Könnyű álmok (9. rész)

Felhasználóazonosítás és a PAM

A Linux többfelhasználós rendszer, ami azt jelenti, hogy egy rendszert több ember használhat – akár egyszerre is, úgy, hogy adataik egymástól megbízhatóan el vannak választva. Ez azonban több kérdést is felvet: hogyan különböztetjük meg egymástól a felhasználókat? Miként tudja egy felhasználó meggyőzően bebizonyítani, hogy ő valóban az, akinek mondja magát? Hogyan választjuk el egymástól a különböző felhasználókat? Ha a felhasználókat már megnyugtatóan azonosítani tudjuk, milyen módon érjük el, hogy több különböző program azonosítási eljárásai hasonlóan vagy egyformán hangolhatók legyenek? Miként érhető el, hogy a felhasználó azonosítása ne csak egy alrendszerre legyen megfelelő? Most a felvetett kérdéseknek csupán az azonosítással kapcsolatos oldalát vizsgáljuk meg.

A felhasználók azonosítása

A többfelhasználós rendszereknek lehetővé kell tenniük a felhasználók megkülönböztetését, hogy képesek legyenek adataik elválasztására. Ha a felhasználóknak nevet adunk, akkor nincs más dolgunk, mint meggyőződni arról, hogy valóban a megfelelő felhasználó kapcsolódott-e a számítógéphez. Ennek ellenőrzésére a leggyakrabban a következő három módszert vagy keveréküket használják:

- a felhasználó valamilyen tudás birtokában van (jelszó, személyi kód (pin), titkos kérdés-válasz párok);
- a felhasználó fizikai azonosítóeszközt birtokol (hagyományos kulcs, CryptoCard, chipkártya);
- a számítógép a felhasználót valamilyen fizikai tulajdonsága alapján azonosítja, ezek a biometrikus azonosítási eljárások (ujjlenyomat, recehártya (retina), hang, szag stb.).

A rendszert nem lehet tökéletesen megvédeni, mivel a jelszó visszafejthető vagy lehallgatható, a felhasználó azonosítóeszközét pedig el lehet rabolni, a hangját rögzíthetik, az ujját levághatják… A megkövetelt módszer vagy módszerek erősségét annak arányában kell eldönteni, hogy milyen típusú a védendő rendszer, mekkora az adat értéke, és milyen erősségű támadásokra lehet számítani. Egy otthoni rendszernél nyilván felesleges az ujjlenyomattal történő azonosítás, míg egy nagy cég pénzügyi adatait tartalmazó gépnél több védelmi szint is megfontolandó (például CryptoCard, valamint erős jelszavak).
Az esetek többségében a módszer ára a döntő. Ha minden monitorra alapkiépítésben ujjlenyomat-olvasót fognak szerelni, akkor az otthoni felhasználók is ezt az eljárást fogják használni.

Kis jelszótörténelem

A jelszavas azonosítás az egyik legegyszerűbb és legolcsóbb biztonsági megoldás. Alkalmazásához nincs szükség semmilyen kiegészítőre, nem igényel hosszú betanulást és magunkkal cipelni sem kell. A Linux-rendszerek alapbeállításban jelszóra épülő azonosítást alkalmaznak, ezért ezt részletesebben kifejtjük.
A korai Unix-rendszereken kutatócsoportok dolgoztak. Az azonosítás szükségtelen volt, hiszen nélküle a munka kényelmesebben folyt, ráadásul akkoriban a rendszert és az adatokat nem kellett még senkitől sem félteniük. Amikor azonban a rendszerek kezdtek elterjedni és hálózatba kapcsolták őket, a fejlesztők hamar rájöttek, hogy valamilyen módon azonosítani kell a felhasználókat és a jogosultságaikat. A jelszavakat először nem titkosítva tárolták, csupán bizonyos feltételekhez kötötték a jelszóállomány hozzáférési jogosultságait. Néha azonban programhibából adódóan a felhasználók hozzáférhetnek egymás jelszavaihoz, ebből következően adataihoz és erőforrásaihoz is. Ezért a rendszer tervezői úgy döntöttek, hogy biztonságosabb lesz, ha a jelszavakat a felhasználók más adataival együtt tárolják ugyan, de titkosítva.

A /etc/passwd állomány

Kezdetben volt a /etc/passwd állomány, ami a felhasználók adatait tartalmazta, beleértve kódolt jelszavukat is. Az állomány a rendszergazda tulajdonában állt, közvetlenül csak ő tudta írni, mivel azonban a benne tárolt adatok nem csak az azonosításhoz voltak szükségesek (a legegyszerűbb `ls` parancs is innen veszi a felhasználói azonosítóhoz tartozó felhasználói nevet), mindenki által olvashatóvá vált, vagyis mindenki hozzáférhetett a felhasználók kódolt jelszavához. Meg kell közelebbről ismernünk az azonosítás folyamatát, hogy a felmerült nehézségek mibenlétére rávilágíthassunk.
A jelszavak kódolására olyan algoritmus használatos, amely egyirányú (úgynevezett hash-függvény), továbbá a jelszó a kódolt alakból nem fejthető vissza (jelenleg nincs rá módszer). Hogyan ellenőrzi a jelszót a rendszer? Nagyon egyszerűen: a felhasználó beírja a jelszavát, a rendszer kódolja, és ha a kódolt alak megegyezik a passwd-állományban tárolttal, akkor a helyes jelszót adták meg. Ebből következik, hogy a jelszavak közvetlenül nem szerezhetők meg. Ha azonban ismerjük egy felhasználó kódolt jelszavát, próbálgatással mi is meghatározhatjuk azt a jelszót, ami az adott kódolt mintát adja vissza. Ez azonban nem feltétlenül egyezik meg az eredeti jelszóval – a jelenséget hash-ütközésnek hívják (***1. ábra***). Amennyiben az eredeti jelszó valamely szótári szó képzett alakja, netán a felhasználó személyéből valamilyen módon kikövetkeztethető (személyi szám, barátnő neve, miegymás), akkor a támadás viszonylag gyorsan kivitelezhető. Elég nagy számítási teljesítménnyel tetszőleges jelszó megtalálható, mivel a hash-függvény értékkészlete véges (1. ábra). A könnyebb megjegyezhetőséghez a felhasználók szívesebben választanak olyan jelszavakat, amelyek valamely szótári szóból képezhetők.
A fent vázolt okokból a korai Unix-kalózok minden elérhető felhasználónak megszerezhették a kódolt jelszavát és célprogramokkal [1.] igyekeztek megtalálni az eredetiket. Az emberek rossz szokásai miatt olyan szótárak álltak össze, amelyek segítségével egy komolyabb felhasználószámú rendszeren másodpercek alatt felhasználók tucatjainak a jelszavát vissza lehetett fejteni. Így a kalózok mások nevében nyomtattak, a processzoridőt és az egyéb erőforrásokat egyaránt használták.

Jelenleg a /etc/passwd állományban a legtöbb rendszeren már nem tárolnak titkosított jelszavakat, de a Linux-telepítőkészletek egy része mind a mai napig rákérdez, akarjuk-e a shadow-támogatást a rendszerünkre telepíteni. Javaslom, mindig válasszuk ki, és most nézzük meg, mit is jelent mindez.

A /etc/shadow állomány

A fent vázolt gondok egy része megoldható lenne, ha az azonosításhoz használt adatokat egy olyan állományban tárolnánk, amelyet kizárólag a rendszergazda tud írni és csupán egy külön csoport tudná olvasni. Ez az állomány lett a /etc/shadow, ami az olyan érzékeny adatokat tartalmazza, melyek az egyéb felhasználást nem érintik (titkos jelszó, mikor kell a jelszót kötelezően lecserélni, mikor jár le a hozzáférés érvényessége és így tovább [2.]). Ezzel a helyi felhasználókat védjük meg jelszavaik ellopásától. Bár egy felhasználó jelszava más úton is megszerezhető (szociális módszerek, login brute force stb.), ezekkel a módszerekkel terjedelmi okokból nem foglalkozunk. Gyakori gond, hogy a jelszóállományt bizonyos SETUID-programok felolvassák, és ha rá tudjuk venni őket egy „jóízű” core dumpra (a program szabálytalan leállása után a memóriában maradt szemetet a rendszer egy core nevű fájlba írja ki) abból kiolvashatóvá válik a kódolt jelszó. Amennyiben odafigyelünk arra, hogy a rendszeren milyen rendszergazdai jogokkal futó programok találhatók, valamint Linux-változatunk figyeli e programok a hibamentességét, ez a hiba nem fordul elő.

1. táblázat **A jelszavak betűszáma**

osztály	leírás	karakterszám
1	digit	10
2	angol kisbetűk	23
3	angol kisbetűk és számok	33
4	angol kis- és nagybetűk	46
5	angol kis- és nagybetűk, számjegyek	56
6	nyomtatható karakterek	95
7	ASCI-karakterek	128

Jelszókódoló algoritmusok

A jelszavak kódolására olyan algoritmusokat használtak, amelyek jelenlegi tudásunk szerint egyirányúak. Ez azt jelenti, hogy a kódolt jelszóból a próbálgatáson kívül semmilyen módon nem állítható vissza az eredeti. Vajon milyen algoritmusokat használnak egy korszerű Linux-rendszerben?

A crypt() és a bigcrypt()

Az első és még ma is sok helyen alkalmazott jelszókódoló algoritmus a crypt(), amely ***Robert Morris*** és ***Ken Thompson*** munkáját dicséri, és a DES-algoritmuson alapul. Működése röviden: az alap DES-algoritmus a nyílt szöveg 64 bites részeinek a titkosítására 56 bites kulcsot használ. A crypt() algoritmus a felhasználó jelszavát titkosító kulcsként használva titkosít 64 nulla bitet, majd annak eredményét és így tovább, mindezt huszonöt alkalommal. A fenti adatokból látszik, hogy a jelszó hossza behatárolt, a használható jelszóhossz pedig nyolc karakter (***1. táblázat***). A végeredményt 11 nyomtatható karakter formájában adja vissza. Ez kerül a ***passwd***- vagy a ***shadow***-állományba. Mivel a DES-algoritmus huszonötszöri végrehajtása a korabeli gépeken akár egy másodpercig is eltartott, az algoritmus ebben a formában tökéletesen megfelelt a kor igényeinek.

A számítógépek rohamos fejlődésével azonban hamarosan túl egyszerűvé vált a jelszavak megkeresése, valamint újabb gond is akadt: ugyanazt a jelszót használva egy másik rendszeren is ugyanaz a titkosított forma állt elő (1. ábra). Morris és Thompson a crypt algoritmusát az úgynevezett ***salt*** hozzáadásával továbbfjlesztették. A ***salt*** egy 12 bites szám, amely a DES-algoritmus végeredményét módosítja. Ezt a kódoláskor a rendszer véletlenszerűen választja ki, így a jelszó visszafejtésének ideje lényegesen növekszik. További előnye, hogy ugyanazt a jelszót két különböző rendszeren beállítva a titkosított alakok eltérőek, így nem nyilvánvaló az azonosság.

Ez természetesen a rendszergazda által átmásolt azonosítókra nem igaz, ilyen esetben akár ugyanazt a jelszót is érdemes újra beállítani. A jelszó hosszának behatároltsága miatt bizonyos rendszereken (HP-UX, Ultrix, BSD) olyan algoritmusokat (crypt16(), bigcrypt()) vezettek be, amelyek 16 vagy akár több karakteres jelszavak használatára is alkalmasak.

Az MD5 és más lehetőségek

A később bevezetett algoritmusok jóval hosszabbak, ezáltal nehezebben megtalálható jelszavak megadását teszi lehetővé. Ezek közül a Linux szempontjából a legnagyobb jelentőséggel a MD5 algoritmus bír. A jelszó hosszát az algoritmus nem korlátozza, csak a ***hash*** értéke mutatja, hogy nagyjából mekkora a lehetséges különböző eredményt adó jelszavak száma (1. táblázat). Az MD5 algoritmust [3.] alkalmazó jelszókódolás is ***salt***-ot használ, ami alapesetben 64 bit, így jóval számításigényesebbé teszi a jelszókeresést. Az algoritmus műveletigénye azonban sajnos nem hangolható, ezért más rendszereken további algoritmusokat fejlesztettek. Az OpenBSD már támogatja a ***bcrypt*** nevű algoritmust, amelynek az erőforrásigénye a `cost` nevű változó segítségével hangolható. Ezzel elérhető, hogy a rendszergazdák jelszavainak megkeresésére nagyságrendekkel nagyobb erőforrást kelljen a támadóknak fordítaniuk.

A jó jelszó titka

Milyen a jó jelszó? Az erős jelszavak előállítása során figyelembe kell venni néhány fontos alapelvet [4.] Ezek nyomán az alábbi jelszavak használatát célszerű elkerülni:

- ha a jelszó valamilyen módon kapcsolatba hozható a felhasználóval (saját, ismerősök vagy házi kedvencek neve, jellemző időpontok, bármely adat, amely a `GECOS` mezőben megtalálható – telefonszámok, személyi azonosító és személyi igazolvány száma, cím, iskolák);
- ha magyar vagy nagyobb világnyelvekben ismert szótári szó bármely formában (akár visszafelé is);
- ha bármely helyi felhasználó neve akármilyen formában (akár többszörözve is);
- ha helységnév;
- ha a billentyűzeten jól ismert minták (qwerty, q1w2e3, qweasd, asdfjkl stb.);
- ha bármely korábban felsorolt forma akár egyetlen számjeggyel kiegészítve (akár visszafelé is).

A jelszó előállításánál a következőket ésszerű betartani:

- legkevesebb nyolc karakter hosszú legyen;

2. táblázat **A választható jelszavak lehetséges száma**

jelszóhossz\osztály	1	2	3	4	5	6	7
4 karakter	1e4	2.8e5	1.2e6	4.5e6	9.9e6	8.1e7	2.7e8
6 karakter	1e6	1.4e8	1.3e9	9.5e9	3e10	7.4e11	4.4e12
8 karakter	1e8	7.8e10	1.4e12	2e13	9.7e13	6.6e15	7.2e16
10 karakter	1e10	4.1e13	1.5e15	4.2e16	3e17	6e19	1.2e21
12 karakter	1e12	2.2e16	1.7e18	9e19	9.5e20	5.4e23	1.9e25
14 karakter	1e14	1.2e19	1.8e21	1.9e23	3e24	4.9e27	3.2e29
16 karakter	1e16	6.1e21	2e24	4e26	9.4e27	4.4e31	5.2e33

- tartalmazzon angol kis- és nagybetűket;
- tartalmazzon számokat;
- lehetőség szerint egyéb jeleket is magában foglaljon (.,:;'"@&#><+!%/=());
- a legjobb esetben vezérlőkaraktereket is tartalmazzon (ezekkel azonban vigyázni kell, mert nem biztos, hogy minden terminál ugyanúgy értelmezi őket);
- legyen könnyen megjegyezhető;
- legyen könnyen begépelhető (gyakoroljuk be!).

Hogyan állíthatunk elő erős, könnyen megjegyezhető jelszavakat? Felhasználható a számos jelszógyártó programok valamelyike (makepassword, pwgen), de az előállított jelszavak ritkán jegyezhetők meg könnyedén. Példa egy előállított erős jelszóra:

```
atya@tensor[18:20][1]$ pwgen -s 14 1
_Rv9T,w[%QB_eU
```

Jóval használhatóbb jelszavak hozhatók létre az alábbi egyszerű módszerrel. Egy olyan mondatot kell kiválasztani, amely könnyen megjegyezhető, majd annak szavaiból bizonyos karakterek kiválasztásával és némi „tupírozással” nagyszerű jelszó nyerhető. Nézzünk rá példát!
A következő mondatot választottam:
A gonosz farkas hosszan üldözte Piroskát az erdőben.
Az ebből nyert jelszó ***+a!ZsNet$Zn.*** lesz.
Egy másik, szintén egyszerű megoldás több rövid szó összeállítása valamilyen egyéb karakterrel. Lássunk egy példát!
fantázia, panasz, lakat
A kapott jelszó: ***fantzia%pansz+lakt!***
Mindkét megoldással az a gond, hogy az előállított jelszavak általában nehezen gépelhetők, ezért gépelés közben könnyebben leleshetők. Az egyre finomabb megoldások felé vezető úton haladva sok olyan jelszó-előállítóval találkozhattunk [5.], amely a fenti kitételek lehető legtöbbjét figyelembe veszi. Megfelelő jelszó választása esetén a kalózok a jelszó kinyeréséhez kénytelenek az összes lehetőség végigböngészésével próbálkozni (brute force).
Megjegyzés: a cikkben közölt jelszavak természetesen semmilyen rendszeren nem használatosak, mivel nyomtatásban is megjelentek.

Az azonosítási rendszerek gyenge pontja

Az egyes Unix-rendszerek fejlesztői hiába dolgoztak ki közösen, kényelmesen használható megoldást, egy gondot mégsem tudtak orvosolni. Minden program csak az előre tervezett és beépített azonosítási eljárásokat támogatta. Bizonyos programok fejlesztői arra vetemedtek, hogy az általánosan használt azonosítási eljárásokat sem támogatták. Mondanom sem kell, egy új azonosítási eljárás beépítése minden azonosítást igénylő programban nagyon bonyolult és kínos feladat.

Általános megoldás: a PAM rendszer

A fenti feladat megoldására hozták létre a PAM-rendszert (Pluggable Authentication Modules). A PAM ötlete először a Sun fejlesztőinek a fejében született meg. Elsőként részlegesen a Solaris 2.5-ös sorozatában vezették be, a teljes támogatás pedig a Solaris 2.6-ban alakult ki. A PAM finoman hangolható azonosítási rendszer, mely a korábban vázolt gondokra oly módon ad egységes megoldást, hogy az azonosítással kapcsolatos feladatokat elválasztja a programtól, így annak tudomást sem kell vennie róla, hogy éppen milyen azonosítási eljárás lett

3. táblázat **Jelszótörés sebessége John The Ripper segítségével**

Algoritmus: Standard DES				
C	több salt	49740 c/s	egy salt	46566 c/s
P	több salt	224345 c/s	egy salt	189811 c/s
Algoritmus: BSDI DES (x725)				
C	több salt	608 c/s	egy salt	417 c/s
P	több salt	513 c/s	egy salt	321 c/s
Algoritmus: FreeBSD MD5				
C	több salt	1092 c/s	egy salt	1092 c/s
P	több salt	1669 c/s	egy salt	669 c/s
Algoritmus: OpenBSD Blowfish				
C	több salt	65 c/s	egy salt	65 c/s
P	több salt	116 c/s	egy salt	116 c/s

C – Kis asztali gép, 450 MHz Pentium II processzorral
P – Kiszolgáló osztályú gép két Coppermine 700 MHz-es processzorral

beállítva. A Linux-rendszer saját PAM-ot használ. A Linux-PAM fejlesztését 1996-ban kezdték meg, legfontosabb fejlesztője a mai napig ***Andrew Morgan***. Az idők folyamán a rendszer rengeteg fejlesztésen ment keresztül. Mi a Linux-PAM 0.72-es vátozatának főbb jellemzőivel ismerkedünk meg.

A PAM lehetőségei

A PAM olyan rugalmas azonosítási rendszert ad a rendszergazda kezébe, mellyel a felhasználókat a rendszeren futtatható bármely PAM-megfelelő program segítségével azonosíthatja [7. 8.]. Az azonosítási módszer a legegyszerűbb bugyuta ***titkos kérdés-titkos válasz*** azonosítási módtól a legbonyolultabb recehártya- vagy ujjlenyomat-felismerésig terjedhet. Morgan a rendszer lehetőségeinek bemutatására az alábbi példát hozza: ha a rendszergazda (mama) a felhasználói (gyerekek) matematikai tudását szeretné tökéletesíteni, beállíthatja, hogy a kedvenc lövöldözős játékuk (természetesen csak ha PAM-megfelelő) azonosításképpen a 12 alatti számok szorzatát kérdezze. Ha a felhasználó tudja a választ, játszhat. Ha nem… Amennyiben a játék érdekes, a gyerekek egykettőre megtanulják a szorzótáblát. A PAM négy különböző szolgáltatást nyújt: azonosítás, felhasználói név (account), munkamenet (session) és jelszókezelés. Ezeket a beállítási állományokban használjuk: `auth`, `account`, `session` és `password`. Az azonosítási szolgáltatás két dolgot jelent: egyrészt a PAM valamilyen eljárással meggyőződik a felhasználó személyéről, azaz azonosítja (authenticate), másrészt a PAM a felhasználót jogosultságokkal ruházhatja fel (authorize) (például valamilyen csoportba sorolhatja be a */etc/group* alapján vagy más módszerrel). A kapcsolatkezelés a rendszeren azonosításfüggetlen szolgáltatások beállítását teszi lehetővé. Jellegzetesen ilyen a kiszolgálók felhasználószámának korlátozása, a rendszergazda bizonyos helyekről (például hálózatról) való belépésének megtiltása. A munkamenet-kezelés szolgáltatásai olyan feladatok megvalósítását teszik lehetővé, melyeknek a felhasználó hozzáférése előtt vagy után kell végrehajtódniuk (naplózás, chroot stb.). A jelszókezelési szolgáltatás a felhasználó azonosítási zsetonjának módosítását teszi lehetővé. A PAM modulokból áll, amelyek akár több szolgáltatást is nyújthatnak.

A PAM beállítása

A PAM a */etc/pam.d* könyvtárban lévő állományokon keresztül hangolható. A beállításokat korábban a */etc/pam.conf* nevű állomány tartalmazta, de elavulttá vált, ezért használatát a Linux-rendszeren nem ajánlom (akad olyan operációs rendszer, amelyben csak ezt alkalmazzák). A */etc/pam.d* könyvtárban lévő állományok neve és a beállítandó szolgáltatás neve kisbetűkkel (su, ssh, ftp stb.) írandó. Az állományokban minden sor (a megjegyzésektől eltekintve) egy PAM-modul valamilyen szolgáltatását hívja meg. Az egyes sorok formátuma a következő:

<modul típusa> <ellenőrző zászló (control flag)>
<modulelérési út> <változó (argument)>

ahol a ***modul típusa*** egy a később megadott modul szolgáltatásai közül. A megadható lehetőségek: `auth`, `account`, `session`, `password`. Jelentésüket korábban már részletesen ismertettük. Az ***ellenőrző zászló*** lehetővé teszi, hogy az adott sorban meghatározott feltétel szükségességét szabályozzuk. Értékei a következők lehetnek:

- required (kötelező): a megadott modulnak sikeresen kell visszatérnie, különben a modultípusban megadott szolgáltatás nem térhet vissza hibátlanul. A modul hibája addig nem jut el a felhasználóhoz, amíg az ugyanilyen modultípussal rendelkező sorok mindegyike ki nem értékelődik.
- requisite (szükséges): az előzőtől csak annyiban különbözik, hogy a hibát a modul azonnal visszaadja az alkalmazásnak.
- sufficient (elégséges): ha a modul sikerrel tér vissza, és a korábban feldolgozott modulok között nincs sikertelenül visszatért ***required*** típusú, akkor erre a szolgáltatásra nem hívódik meg több a később felsoroltakból.
- optional (tetszőleges): miként a neve is mutatja, ez a bejegyzés nem életbevágóan fontos – amennyiben nem önmagában áll, nem okoz sem elutasítást, sem biztos elfogadást. Ilyen esetben csak a beállításainak köszönhetően teszik be a szolgáltatás végrehajtásába (naplóz, valamit kiír, egyéb).

A modul elérési útja megadja a helyét az állományrendszeren, a változók és azok beállítása pedig modulonként és szolgáltatásonként egyediek. A külön nem beállított alrendszerek az ***other*** állományban megadottak szerint vannak azonosítva. Egy kis példa a máshol nem meghatározott PAM-kérések kitiltására:

```
#
# /etc/pam.d/other - default; deny access
#

auth      required /usr/lib/security/pam_deny.so
accoun    required /usr/lib/security/pam_deny.so
password  required /usr/lib/security/pam_deny.so
session   required /usr/lib/security/pam_deny.so
```

A legfontosabb PAM-egységeket és helyes használatukat következő lapszámunkban mutatjuk be.

Hivatkozás jegyzék

[1.] Ismertebb jelszótörő programok: *Openwall Project John The Ripper* ➲ http://www.openwall.com/john/ *Alec Muffett Crack 5.0* ➲ http://www.users.dircon.co.uk/~crypto/ *LC3 (L0phtCrack 3)* ➲ http://www.atstake.com/research/lc3/

[2.] shadow(5) (elérhető a `man 5 shadow` paranccsal)

[3.] MD5: RFC 1321

[4.] GeodSoft: Good and Bad Passwords HOWTO ➲ http://geodsoft.com/howto/password/

[5.] A ***pwgen*** jelszólétrehozó honlapja ➲ http://sourceforge.net/projects/pwgen/

[6.] A Sun PAM honlapja ➲ http://www.sun.com/software/solaris/pam/

[7.] ***Andrew G. Morgan*** Linux-PAM System Administrator's Guide ➲ http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/

[8.] A Linux-PAM honlapja ➲ http://www.kernel.org/pub/linux/libs/pam/

Kötelező olvasmányként ***Simson Garfinkel***, ***Gene Spafford*** tollából a Practical Unix And Internet Security című könyvet ajánljuk. (ISBN: 1-56592-148-8)

Mátó Péter (atya@andrews.hu), informatikus mérnök és tanár. Biztonsági rendszerek ellenőrzésével és telepítésével, valamint oktatással foglalkozik. 1995-ben találkozott először linuxos rendszerrel. Ha teheti, kirándul vagy olvas.