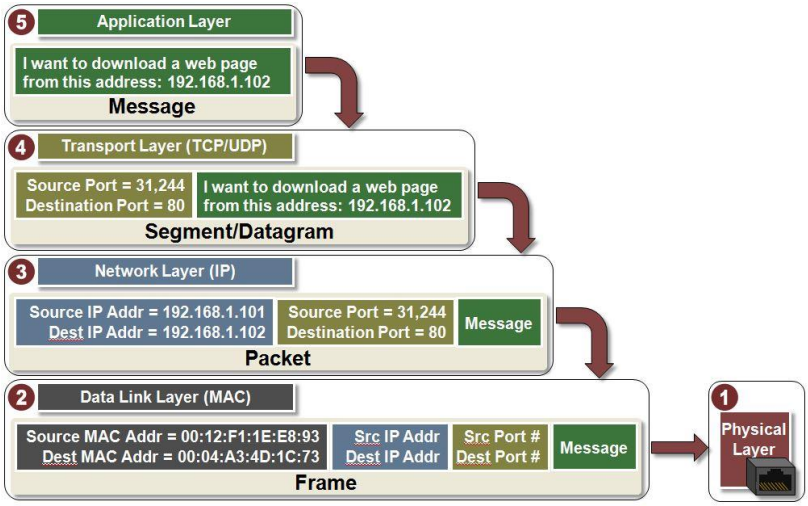
HTTP

5-rétegű Internet protokoll stack (TCP/IP)

A rétegződés a szolgáltatások és a protokollok szerint követhető végig. Minden réteg szolgáltatást nyújt a közvetlenül fölötte levő rétegnek. A fogadott szolgáltatások segítségével adott műveletek hajtódnak végre egy rétegen belül.

TCP/IP modell

A protokoll olyan szabályok halmaza, amelyek azt mondják meg, hogy milyen legyen a formátuma, és mi legyen a jelentése azoknak a csomagoknak és üzeneteknek, amelyeket egy adott rétegen belül a társak küldözgetnek egymásnak. Egy protokoll réteg implementálható szoftveresen, hardveresen vagy akár a kettő kombinációjaként is.



Példa a rétegződésre a TCP/IP modellben

Ezt a rétegződést ma is használják a gyakorlatban. A jobb oldali ábrán látható egy példa, hogy egy csomagban hogyan valósul meg a rétegződés.

Alkalmazási réteg - HTTP

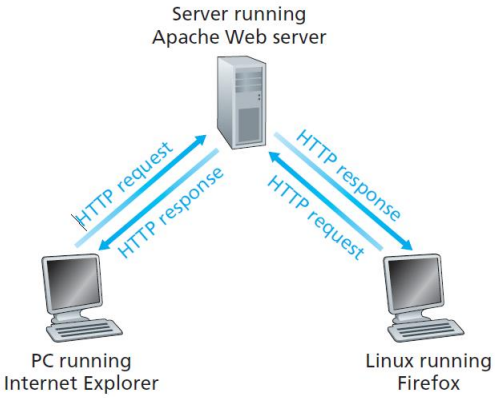
Az alkalmazási réteg protokolljai - mint például a HTTP (HyperText Transfer Protocol) - szinte kivétel nélkül szoftveresen kerülnek megvalósításra. Maradva a HTTP-nél, tekintsük át a működését!

A HTTP a Web alkalmazási rétegbeli protokollja. A HTTP két programban kerül implementálásra: egy kliens programban és egy szerver programban. A két program kü-

lönböző rendszereken fut, melyek HTTP üzenetváltások segítségével kommunikálnak egymással. A kliens oldalért felelős a Web böngésző, a szerver oldalért pedig a Web szerver.

HTTP kérés-válasz

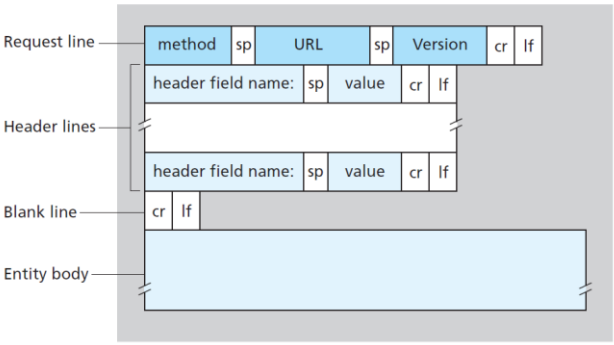
Amint azt fentebb tárgyaltuk a HTTP üzenetváltáshoz egy ún. kliens-szerver architektúra szükséges. A kliens a böngésző segítségével küld egy HTTP kérést (HTTP request) egy objektum iránt a szerver felé. A szerver fogadja a kérést és visszaküld egy HTTP választ, amely tartalmazza a kért objektumot. Az alsóbb rétegekről, és hogy azokban mi történik, a későbbi gyakorlatokon lesz szó.



HTTP kérés-válasz interakció

HTTP kérés

Az alábbi ábrákon látható, hogy egy HTTP kérés üzenet milyen mezőket tartalmaz, illetve azok milyen tartalmakkal rendelkeznek.



HTTP kérés minta

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/5.0
Accept-language: fr
```

HTTP kérés példa

Ahogy látható a 6. ábrán, az üzenet hagyományos - ember számára is olvasható - ASCII szövegként íródik. Minden HTTP kérésben szerepel egy kérés sor (request line), néhány fejléc sor (header lines), egy további üres sor. Ezek együtt alkotják a HTTP kérés üzenet fejlécét. Az üzenet hátralévő részében található az üzenet törzse (entity body), ha van. Lássuk részleteiben a mezőket!

Request line

- **`Method:`**
  - GET - ha valamilyen objektum iránt indítunk kérést (leggyakoribb)
  - POST - adatot továbbítunk a szerver felé, melyet az fel fog dolgozni (pl.: kitöltött űrlap)
  - PUT - adott erőforrás feltöltése
  - DELETE - adott erőforrás törlése
  - stb...
- **`URL:`** a kért objektum elérési útvonala
- **`Version:`** HTTP verziószám (1.0, 1.1, 2.0)

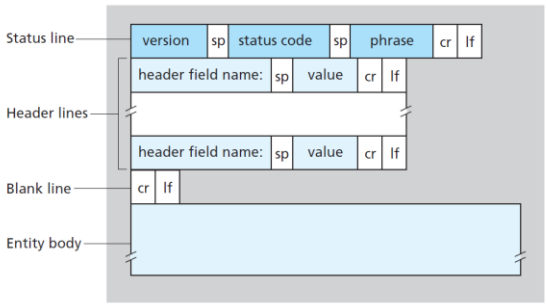
Header lines

- **`Host:`** a host, amelyen a kért objektum elérhető
- **`Connection:`** a válasz megérkezését követően mi történjen a megnyitott kapcsolattal
- **`User-agent:`** ami indította a kérést (pl.: böngésző)
- **`Accept-language:`** milyen típusú objektumot részesít előnyben a felhasználó

Entity body

HTTP GET kérés esetében ez a mező nem kerül kitöltésre, ugyanis csak egy objektum iránti kérést indítottunk, és a szerver minden szükséges információt megkap a fejlécből. Viszont egy HTTP POST kérés esetén adatokat szeretnénk továbbítani a szerver felé (pl.: egy űrlap kitöltését követően az űrlap mezőibe írt adatokat). Ekkor kap értelmet a törzs rész.

HTTP válasz



HTTP válasz minta

```
HTTP/1.1 200 OK
Connection: close
Date: Tue, 09 Aug 2011 15:44:04 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Tue, 09 Aug 2011 15:11:03 GMT
Content-Length: 6821
Content-Type: text/html

(data data data data data ...)
```

HTTP válasz példa

A kérésünkre érkező válasz is érdekes. Vizsgáljuk meg az itt található mezőket is! Továbbra is a felhasználó számára értelmezhető formában íródik az üzenet. A válasz üzenet egy állapot sort tartalmaz (status line) kérés sor helyett. Az alatta levő sorok nevei megegyeznek a kérés üzenetben szereplőkkel. Nézzük a mezőket részletesen!

Status line

- **`Version:`** 1.0, 1.1, 2.0 (ami a szerveren fut)
- **`Status code, phrase:`**
  - **`200 OK:`** Sikeres kérés, és az információk vissza lettek küldve a válaszban.
  - **`301 Moved Permanently:`** A kért objektum át lett mozgatva máshova. Az új URL egy ún. Location mezőben kerül specifikálásra.
  - **`400 Bad Request:`** Általános hibakód, mi szerint a szerver nem tudja értelmezni a kérést.
  - **`404 Not Found:`** A kért objektum nem létezik a szerveren.
  - **`505 HTTP Version Not Supported:`** A HTTP kérés verziószáma nem támogatott a szerver által.

Header lines

- **`Connection:`** a válasz elküldését követően mi történjen a megnyitott kapcsolattal
- **`Date:`** a válasz elküldésének időpontja
- **`Server:`** milyen Web-szerver generálta a választ
- **`Last-Modified:`** a fájl utolsó módosításának időpontja
- **`Content-Length:`** hány bájtos a válasz üzenet
- **`Content-Type:`** a válasz törzse (Entity body) milyen típusú (pl.: szöveg)

Entity body

A kért objektum tényleges tartalma.

Sütik

Bizonyára mindenki találkozott már olyan érdekes jelenséggel, hogy egy tetszőleges web-oldal (nem első alkalommal történő) látogatása közben az oldalon korábban meglátogatott részei vagy az ahhoz kapcsolódó részei közvetlenül elénk vannak rakva.

Például, egy webshop-ban szétnéztem az akciós laptopok között, és némelyiket részletesebben is megvizsgáltam (azaz rákattintottam a termékre). Másnap pedig visszalépve az oldalra egyből néhány laptop fogad. Ez hogyan lehetséges? Hogyan kerülnek ezek a termékek az oldalra bármilyen keresés és szűrés nélkül?

Ezekre a kérdésekre adnak választ a sütik (cookies). A web-oldalaknak a legtöbb esetben érdekük, hogy nyomon kövessék a felhasználót, milyen oldalakat látogatott. Ezekből az információkból viselkedési szokásokat lehet felállítani, és ezek alapján a felhasználó passzív igényeit tudjuk kielégíteni. Lássuk, ezek hogyan működnek!

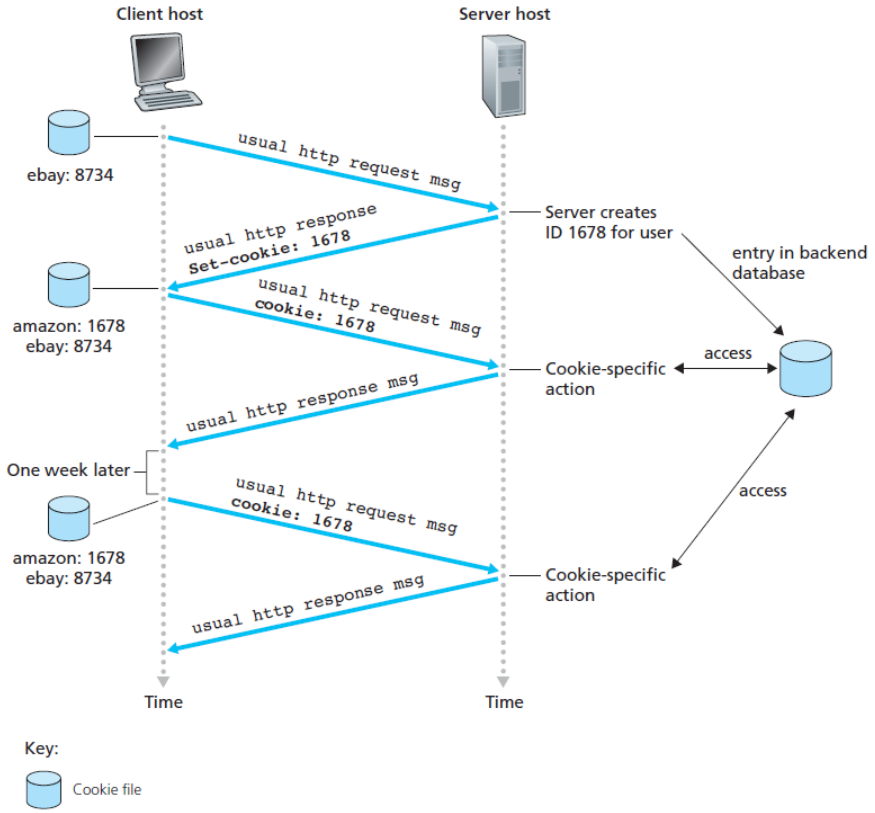
Sütik működése

A süti technológia négy komponensből áll:

1. egy süti-fejléc sor a HTTP válasz üzenetben
2. egy süti-fejléc sor a HTTP kérés üzenetben
3. egy süti-fájl a felhasználó rendszerében, amelyet a böngésző menedzsel
4. a back-end adatbázis a Web-oldalnál

A folyamat egy hagyományos HTTP kéréssel indul. A kérést fogadja a web-szerver, majd készít egy egyedi azonosítót, és egy, az azonosítóval ellátott belépési pontot back-end adatbázisában. A web-szerver reagál a kérésre egy válasszal, azonban ez kiegészül a korábbiakban látott válaszhoz képest egy **`Set-cookie:`** fejléccel. A böngésző fogadja a választ, és látja a **`Set-cookie:`** fejlécet. Ekkor a rendszerünkben

található süti-fájlt kiegészíti a böngésző egy új sorral, mely tartalmazza a szerver címét, és a fejlécben található azonosítót. Ezt követően, ha ugyanezen szerver felé indítunk kérést, a HTTP kérés üzenetben megjelenik egy **`Cookie:`** fejléc az azonosítóval ellátva. Ez a folyamat idézi elő, hogy a szerver kövesse a felhasználó aktivitását egy web-oldalon.



Példa a sütik működésére

Tekintsük meg az ábrán látható példát!

1. A kliens süti-fájlja már tartalmaz egy bejegyzést, miszerint már korábban meglátogatta az ebay oldalát. Első látogatását megkezdi az Amazon web-oldalára. A web-szerver felé indított első HTTP kérés egy hagyományos HTTP kérés üzenet.
2. A szerver fogadja a kérést, de mivel nem volt **`Cookie:`** fejléc a kérésben, így készít egy azonosítót a felhasználónak (1678), és ezzel az azonosítóval létrehoz egy belépési pontot is az adatbázison. A létrehozott azonosítót a **`Set-cookie: 1678`** mezőben visszaküldi egy HTTP válasz formájában.
3. A kliens böngészője fogadja a választ, megtalálja a **`Set-cookie:`** fejlécet, és a lokális süti-fájlba be is írja ezt az új sort.
4. A következő klienstől indított kérésben már szerepel a **`Cookie:`** fejléc a korábban fogadott azonosítóval együtt. Innentől a web-szerver képes süti-specifikus műveleteket végrehajtani (pl.: mit látogatott meg a felhasználó), és rendelkezésre bocsátja az adatbázist.
5. A válasz üzenet már a hagyományos HTTP response formát követi, ugyanis a süti azonosító már korábban el lett küldve a felhasználónak (nincs **`Set-cookie:`** mező).
6. A süti egy hét múlva ugyanúgy működik, a lokálisan eltárolt süti-fájlból betöltött azonosító elküldésével.