1 **Adatbiztonság, adatvédelem**

Kommunikációs protokollok, a Kerberos és az SSL, valamint a Heartbleed hiba.

2 **Kerberos**

- Jegy alapú protokoll, ami nem biztonságos hálózaton tesz lehetővé biztonságos beléptetést.
- Főbb jellemzői:
  - Szerver-kliens felépítésű
  - Kölcsönös azonosítás
  - Szimmetrikus kulcsos titkosítást alkalmaz
  - Opcionálisan két kulcsot az azonosítás bizonyos szakaszaiban
-

3 **Kerberos története**

- Az MIT-n dolgozták ki a Project Athena számára
- Ez egy egyetemi elosztott számítási rendszer
- Nem mai szoftver, a fejlesztése valamikor 1983 környékén kezdődött meg.
- Nem a 0-ról alkották meg egy korábbi rendszer a Needham–Schroeder adja a rendszer alapját.

4 **Kerberos története**

- Első publikus kiadás a 4-es verzió volt, valamikor az 1980-as évek vége felé
- Az 5-ös változata 1993-ban jelent meg, ez már RFC számot is kapott: RFC 1510.
- Ez 2005 óta elavultnak számít, mivel az RFC 4120 megjelent.
- Ez az új RFC a régi 4-es kompatibilitást távolította el a rendszerből. Biztonsági okok miatt.

5 **Kerberos története**

- MIT Licenc alatt lett kiadva, vagyis bárki használhatja szabadon a kész implementációjukat.
- 2007-ben alakult meg a Kerberos Consortium, amit az MIT azért hozott létre, hogy a további fejlesztést ne nekik kelljen elsősorban tovább vinni.

6 **Kerberos története**

- Fontosabb fejlesztők: Microsoft, Apple, Google, MIT, Stanfordi egyetem, stb…
- Sajnos az amerikai törvények miatt a 2000-es évek előtt csak Amerikában volt használatos
- Ennek oka az, hogy a rendszer jobb megoldás hiányában DES algoritmust használt

7 **Kerberos története**

- Így a törvények szerint a rendszer exportja fegyverkereskedelemnek minősült.
- Ezért kifejezetten megtiltották a rendszer exportálását Amerikán kívülre.
- A 2000-es évek elején változtattak a törvényi szabályozáson
- Onnantól kezdett el terjedni rohamosan.

8 **A kerberos név eredete**

- Görög mitológiára vezethető vissza
- Angolosan Cerberus (Mass Effect játékosoknak ismerős lehet) Hádész, az alvilág

istenének a Három fejű őrző kutyája.
- A kutya feladata az volt, hogy megakadályozza az alvilágba belépett lelkek megszökését.

9 **A rohamos terjedés**
- A Windows 2000-ben jelent meg először az Active Directory szolgáltatás azonosítási mechanizmusaként.
- Érdekesség, hogy bár teljesen nyílt az implementáció a Microsoft mégsem használja
- Saját implementációjuk van, ami RFC 3244 és RFC 4757 alatt van dokumentálva.

10 **A kerberos név eredete**
- A saját MS implementációnak komoly okai vannak. Nem pedig az az oka, hogy a Microsoft rossz cég:
  - 2000-ben már használták.
  - Ekkor még nem volt AES, de már a DES-se megbízható
  - Kerberos még mindig DES-re épült
  - Változtattak: Titkosítás RC4-el.
  -

11 **Egy kicsit az RC4 algoritmusról**
- Implementálási egyszerűsége egy faékkel vetekszik.
- Ezért sok helyen alkalmazzák pl:
  - WEP (nem a legjobb példa)
  - WPA
  - RDP
  - Bittorent
  - Kerberos

12 **Egy kicsit az RC4 algoritmusról**
- Ron Rivest tervezte (RSA egyik megalkotója)
- RC4 név márkanév ezért sok helyen ArcFour algoritmusnak nevezik
- Véletlenszerűen generált kulcsokra épül.
- Azonban ha a kulcsok között összefüggés van, vagy esetlegesen nem eléggé véletlenszerűek, akkor bukik a biztonság.
- Lásd: WEP...

13 **A kerberos működése**
- Három szereplős:
  - Kliens
  - Azonosítási szerver (AS)
  - Szolgáltatás szerver (SS)

14 **A kerberos működése**

15 **A kerberos működése**
- Felhasználó begépeli nevét és a jelszavát a kliensen
- Ebből a kliens képez egy hash-t, ami a felhasználó titkos kulcsa
- Ezután a kliens azonosítási kérelmet küld a AS-nek, majd elküldi a titkos kulcsot.

16 **A kerberos működése**

- A titkos kulcs alapján az AS szerver ellenőrzi, hogy az adatbázisában van e a kliens.
- Ha igen, akkor a felhasználó tárolt jelszavából képez egy hash-t

17 **A kerberos működése**

- Az AS két választ küld a kliensnek (A, B):
  - Egy munkamenet azonosítót titkosítva az általa képzett hash-el.
  - Valamint egy azonosítási jegyet, amiben benne van a kliens címe, a jegy érvényességi ideje, a felhasználó azonosítója szintén titkosítva

18 **A kerberos működése**

- A kapott válaszokat a kliens dekódolja a felhasználó jelszavából képzett hash-el.
- Amennyiben a jelszó nem volt megfelelő nem tudja dekódolni a kapott üzenetet, így nem tud a továbbiakban kommunikálni

19 **A kerberos működése**

- Szolgáltatás igénybevételekor a kliens a kapott jegyből és az igényelt szolgáltatás ID-ből képez egy üzenetet. (C)
- Továbbá csatolja mellé a kliens azonosítóját és a az azonosítás idejeté titkosítva a jegy adatokkal. (D)

20 **A kerberos működése**

- Ezen információk segítségével az azonosító szerver képez egy szolgáltatás igénybe vevő jegyet (E)
- Valamint egy titkosító kulcsot (F), amit titkosítva küld el.

21 **A kerberos működése**

- Ezután a kapott információk segítségével tudja a kliens igénybe venni a tényleges szolgáltatást.
- Az előző körben kapott E üzenet és egy új G üzenet segítségével.

22 **A kerberos működése**

- A G üzenet azt tartalmazza titkosítva az azonosításra vonatkozó információkat. Vagyis garantálja azt, hogy a kliens valóban jogosult a szolgáltatás használatára.

23 **A kerberos működése**

- Miután az SS szerver megkapta az üzenetet, visszaküld egy H választ, ami az E üzenetben található időbéjjeg+1
- Ezt megkapva a kliens ellenőrzi, hogy jó választ kapott e.

24 **A kerberos működése**

- Ha igen, akkor bízhat a szerverben.
- A tényleges kommunikáció ezután indul csak meg.

25 **Hátrányok, limitációk**

- Folyamatos hálózati kapcsolat szükséges.
- Ha csak egy Szerver van, akkor senki sem tud bejelentkezni, ha az kidől.
- Windows esetén ez úgy van megoldva, hogy egy Active Directory tartományban a tartomány vezérlők szinkronizálnak egymás között, így automatikusan elosztott lesz a rendszer.

26 **Hátrányok, limitációk**

- Vagyis ha egy szerver kidől, akkor is be lehet jelentkezni.
- Sajnos csak nagy tartományok esetén van egynél több szerver.
- Pedig a célszerű az lenne, hogy telephelyenként legyen egy szerver legalább.

27 **Hátrányok, limitációk**

- Pontos idő kell a működéshez.
- Ennek az az oka, hogy minden üzenet időbéjjeggel van ellátva, valamint az azonosítás egy adott ideig érvényes csak.
- A kliens és a szerver közötti időeltolódás maximum 5 perc lehet.
- Gyakorlatban ezt NTP szerverhez való szinkronizálással lehet kivédeni

28 **Hátrányok, limitációk**

- Az azonosítási folyamat (vagyis, hogy miként képzünk hash-t a jelszavakból, hogy titkosítunk) nincs szabványosítva
- Így az egyes implementációk kompatibilitása kérdéses.
- Továbbá mivel központilag azonosítunk problémát okozhat ha a támadó hozzáfér az azonosító szerverhez.

29 **SSL**

30 **Az SSL**

- Szállítási réteg biztonság magyarra fordítva. Két része van:
- Transport Layer Security és Secure Sockets Layer
- A TLS az SSL-ből fejlődött ki.
- Az SSL-t eredetileg a Netscape fejlesztette ki.

31 **Az SSL**

- Az 1.0-ás változatot sosem publikálták belőle, az első kiadott változat a 2.0 volt, amit 1995-ben adtak ki.
- 1996-ban jelent meg a 3.0, mivel a 2.0-ban voltak hibák. Ez már RFC számot is kapott: RFC 6101
- A TLS az SSL 3.0 alapján készült el 1999-ben. RFC száma: 2246

32 **A TLS**

- A TLS megjelenése óta SSL ritkán alkalmazott, mivel felváltotta teljesen.
- Visszafelé kompatibilis az SSL 3.0-val.
- Jelenlegi legfrissebb változata a TLS 1.2
- Ez a biztonságot növeli, azonban nem minden támogatja. (Kb semmi ☺, Windows-ban és Inernet Exporerben benne van, de le van tiltva)

33 **A TLS**

- Így a legelterjedtebb változat még mindig az 1.0
- Az SSL működéséről vázlatosan volt szó a kétkulcsos rendszereknél.
- Ennél többet felesleges tudni róluk, mivel az eltérés a változatok között az algoritmusokban van.

34 **A TLS**

- A jelenleg használt 1.0-ban vannak hibák, vagyis nagyon sok erőfeszítéssel és számítási idő befektetésével elvileg belátható időn belül lehetséges feltörni.
- Eddig 1x tették meg demonstrációs célból 2011-ben

- Ez a TLS 1.1-ben már nem lehetséges

35 **A Bug, ami felfordította a világot**

36 **Heartbleed bug**

- Nagyon súlyos OpenSSL sebezhetőség
- 2011-ben került bele a program forráskódjába
- A múlt héten erről szólt minden az internetes biztonsági fórumokon

37 **Honnan jön, mi ez, miért fontos?**

- Az OpenSSL egy nyílt forráskódú SSL/TLS függvénykönyvtár, ami kb mindenütt használt az interneten: Szerverek, böngészők, telefonok, stb…
- Ebben találtak egy nagyon durva programozási hibát, amit kihasználva az SSL és TLS nem lesz biztonságos.

38 **Honnan jön, mi ez, miért fontos?**

- Az SSL és TLS kommunikációs csatorna felállítása a kétkulcsos mivoltból adódóan sok időt vesz igénybe.
- Ezért a HTTPS és egyéb protokollok esetén, amelyeknek alapvetően kiszolgálás után (HTTP Session-ről most nem beszélünk) nem lenne szükségük a csatorna fennmaradására életben tartják a titkosított csatornát egy ideig.

39 **Honnan jön, mi ez, miért fontos?**

- A fennmaradási időt a kliensnek kell jeleznie.
- Ez egy TCP csomag, amit heartbeat-nek neveznek. Lényegében ezzel a kliens azt jelzi a szervernek, hogy még itt vagyok, ne zárd be a csatornát.
- Ebbe a heartbeat mechanizmusba csúszott be egy csúnya hiba.

40 **A heartbeat működése és a hiba**

- A kliens egy kérést küld a szervernek, ami kb az alábbi formában fogalmazható meg szöveegesen: Szerver, ha itt vagy még, küldj négy bájt adatot vissza. A négy bájt tartalma: hello
- A bug egy „egyszerű” tömb túlindexelés, ami hatására megtörténhet az alábbi extrém szituáció is:

41 **A heartbeat működése és a hiba**

- Szerver, ha itt vagy, küldj 64kb adatot, ami: hello
- Ennek hatására a szerver a hello üzenetet fogja elküldeni, illetve 64kb-4byte adatot pluszba, ami éppen a memóriában van.
- Ez azért aggályos, mert egy egyszerű méret ellenőrzés maradt ki

42 **A következmény**

- A támadó után semmilyen nyom nem marad az OpenSSL felépítéséből adódóan.
- Kellő idővel a teljes szerver memória átmenthető a támadó gépére
- Ebből aztán elvileg és gyakorlatilag is lehetséges a privát kulcs megszerzése
-

43 **A hiba megszületése**

- Összeesküvés elméletek szerint az NSA keze is benne van a dologban, természetesen az NSA tagad, de a hiba felfedezésének a körülményei több, mint érdekesek.
- Ezért nem zárható ki teljesen a szerepvállalásuk.

44 **A hiba megszületése**

- Ami biztosan tudott, az az, hogy egy fejlesztő 2011. december 31.-én este kommitolta a hibás kódrészletet, amiért bocsánatot is kért.
- Arról nincs információ sajnos, hogy mennyire volt jó a buli, ami keretében ez a hiba megszületett ☺

45 **Köszönöm a figyelmet.**