Barkácsoljunk OpenSSL-lel

Az OpenSSL függvénykönyvtárhoz tartozik egy parancssori eszköz, amivel gyakorlatilag mindent kipróbálhatunk, amit a könyvtár lehetővé tesz.

Az *OpenSSL* a *Secure Sockets Layer (SSL)* kriptográfiai protokoll rendkívül jól használható implementációja. Az *Apache a HTTPS*, az *OpenSSH* az *SSH* megvalósításához használja. Bár függvénykönyvtárnak készült, sokoldalú, platformfüggetlen eszközként is hasznosítható.
Kezdetben volt *Eric A. Young SSL* implementációja: az *ssleay*. Jól sikerült továbbfejlesztésével később megszületett az *OpenSSL*, hasonlóan ahhoz, ahogy az *NCSA HTTPd* az *Apache* webkiszolgálóvá változott.
Az *OpenSSL* ma tucatnyi titkosító algoritmust és protokollt támogat, kapcsolók százával.
E rövid történeti áttekintés után, térjünk rá az *OpenSSL* tulajdonságainak bemutatására, melyek közül számos alkalmas *SSL*-ügyfél és -kiszolgáló megvalósítására. Ezen felül a következőket mondhatjuk el róla:

- Rendelkezik az *Egyesült Államok* szövetségi kormányának *NIST FIPS 140-2 1.* szintű minősítésével
- Támogatja az *SSL* következő generációját, a *TLS*-t
- Képes *X.509*-es kulcsot és bizonyítványt előállítani
- Képes *X.509*-es bizonyítványok tanúsítására
- Támogatja az *S/MIME* titkosítást
- Képes a fájlok titkosítására és lenyomatuk (*hash*) elkészítésére
- Képes a *UNIX*-jelszavak lenyomatának elkészítésére
- 9 kereskedelmi forgalomban kapható titkosító hardvereszközt támogat
- Segítségével kriptográfiai teljesítménytesztek végezhetők
- 36 paranccsal vezérelhető
- 6 algoritmust tartalmaz üzenetpecsétek készítéséhez
- 9 titkosító algoritmust támogat, összesen 4 blokkmóddal
- Több kriptográfiai protokollt támogat

Bár az *OpenSSL* meglehetősen bonyolult, ennek nagy részével nem kell szembesülnünk. A cikk hátralévő része a könnyen használható tulajdonságokat mutatja be, mindössze néhány parancssorral.
Most is a korábbi, *GnuPG*-ről írt cikkemben használt fejezetcímeket használom, az *OpenSSL* és a *GnuPG* összehasonlításának megkönnyítése érdekében (a másik cikk *GnuPG trükkök* címmel jelent meg).

Az első lépések

Elsőként győződjünk meg róla, hogy az *OpenSSL* telepítve van és szerepel az elérési utak között. Sok *Linux* disztribúció, köztük néhány kisebb változat is alapértelmezésben tartalmazza az *OpenSSL*-t, amely a legtöbb kötelező csomaghoz hasonlóan a */usr/bin* könyvtárban található.
A parancssori értelmező készenléti jele a következő példák mindegyikében $.

Adjuk ki a következő parancsot:

```
$ openssl version
```

Figyeljünk rá, hogy a version kapcsoló előtt nincs kötőjel! Az eredmény valami ehhez hasonló lesz:

```
OpenSSL 0.9.7d 17 Mar 2004
```

A verziószám, a dátum és az egyéb részletek ettől eltérhetnek. A cikk írásakor a legfrissebb változat az *OpenSSL 0.9.8a* volt. A bemutatott példák minden bizonnyal működni fognak a legtöbb verzióban.
Ha kapcsolók nélkül adjuk ki az `openssl` parancsot, akkor a következőt látjuk:

```
OpenSSL>
```

Ez az *OpenSSL* beépített parancsértelmezője, melynek sem parancssori szerkesztője, sem közvetlen segítségnyújtása nincsen, de egy érvénytelen parancs kiadásakor kiírja az érvényeseket. Egyelőre hagyjuk békén.
Ha mégis ide jutottunk, a `quit` paranccsal vagy a *Ctrl+C* billentyűkombinációval léphetünk ki.

Bináris fájlok védelme

A bináris állományokat elektronikus levélben többnyire *MIME* protokollal küldjük. Ha azonban ezt a levelezőprogram nem támogatja, akkor marad a `uuencode` vagy az *OpenSSL base64* kódolása. Valójában a *MIME* is ez utóbbira épül, de sokkal bonyolultabb és nem is kompatibilis vele.
A következőképpen oldható meg egy fájl szöveges *base64* kódolása:

```
$ openssl base64 < filename.bin
➥> filename.txt
```

Ugyanez visszafelé:

```
$ openssl base64 -d <
➥filename.txt > filename.bin
```

Ne felejtsük el, hogy az *OpenSSL*-t cseppet sem érdekli a fájl kiterjesztése. Eltérően a *GnuPG*-től és a *MIME*-tól, az *OpenSSL* rövid szövegek kódolására is alkalmas:

```
$ echo "The Linux Journal" |
➥openssl base64
VGhlIExpbnV4IEpvdXJuYWwK
```

Ugyanez visszafelé, ahol a `-d` kapcsoló jelzi a dekódolást:

```
$ echo
➥"VGhlIExpbnV4IEpvdXJuYWwK" |
➥openssl base64 -d
The Linux Journal
```

Jobb ellenőrzőösszegek

A `sum` és a `cksum` hagyományos *UNIX* programok ellenőrzőösszegek kiszámítására. A célnak megfelelnek, amíg nincs szükség platformfüggetlenségre, biztonságra, és nem zavar bennünket, ha két különböző fájl esetén ugyanazt az értéket kapjuk.
Bár a legtöbb *Linux* rendszeren alapból megtalálható az `md5sum`, ezt egy nemrégiben felismert biztonsági rés miatt nem célszerű használni.
Ha a biztonságosabb sha1sum telepítve van, használjuk inkább azt. Vigyázzunk, mert számos különféle program fut ezen a néven. Némelyik egyszerre csak egy fájllal tud megbirkózni, esetleg nem tud adatot fogadni a szabványos bemenetről, vagy valami egészen más bajjal küszködik. Ha ilyen problémával szembesül, vagy egyszerűen csak egy konzisztens, közismert, platformfüggetlen szoftverre van szüksége, fontolja meg az *OpenSSL* használatát.
Az *OpenSSL* kimeneti formátuma kissé eltér *GnuPG*-étől, de tartalmilag természetesen megegyeznek. Az *OpenSSL* eredménye mindig tartalmazza a felhasznált algoritmust és a lenyomatot, csupa kisbetűvel, üres karakterek nélkül. Egyesek szerint ez a formátum könnyebben használható. Néhány példa:

```
$ openssl sha1 filename
SHA1(filename)=
➥e83a42b9bc8431a6645099be50b63
➥41a35d3dceb
$ openssl md5 filename
MD5(filename)=
➥26e9855f8ad6a5906fea121283c7
➥29c4
```

A hasonszőrű, *GnuPG*-ről szóló cikkem példáiban a fájl tartalma *„The Linux Journal”* volt. Fontos, hogy a *Journal* után nem szerepel pont.
Ha valami okból nem sikerül a fenti eredményeket reprodukálni, a fájl tartalmának hexadecimális leírása talán segíthet az ok megtalálásában. Érdemes odafigyelni a sor vége karakterre, amit a `vi` automatikusan adott a fájlhoz:

```
T h e     L i n u x
➥J o u r n a l \n
54 68 65 20 4c 69 6e 75 78 20
➥4a 6f 75 72 6e 61 6c 0a
```

Míg a *GnuPG*-ben van *SHA-512*, az *OpenSSL*-ben nincs, amit talán kárpótol a régi *MD2*, *MD4* és *MDC2* algoritmusok támogatása, a visszafele kompatibilitás érdekében. Az *MD5*-höz hasonlóan, már ezek használata sem ajánlott.

Gyors és egyszerű titkosítás

Az *OpenSSL*-lel fájlokat is titkosíthatunk, bár nem ez a legfőbb erénye. Rugalmassága miatt kissé bonyolultabb a használata, mint a *GnuPG*-é.
Nagyon kevés az alapértelmezett érték, így több kapcsolót kell használni. Több algoritmust is támogat, amik közül választani kell. Ezek közül néhányat, például a *DES*-t és az *RC4-40*-et csak a visszafele kompatibilitás érdekében tartottak meg, de használatuk többé nem ajánlott. Használjunk erős algoritmusokat, például a `bl`-t (*Blowfish*) és a 128 bites, a titkosító blokkok láncolásával működő `aes-128-cbc`-t (*US NIST Advanced Encryption Standard*).
Íme egy példa:

```
$ openssl enc -aes-128-cbc <
➥filename > filename.aes-128-
➥cbc
enter aes-128-cbc encryption
➥password:
Verifying - enter aes-128-cbc
➥encryption password:
```

A *GnuPG*-hez hasonlóan az *OpenSSL* is kétszer kéri a jelmondatot, amit nem jelenít meg a képernyőn. A `-d` kapcsolóval kérhető visszafejtés is valamivel nehézkesebb:

```
$ openssl enc -d -aes-128-cbc
➥-in filename.aes-128-cbc >
➥filename
enter aes-128-cbc decryption
➥password:
```

A *GnuPG*-től eltérően, az *OpenSSL* nem találja ki magától a fájl típusát, az algoritmust, a kulcs hosszát és a módot. Ezt nekünk kell megtennünk. A fenti példában ezeket az adatokat a fájl kiterjesztésében helyeztem el. Az *OpenSSL* nem gondoskodik a fájlok és a kiterjesztések kezeléséről, nekünk kell megmondanunk, hová kerüljön az eredmény.
Ha az algoritmust nem adjuk meg, akkor az *OpenSSL* vagy szemetet ad, vagy hibás varázsszámra panaszkodik. A visszafejtés egyik esetben sem fog sikerülni. Őszintén szólva az *OpenSSL* nem igazán erre a célra készült, de erre is használható.

Jelmondatok

A lendület hevében időzzünk el egy kicsit a jelmondatok fontosságának témakörénél. A legtöbb kriptográfiai rendszerben létezik olyan speciális jelszó, az úgynevezett jelmondat (*passphrase*), amely további titkokat véd. Mivel ez a leggyengébb láncszem, fontos, hogy nehezen megfejthető legyen. Az erős jelmondat képzése azonban a megfelelő eszköz nélkül rendkívül nehéz feladat. Az *OpenSSL* egy ilyen megfelelő eszköz.
Az első ökölszabály, hogy az erős jelmondat hosszú, és a 8 karakter biztosan kevés (lásd az *1. Táblázatot*).
Az általános cél az, hogy olyan titok legyen a birtokunkban, amire könnyen emlékszünk, más nem tudja, nem képes kitalálni, és véletlenül sem ejti ki a száján.

Jelmondat képzése

Az *OpenSSL* segítségével gyorsan készíthetünk nagyon erős jelmondatokat, például:

1. táblázat *A jelmondatok erőssége, illetve a kitalálásukhoz szükséges idő nagysága (amely a körülményektől függően rendkívül változó lehet)*

Típus	Bájt	Karakter	Bit/karakter	Bithossz	Megfejtés ideje
Base64 [A-Za-z0-9+/=]	6	8	6	48	percek-órák
Base64 [A-Za-z0-9+/=]	9	12	6	72	évek
Base64 [A-Za-z0-9+/=]	12	16	6	96	évtizedek
Base64 [A-Za-z0-9+/=]	15	20	6	120	végtelen?
Diceware jelmondat		8 szó	12.9 / szó	120	végtelen?

```
$ openssl rand 15 -base64
wGcwstkb8Er0g6w1+Dm+
```

Minden futtatásnál más-más eredményt kapunk, hiszen a jelmondat véletlen alapú.
Az első paraméter (a példában 15) határozza meg, hány bájt hosszú legyen a generált jelszó, a második (a példában -base64) a karaktereket kódoló algoritmust állítja be. 15 byte esetén a kimenet mindig 20 bájt hosszú lesz, nem számítva a sor vége karaktert.
A base64 karakterkészletben megtalálhatók a kis és nagy betűk, a számok 1-9-ig, továbbá a +, a / és az = írásjelek. A karakterkészletet szándékosan korlátozták ezekre a jelekre, és a több nem is feltétlenül jobb. Biztonsági szempontból ez mindössze egyetlen karakter hozzáadásával kompenzálható, mert egy 8 karakteres ***ASCII*** jelszó körülbelül olyan erős, mint a kilenc karakteres base64-es társa.
Ha nem is olyan sebesen, mint az ***OpenSSL***, a ***Diceware*** erős és gyakran könnyen megjegyezhető jelmondatokat állít elő. Melegen ajánlom.

Titkosított jelszavak

Végre valami, amit a ***GnuPG*** már egyáltalán nem tud: az ***OpenSSL***-ben van egy beépített parancs, amivel pontosan olyan titkosított jelszavakat készíthetünk, mint amilyet a ***/bin/passwd***-vel.
Ha az Olvasót zavarja a szőrszálhasogatás, ezt a fejezetet inkább ugorja át. Bár a ***Linux*** jelszavait gyakran titkosítottnak nevezik, azokról valójában lenyomat készül az ***MD5*** vagy a régi (***DES*** titkosító algoritmuson alapuló) ***UNIX*** jelszópecsét-algoritmussal. Ezáltal válik lehetővé, hogy a ***Linux*** akkor se tudja a jelszót, amikor megbizonyosodik arról, hogy a felhasználó helyesen adta meg. Amikor a felhasználó beállítja a jelszavát, arról lenyomat készül a ***/etc/shadow*** fájlba. Bejelentkezéskor a begépelt jelszóról ismét lenyomat készül, és ha ez megegyezik a ***/etc/shadow*** fájlban lévővel, akkor a kapu kitárul. Ellenkező esetben rossz jelszót adtunk meg, és a gép természetesen továbbra sem tudja a helyeset.
Nos, ezért rendkívül hasznos, ha saját jelszópecsétet készítünk. Tegyük fel, hogy egy másik számítógépen van szükségünk jelszóra. Lehet ez egy új felhasználói fiók miatt, vagy mert elfelejtettük és megkértük a rendszeradminisztrátort, hogy hozza alaphelyzetbe. Ha tudunk beszélni a rendszeradminisztrátorral, akkor könnyű helyzetben vagyunk, de mi van, ha mégsem? Lehet, hogy még sohasem találkoztunk vele korábban. Hogy beszéljük meg vele az új jelszót?
Az elektronikus levél nem biztonságos, de a telefon sem sokkal jobb.
A hagyományos posta napokig tartana, a biztonsági problémákról nem is beszélve. A fax, a szöveges üzenetküldő és a csipogó sem különb.
És még ennél is lehet rosszabb. Talán nem is bízunk meg a rendszergazdában. Való igaz, hogy ő a root felhasználó, de a jelszó ismerete túlmutat a hatáskörén. Lehet, hogy ugyanazt a jelszót akarjuk használni több számítógépen is, és azok rendszergazdáiban sem bízunk.
Szóval, ha paranoiásak vagyunk, tegyük a következőt:

```
$ openssl passwd -1
Password:
Verifying - Password:
$1$zmUy5lry$aG45DkcaJwM/GNlpBLT
➥DyO
```

A jelszót kétszer kell megadni, nem látszik a képernyőn. Ha több felhasználói fiókunk van, hajtsuk végre a parancsot többször. Az eredmény a jelszó véletlennel fűszerezett kriptográfiai lenyomata, így minden futtatáskor más és más eredményt ad, még akkor is, ha a jelszó ugyanaz. Példánkban a jelszó lenyomata:

```
$1$zmUy5lry$aG45DkcaJwM/
➥GNlpBLTDyO
```

Ha ugyanezt kipróbáljuk, a kezdeti `$1$` karaktereket leszámítva teljesen más kimenetet kapunk.
A lenyomatot nyugodtan faxolhatjuk, elküldhetjük elektronikus levélben, szöveges üzenetben, de akár szóban is megadhatjuk a rendszergazdának, hogy ezt állítsa be jelszópecsét gyanánt a ***/etc/passwd*** fájlban manuálisan vagy a `chpasswd` paranccsal. Utóbbi esetben szükség van egy ideiglenes fájlra, nevezzük `newpassword`-nek, benne a felhasználónévvel és a jelszólenyomattal, például:

```
felhasznalonev:$1$ywrU2ttf$yjm9
➥OXTIBnoKJLQK2Fw5c/
```

A fájlban több sor is szerepelhet, más felhasználói fiókokhoz.
A rendszergazda eztán root felhasználóként futtatja a következő parancsot:

```
chpasswd –encrypted <
➥newpassword
```

Az új jelszó beállítása ezzel befejeződött. Hacsak nem egy elképesztően erős jelmondatot választottunk, tanácsos a jelszót megváltoztatni az első belépéskor. Erre azért van szükség, mert a lenyomat ismeretében az eredeti jelszó a nyers erő stratégiájával megfejthető, de minél erősebb a jelszó, annál hosszabb idő alatt.
A jelszó megváltoztatásának ez a módja elég biztonságos. Ha például valaki megszerzi a jelszópecsétet, és megtudja, hogy melyik számítógépen található a hozzá tartozó felhasználói

2. táblázat *Lenyomatkészítés és blokkos titkosítás teljesítménye (a számok másodpercenként 1000 bájtban vannak megadva, 1024 bájtos blokkmérettel)*

	AMD K6-2 300MHz, Linux 2.6.12, OpenSSL 0.9.7g	AMD Athlon 1.333GHz, Linux 2.4.27, OpenSSL 0.9.7d	PowerMac G5 1.6GHz, Darwin Kernel Version 8.0.0, OpenSSL 0.9.7b
md5	26,733.93k	169,207.13k	76,921.71k
sha1	12,665.41k	113,973.25k	76,187.82k
blowfish cbc	9,663.40k	56,940.54k	44,433.14k
aes-128 cbc	5,134.78k	31,355.90k	54,987.78k

fiók, nem sokat ér vele, hiszen magát a jelszót csak a lenyomatot előállító személy tudja. A jelszópecsét akár egy népszerű magazinban is megjelenhet.
Apropó, a példa jelszava 28 véletlenszerűen választott base64 karakterből áll, így rendkívül nehezen törhető. Ennek ellenére ezt véletlenül se használjuk, mert a jelszó is szerepel a cikkben, itt:

```
HXzNnCTo8k44k8v7iz4ZkR/QWkM2
```

A jelszó és a lenyomat a következő parancsokkal készíthető el:

```
$ openssl rand 21 -base64
HXzNnCTo8k44k8v7iz4ZkR/QWkM2
$ openssl passwd -1
➥HXzNnCTo8k44k8v7iz4ZkR/
➥QWkM2
```

A példák a Linux rendszerekben használatos ***MD5*** lenyomatokat használják. Ha a régi ***UNIX*** jelszópecsétekre van szükség, egyszerűen hagyjuk el a -1-et:

```
$ openssl passwd
Password:
Verifying - Password:
xcx7DofWC0LpQ
```

A jelszó ebben a példában: ***TheLinux***

Kriptográfiai teljesítményteszt

Mivel az ***OpenSSL*** sok algoritmust támogat, jól használható kriptográfiai benchmarkok készítéséhez. A konzisztens kód alapján összevethető a kriptográfiai algoritmusok és hardverarchitektúrák teljesítménye. Ráadásul még beépített benchmark parancsa is van.

3. táblázat *Nyilvános kulcsú titkosítás teljesítménye*

	aláírás/s	ellenőrzés/s	
rsa 1024 bit	30.8	563.1	AMD K6-2 300MHz, Linux 2.6.12, OpenSSL 0.9.7g
dsa 1024 bit	61.6	50.7	AMD K6-2 300MHz, Linux 2.6.12, OpenSSL 0.9.7g
rsa 1024 bit	239.9	4,514.6	AMD Athlon 1.333 GHz, Linux 2.4.27, OpenSSL 0.9.7d
dsa 1024 bit	498.2	410.6	AMD Athlon 1.333 GHz, Linux 2.4.27, OpenSSL 0.9.7d

Az `openssl speed` parancs alapértelmezés szerint minden egyes algoritmust kipróbál minden létező móddal és kapcsolóval, különféle méretű bemeneti adatokkal. Ez utóbbi az algoritmusok induló vesztesége miatt fontos. A teljes benchmark teszt mintegy 6 percig tart, függetlenül a hardver teljesítményétől, és 124 sornyi teljesítményadatot állít elő, 29 sor összegzéssel.
Jegyezzük meg, hogy a kriptográfiai algoritmusok teljesítménye jelentősen függ az implementációtól. A nagyobb sebesség érdekében az ***OpenSSL*** kódja több algoritmusban is x86-os assembly részekkel tarkított. Más architektúrákhoz, például az ***ia64***-hez, a ***SPARC***-hoz és az ***x86-64***-hez sokkal kevesebb, míg megint másokhoz egyáltalán nem tartozik assembly kód. A gépi kódok a ***crypto/*/asm*** könyvtárakban találhatók. A 2. és *3. Táblázat* három különféle rendszer eredményeit mutatja.

Merre tovább

A cikk csak ízelítő abból, amire az ***OpenSSL*** képes a parancssorban. Található dokumentáció az ***OpenSSL*** weboldalán a dokumentumok, illetve a kapcsolódó anyagok címszó alatt, és több levelezőlista is a támogatás címszó alatt.
Az ***OpenSSL***-t ***C/C++***-ban írták ***C/C++***-hoz, de számos más nyelvre is adaptálták, így például ***Ruby***-ra. Mindezeken túl, a 2006 márciusában elnyert ***FIPS 140-2 1.*** szintű minősítés az ***OpenSSL***-t komoly versenyhelyzetbe hozta a kriptográfia vállalati és kormányzati piacán.

Linux Journal 2006., 147. szám

Anthony J. Stieber az informatikai biztonság szakértője és a UNIX rendszerek, a kriptográfia, a fizikai biztonság és néhány ránk nem tartozó dolog megszállottja. Mostanában az Egyesült Államokban, a Minnesota állambeli Minneapolisban melegszik. Ez a második cikke.

KAPCSOLÓDÓ CÍMEK

A cikkel kapcsolatos további anyagok a ➲ www.linuxjournal.com/article/9020 címen találhatók.