

Pintér-Husztai Andrea

2020. szeptember 7.

# Tartalom

## 1 Azonosítás, Hitelesítés, Feljogosítás

- Alapfogalmak
- Jelszavas hitelesítés
- Birtoklás alapú hitelesítés
- Biometrikus hitelesítés
- Távoli hitelesítés

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡

( )

- [illegible]

# Hitelesítés módjai

- **Tudás alapú:** jelszó, PIN (personal identification number), előre meghatározott kérdésekre vonatkozó válaszok
- **Birtoklás alapú:** kulcs, elektronikus kártyák, smart kártya. Ezeket az eszközöket tokeneknek nevezzük.
- **Biometrikus**
  - **Statikus:** ujjlenyomat, írisz, retina, DNS
  - **Dinamikus:** aláírás, kézírás, beszédhang, gépelési ritmus, járási mód, szóhasználat, testbeszéd, arcmimika

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ↺ 🔍 ↻

---

- Az azonosító alapján dönt a rendszer arról, hogy egy felhasználó jogosult -e hozzáférésre.
- Az azonosító alapján rendel a rendszer privilégiumokat egy felhasználóhoz. (superuser, guest or anonymous accounts etc.)
- Az azonosítóknak nagy szerepe van a diszkrecionális hozzáférés-vezérlésben (DAC) is. Pl. más felhasználók azonosítóinak megadásával jogosultságot adhatunk olvasásra

## Jelszavak népszerűsége

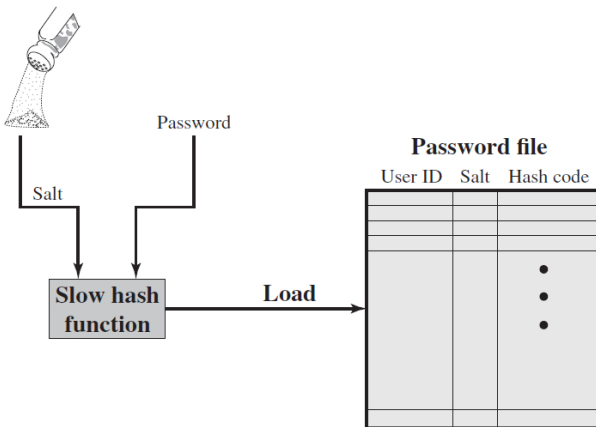
- Könnyen, bárki számára érthető az azonosítás folyamata.
- Nincs szükség semmilyen kliens oldali szoftver telepítésére, hogy a kliens oldali hardvert (pl. kártyaolvasó) használni tudjuk.
- A tokenek drágák és sokszor kényelmetlen hordozásuk, főleg, ha több token is használunk.

## Helyi hitelesítés

Unix állományok: A jelszó és egy só hash értékét adjuk meg.

- 1 A felhasználó választ vagy kap egy jelszót.
- 2 Ehhez a jelszóhoz tartozik egy fix hosszúságú só érték. A só idő-alapú (régebbi implementációk), álvéletlen vagy véletlen (új implementációk) generátorok által megadott véletlen szám.
- 3 Kiszámoljuk a jelszó és a só konkatenációjának lenyomatát.
- 4 A jelszófájlban tároljuk a sót (nyíltan) és a lenyomatot a megadott azonosítóval.

# UNIX password files - Új jelszavak



## Unix password file

Password file (nem tartalmazza a lenyomatot):

```
oracle:x:1021:1020:Oracle user:/data/network/oracle:/bin/bash
```

1	felhasználói név	2	jelszó( csak *)
3	felhasználó azonosító (UID)	4	csoport ID
5	felhasználói info.	6	home könyvtár (absz. út)
7	shell (absz. út)		

# Unix shadow file

Shadow file (lenyomatot tárolja, csak a superuser olvashatja ):

vivek:\$1\$fnfffc\$PgteyHdicpGOfffXX4ow#5:13064:0:99999:7:::

↓
↓
↓
↓
↓
↓

1
2
3
4
5
6

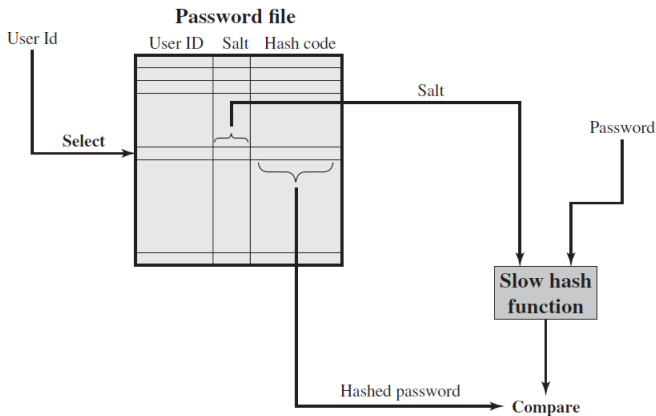
1	felhasználói név
2	hashed jelszó
3	utolsó jelszótárolás ideje
4	min. hány nap múlva változtatható
5	érvényességi idő
6	lejárat előtt hány nappal jelezzen

\$1\$ md5, \$2a\$ Blowfish, \$2y\$ Blowfish, with correct handling of 8 bit characters, \$5\$ sha256, \$6\$ sha512 + salt (fnfffc) + hash

# UNIX azonosítási folyamat

- A felhasználó megadja azonosítóját (ID) és jelszavát.
- Az operációs rendszer az ID segítségével kikeresi a nyílt sót és a lenyomatot.
- Kiszámítja a lenyomatot és összehasonlítja a tárolttal.
- A só jelentősége:
  - 1 Ugyanazon jelszó mellett is különböző lenyomatok tárolódnak.
  - 2 Nehezíti az offline szótártámadást. A támadónak minden egyes jelszó sejtését minden egyes az állományban szereplő sóval ki kell próbálnia (többszörözi a próbálgatások számát).
  - 3 Megnehezíti annak ellenőrzését, hogy egy felhasználó ugyanazt a jelszót használja -e a különböző rendszereknél.

# UNIX password/shadow files - A jelszó ellenőrzése



# A UNIX-szerű implementáció - OpenBSD

- Az OpenBSD operációs rendszer ismert arról, hogy a biztonságra fókuszál.
- OpenBSD a Blowfish szimmetrikus titkosításon alapuló hash függvényt (Bcrypt) használ. Elég lassú.
- Bcrypt legfeljebb 55 karakter hosszú jelszavakat és 128 bites sóhoz 192 bites hash értéket ad meg.
- Bcrypt egy költség változót is kezel, melynek növelésével a Bcrypt hash kiszámítása több időbe kerül.
- Új jelszó esetében a költség változó konfigurálható, magasabb privilégiumú felhasználóhoz nagyobb költség rendelhető.
- SUSE Linux is Bcrypt-et használ.

# Offline szótár támadás, szivárvány táblák

- **Offline szótár támadás** A támadók megszerzik a lenyomatokat tartalmazó állományt. A támadók a lehetséges jelszavakból egy nagy szótárt készítenek. A szótár elemeit (minden egyes sóval) hash-elik és hasonlítják az állományban levőkkel. Ha sikertelen, akkor a szótár elemein transzformációkat hajtanak végre, és úgy keresik az egyezést. Transzformációk: fordított soorend, tükrözés, speciális karakterek/sztringek beszúrása stb.
- **Szivárvány táblák** A támadó előre kiszámolja a lehetséges hash értékeket a szótárban levő elemekre, a lehetséges sókkal véve. Az eredmény egy hatalmas tábla.

## Offline szótár támadás, szivárvány táblák: Védelmi intézkedések

- Elég hosszú só és elég hosszú hash érték használata. (OpenBSD biztonságosan működik belátható ideig.)
- Jelszófájlok illetéktelen hozzáféréstől való védelme (shadow file).
- Behatolásérzékelő rendszerek jelzik az incidenst, új jelszó gyors újratelepítése.

# Szivárvány tábla támadás példa

- A szerző(1990 Klein, D.) UNIX jelszófájlokat gyűjtött össze, közel 14,000 hash értéket. A jelszók negyedét sikerült kitalálnia kevesebb, mint egy óra alatt.
- Algoritmus:
  - 1 A felhasználó neve, annak kezdőbetűi, felhasználói név más személyes információ felhasználása.
  - 2 Különböző szótárak szavainak kipróbálása.
  - 3 A kapott szavakra különböző transzformációk végrehajtása: permutációk, első betű vagy a teljes szó nagybetűssé konvertálása, vagy az "o" betű "zero"-val való helyettesítése, stb.

# Offline szótár támadás, szivárvány táblák: Modern megvalósítások

- **Számítási kapacitás** jelentősen megnőtt. (e.g. grafikus processzorok) 2012: A AMD Radeon HD7970 GPU-vel rendelkező PC átlagosan  $8.2 * 10^9$  jelszót tud kipróbálni másodpercenként. A 2000-es évek elején csak drága szuperszámítógépek voltak képesek erre.
- **Modellek** alkalmazása, melyek a természetes nyelv bennük előfordulási valószínűségét vizsgálják (pl. standard Markov modell, valószínűségi környezetfüggetlen nyelvtanok). Cél: A szótár/tábla méretének csökkentése. 2009-ben egy SQL befecskendezéses támadással (RockYou.com online játék) 32 millió nyílt jelszóra tettek szert. Azóta egyre több minta áll rendelkezésre analízis szempontjából.

# Általános védelmi intézkedések

- **Felhasználók oktatása:** A felhasználókban tudatosítjuk a jelszóválasztás fontosságát és módját (hossz, nagybetűk, számok, speciális karakterk, “My dog’s first name is Rex” -> MdfniR). Sajnos nem elegendő ez az intézkedés a sikerhez.
- **Számítógép-generált jelszavak:** Ha a jelszó elég random, a felhasználó nem fogja megjegyezni. FIPS 181 megoldást ad megfelelő számítógépes generálásra. Az algoritmus olyan szótagokat generál, melyek konkatenációja könnyen kiejthető, megjegyezhető.

# Általános védelmi intézkedések

- **Reaktív jelszó ellenőrzés:** Felhasználói jelszavak folyamatos ellenőrzése valamely jelszó feltörő programmal. A kitalált jelszavakat jelentik a felhasználóknak. Hátrány: erőforrás igényes. Egy password cracker:  
<http://www.openwall.com/john/doc/>.
- **Összetett eljárásrend:** A felhasználó maga választja jelszavát, bár a rendszer ellenőrzi annak megfelelőségét. Cél: egyensúly a felhasználói elfogadottság és a megfelelő jelszóerősség között.

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ 🔍 ↺

# Birtoklás alapú hitelesítés: Kártyák

- **Dombornyomott kártyák:** Vésett karakterek az elején.
- **Mágnescsíkos kártyák:** Mágnescsík hátul, karakterek az elején.
- **Memória kártyák:** A chipkártya legegyszerűbb formája, tárolni tud adatokat, de számításokat nem végez.
- **Smart kártyák:** A kártya egy mikrokontrollert és memóriát tartalmaz, valamint hatékony kriptográfiai és hitelesítési algoritmusok bevezetésére nyújt lehetőséget.



# Smart kártya

- **Kontaktusos:** Az olvasóba kell helyezni. A kártya és az olvasó fizikai kapcsolaton keresztül kommunikálnak.
- **Kontaktusmentes:** A kontaktusmentes kártyát elég néhány centiméternyire közelíteni a leolvasó berendezéshez és a tranzakció máris megvalósítható. Az olvasó és a kártya rádió frekvencián keresztül kommunikál.

## Hitelesítés:

- **Statikus:** A felhasználó hitelesíti magát a tokennek, és a token hitelesíti a felhasználót a rendszernek.
- **Dinamikus jelszó generátor:** A token időközönként (pl. percenként) új jelszót generál, melyet kell majd a felhasználónak megadnia. (időszinkronizációs egyszer használatos jelszó)
- **Kihívás-és-válasz:** A rendszer generál egy véletlent (kihívást), melyre a token választ (pl. digitális aláírás) ad.

## Biometrikus hitelesítés

# Biometrikus hitelesítés

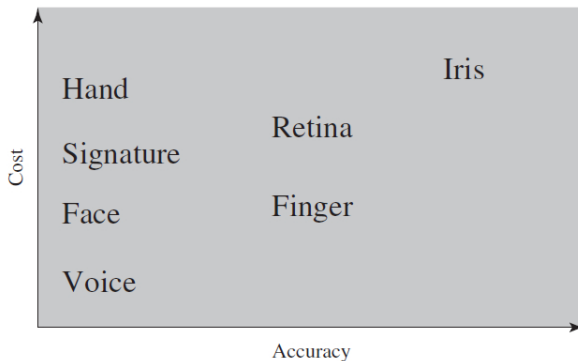
- A felhasználót egyedi, fizikai tulajdonságai alapján hitelesítjük.
- Technológiai szempontból összetettebb és drágább is, mint a jelszavas vagy birtoklás alapú.
- Fizikai tulajdonságok:
  - 1 **Arcfelismerés**: Az arc elemeinek (szem, szemöldök, orr stb.) alakja, elhelyezkedése alapján hitelesítünk. Alternatív megoldás infravörös hőkamera használata.
  - 2 **Ujjlenyomat(ujjnyom)**: A bőrlécrendszer által hátrahagyott nyom a felületeken. Egyediek.
  - 3 **Kéz-geometria**: A kéz alakja, az ujjak szélessége és hossza alapján hitelesítünk.

# Fizikai tulajdonságok

- Fizikai tulajdonságok:

- 1 **Retinal pattern:** A szem hátsó falán található vérerek mintázata egyedi. Alacsony intenzitású infravörös sugarakkal világítja át a leolvasó a szemfenéket. A fejet rögzíteni kell.
- 2 **Írisz:** Az írisz a szem szivárványhártyája. A szivárványhártya képét a szemet egyedivé tevő összes jellegzetességgel (gödröcskék, körök, árkok, korona, szövetszálak) háromdimenziós kontúr-térképpé alakítják.
- 3 **Aláírás:** Mindenkinek egyedi az aláírása, bár egy tulajdonostól többféle minta származhat.
- 4 **Hang:** A hang jobban köthető tulajdonosához, mint az aláírás. Idővel változik.

# Költség vs. Pontosság



## Távoli hitelesítés

- Távoli hitelesítés az Interneten, hálózaton keresztül történik.
- További fenyegetés: lehallgatás, üzenet-visszajátszás.
- A támadások kivédésére kihívás-válasz protokollokat használunk.

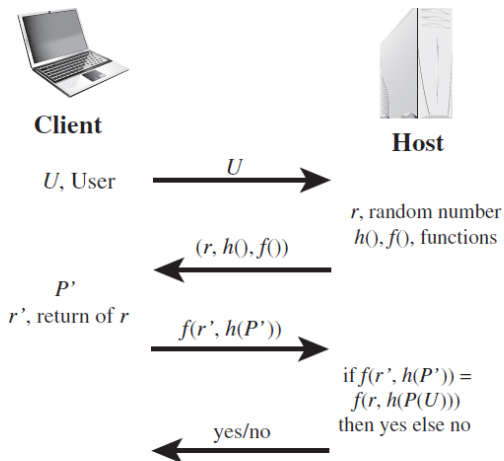
## Távoli hitelesítés: Jelszó protokoll

- A felhasználó elküldi azonosítóját.
- Az ellenőrző fél generál egy  $r$  véletlent, melyet **noncenak** hívunk, és a noncet két függvény nevével visszaküldi.
- Felhasználó válasza:  $f(r'h(P'))$ , ahol  $r' = r$  és  $P'$  a jelszó,  $h$  egy hash függvény.
- Az ellenőrző fél tárolja a hash értékeket:  $h(P(U))$  adott  $U$  felhasználónál.
- Ha  $f(r', h(P')) = f(r, h(P(U)))$ , a felhasználót sikeresen hitelesítettük.

## Biztonsági kérdések:

- A jelszó hash-e van letárolva.
- Az  $f$  védi a lenyomatot a lehallgatástól.
- A nonce a visszajátszásos támadással szemben véd.

# Távoli hitelesítés: Jelszó protokoll

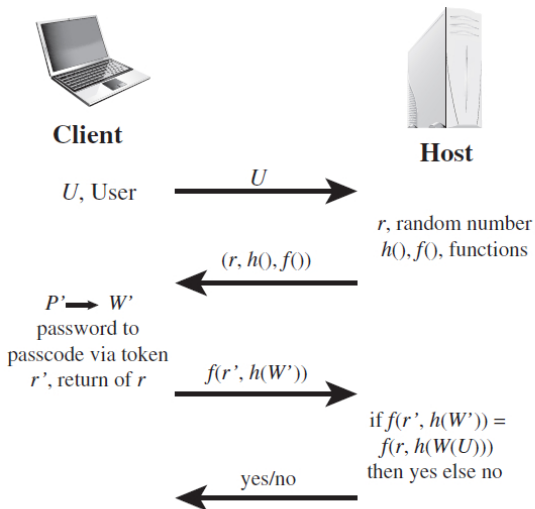


(a) Protocol for a password

# Távoli hitelesítés: Token protokoll

- A felhasználó elküldi azonosítóját.
- Az ellenőrző fél generál egy  $r$  noncet, és a noncet két függvény nevével visszaküldi.
- A token megad egy  $W'$  jelszókódot. Ez a jelszó kód vagy statikus, tárolt, vagy egyszer használatos, generált.
- A felhasználó  $P'$  jelszavának megadásával aktiválja a jelszó kódot.
- A token elküldi:  $f(r'h(W'))$ , ahol  $r' = r$  és  $P'$  a jelszó,  $h$  egy hash függvény.
- Statikus jelszó kód esetén az ellenőrző fél tárolja a  $h(W(U))$  értéket az  $U$  felhasználónál. Dinamikus esetben az egyszer használatos jelszó legenerálása után kiszámítja a lenyomatot.
- Ha  $f(r', h(W')) = f(r, h(W(U)))$ , a felhasználót sikeresen hitelesítettük.

# Távoli hitelesítés: Token protokoll

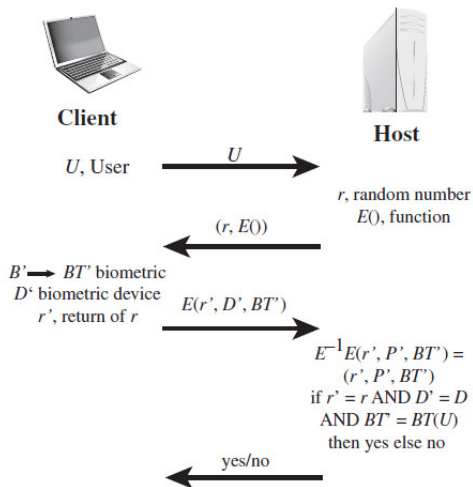


(b) Protocol for a token

# Távoli hitelesítés: Statikus biometrikus protokoll

- A felhasználó elküldi azonosítóját.
- Az ellenőrző fél generál egy  $r$  noncet, és a noncet egy titkosító függvény nevével visszaküldi.
- Kliens oldalon legenerálódik egy  $BT'$  biometrikus minta a felhasználó  $B'$  biometrikus adatából, majd visszaküldi a  $E(r', D', BT')$  titkosított üzenetet, ahol  $D'$  a biometrikus eszköz azonosítója.
- Az ellenőrző fél visszafejtés után a kapott értékeket összehasonlítja a tároltakkal.

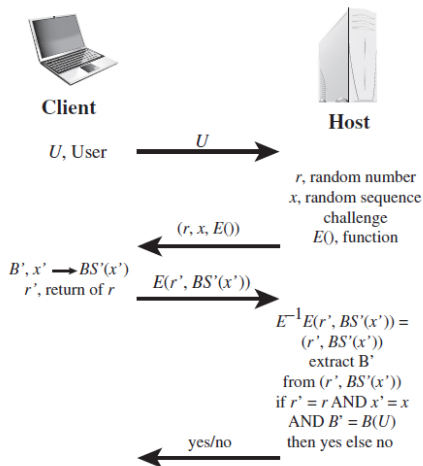
\_\_\_\_\_



**(c) Protocol for static biometric**



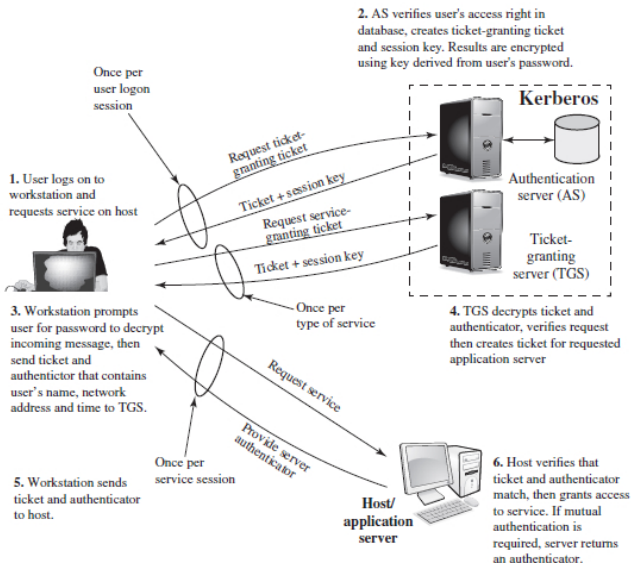
100



#### (d) Protocol for dynamic biometric

## Internet alapú hitelesítés: Kerberos

- Az autentikációt nem több szerver, hanem egy autentikációs szerver végzi.
- Az MIT fejlesztette ki, Internet szabvány.
- Kerberos elvárja, hogy a felhasználó minden szolgáltatásnak hitelesítse magát és minden szerver hitelesítse magát a kliensnek.
- A protokoll résztvevői: kliensek, alkalmazás szerverek és egy Kerberos szerver.
- Az autentikációs szerver (AS) tárolja valamennyi kliens jelszavát egy központi adatbázisban.
- Ha AS hitelesíti a klienst és erről értesíti az alkalmazás szervert. AS minden alkalmazás szerverrel kicserél egy-egy titkos kulcsot.



# Kerberos azonosítási folyamat

- A felhasználó belép egy munkaállomásra és kéri valamely szerver szolgáltatását.
- Kliens processz elkezdődik: A kliens elküldi az ID-t és kéri a TGT(ticket-granting ticket)-et.
- AS visszaküldi: TGT, session key (egyszer használatos titkos kulcs) titkosítását, ahol a kulcs a felhasználó jelszavából származtatott.
- A kliens kéri a felhasználó jelszavát, mellyel visszafejti az üzenetet. Ha a jelszó helyes, akkor a TGT-t és a session key-t megkapja.
- Sem a jelszó, sem a lenyomata nem lett átküldve.

- A ticket jelszi, hogy AS hitelesítette a klienst és a felhasználót. A ticket tartalmazza: felhasználó ID, TGS ID, időbélyeg, érvényességi idő, és ugyanaz a session key. A ticket titkosítva van a szimmetrikus kulccsal, melyet AS és a szerver cserélt ki.
- Ez a ticket hozzáférést biztosíthatna közvetlenül az adott alkalmazás szerverhez, ekkor viszont minden egyes szerver hozzáféréssel az AS-hez kellene fordulni. Ehelyett a ticket TGS (ticket-granting server) számára generált.
- A ticket-tel még több ticket-et lehet igényelni. TGT többször felhasználható.
- Támadó célja: TGT megszerzése
  - AS és TGS közös titkos kulcsával titkosított TGT titkosítva van egy a jelszóból származtatott kulccsal
  - rövid érvényességi idő (8 óra)

## CONCLUSIONS

© 2006 The Authors