Beköszöntő

Februári számunk fő témája a biztonságtechnika.

Ha az ember ***Linuxot*** használ, az persze önmagában is viszonylag nagy biztonságot jelent a mai „vírusterhes” időkben, ugyanakkor két dolgot nem szabad elfelejtenünk.

Egyrészt teljes biztonság nem létezik, tehát azt a linuxos rendszergazdát, aki elhanyagolja a rendszerét, előbb vagy utóbb de biztosan meglepetés fogja érni. Hogy miként maradhatunk naprakészek a biztonsági frissítésekkel, azt ***Jeremy Turner*** cikkéből tudhatjuk meg.

A másik lényeges szempont, hogy a ***Linux*** használói között számos olyan ember akad, aki nem csupán a saját gépének biztonságáért felel, hanem mások testi, lelki és digitális épségét is a szívén viseli. Egyesek puszta felebaráti szeretetből teszik ezt, míg mások azért, mert kifejezetten ezért fizetik őket.

A rendszer belső biztonságát nyilván nagyban növeli, ha a különböző objektumokkal, és a rajtuk végezhető műveletekkel kapcsolatos jogosultságokat a felhasználók szerepe, munkaköre alapján osztjuk ki. Mivel a UNIX hagyományos jogosultsági rendszerével bizonyos dolgokat nehéz, vagy egyenesen lehetetlen megoldani, az NSA elkészített a Linuxhoz egy speciális biztonságtechnikai kiegészítést (***Security Enhanced Linux***), amely képes a szerepek és jogosultságok összerendelésére. Egy ilyen megerősített rendszer hálózati funkcióit tekinti át ***James Morris*** cikke.

Egy olyan világban, ahol a posta szerepét egyre inkább átveszi az elektronikus levél, sőt a kormányhivatalok is törekednek az elektronikus ügyintézésre, nyilván egyre komolyabb veszélyt jelent az üzenetek lehallgatása, vagy rosszindulatú módosítása. ***Roy Hobler*** erről a témáról, vagyis az elektronikus levelek gyors és egyszerű titkosításáról ír.

A biztonságtechnika egyik alapszabálya, hogy soha nem szabad alábecsülni a triviális hibalehetőségeket. Lehet bármilyen jól szervezett egy gép belső biztonsági rendszere, lehetnek akármilyen körmönfont módon titkosítva a hálózati kapcsolatai, bárki hozzáférhet a teljes adattartalmához, ha egyszerűen ellopja. ***Mike Petullo*** cikkéből megtudhatjuk, miként titkosítható ***Linux*** alatt akár a gyökér fájlrendszer is.

Természetesen folytatódik néhány sorozat is.

Beszédes Balázs a felhasználók jószándékú megfigyelésének immár egészen gyakorlati módszereit tárgyalja, ***Auth Gábor*** a ***FreeBSD***, ***Komáromi Zoltán*** pedig a ***PHP 5*** bemutatásával haladt tovább. ***Illés Viktor*** ugyanakkor egy új, a ***Samba 3*** képességeit tárgyaló sorozatba kezdett bele. Ez a rendszer azért különösen érdekes, mert egyesek a ***Windows 2003 Server*** „utódát” látják benne. ***Garzó András*** a hálózatokról szóló sorozat immár 14. részében a forgalomirányítás rejtelmeit taglalja, ***Dudás Ágnes*** pedig arról ír, hogy szerzői jogi szempontból mi a helyzet az olyan alkotásokkal, amelyek munka- vagy szolgálati viszony keretében keletkeztek.

Jó szórakozást kíván
a Linuxvilág stábja!

Környezetvédelem csak eredetiben

Kibővült az *OKI* kellékanyagok visszavételére irányuló környezetvédelmi programja. A cég július 4-től már visszaveszi a használt tintapatronokat, ezt a lehetőséget most a lézernyomtatók tonerjeire és fényhengereire is kiterjesztették. Sőt, nem egyszerű visszavételről, hanem – igaz, jelképesnek mondható áron – visszavásárlásról van szó, a kedvezményt új, eredeti *OKI* kellékanyag vásárlásakor lehet érvényesíteni. Sajnos az *OKI* magyar webhelyén található tájékoztatás meglehetősen szűkös (idézet: „Ha igénybe kívánja venni ezt a lehetőséget, látogasson el a részletekért saját országának webhelyére."), ezért a felhasználók elsősorban a kereskedőkhöz fordulhatnak.

➲ www.okihu.hu

Kipróbálható az Nvu

Múlt év decemberének elején megjelent az *Nvu* weblapfejlesztő alkalmazás első próbaváltozata, amelyet kicsivel több mint egy hónappal később újabb próbakiadás követett. A *Linux* és *Windows* alá készülő, utóbbi alatt a jól ismert *FrontPage* és *Dreamweaver* programokat versenyre hívó *Nvu* segítségével a *HTML* nyelv ismerete nélkül is bárki készíthet weboldalakat és gondozhat webhelyeket. Az *Nvu* a *Mozillában* is megtalálható *Gecko* megjelenő motorra épül, tőle örökli az *XML*, a *CSS* és a *JavaScript* támogatást. A többek közt *FTP* alapú webhelykezelővel, *CSS* szerkesztővel, testreszabható eszköztárakkal, űrlapszerkesztővel, kódellenőrzővel és – tisztítóval, helyesírás-ellenőrzővel rendelkező programra minden hasonló területen dolgozónak érdemes lehet egy pillantást vetnie.

➲ www.nvu.com

Betörők, mosolyogni!

Az olasz *CFD Elettronica* beágyazott *Linuxra* épülő otthoni riasztórendszert mutatott be. A *Plenitude Premium* rendszer többféle egységet, például hagyományos érzékelőszemeket is magába foglal, ám a legérdekesebbek ezek közül az infravörös érzékelőkkel irányított vezeték nélküli kamerák. Ezek, ha mozgást észlelnek, két fekete-fehér fotót készítenek és továbbítanak a központi egység felé, amely viszont képes arra, hogy a képeket *GPRS*, *Bluetooth* vagy *Wi-Fi* összeköttetésen keresztül mobiltelefonra vagy egyéb eszközre küldje tovább. A központi egység saját kijelzővel is rendelkezik, ezen igény szerinti mozgóképes felügyeletre vagy a korábbi eseményekhez tartozó képek visszanézésére nyílik lehetőség. A központ egy 400 MHz-es *Intel XScale* processzort, 64-128 MB memóriát és 32-64 MB Flash memóriát tartalmazó egylapkás számítógépet rejt, operációs rendszere a 2.4-es *Linux* rendszermagra épül. Az üzletekbe várhatóan csak nyáron kerülő rendszer ára 4500 euró körül fog alakulni.

➲ www.cfd.it

Egy gép előtt a család

Az *Open Sense Solutions Groovix* néven olyan számítógépeket mutatott be, amelyek egyszerre akár három felhasználónak is képesek teljes értékű munkakörnyezetet biztosítani. Hasonló megoldások már eddig is léteztek, ezek például egy kiegészítő kártya révén oldották meg a második felhasználó csatlakoztatását, ám a *Groovix* esetében csak külön VGA- és – szükség esetén – hangkártyákra van szükség, a megosztás többi részét szoftverkönyvtárakra bízták. A *Groovix* fontos jellemzője, hogy a saját billentyűzet, egér és kijelző mellett mindegyik felhasználó 3D gyorsított megjelenítést és saját hangkártyát kap. A *Groovix* gépek hardveres oldalról ugyan minőségi, de teljesen hétköznapi összeállítások, különlegességük a *Simultaneous Local Independent Multi-User (SLIM, egyidejű, helyi, független többfelhasználós)* megoldásban rejlik – ezek a *Debian Linuxra* készített könyvtárak gondoskodnak a felhasználók elkülönítéséről. Mivel a megosztás szoftver alapú, a felhasználóknak nem javasolják, hogy más operációs rendszert telepítsenek a gépre, illetve, ha ezt mégis megteszik, akkor le kell mondaniuk a többfelhasználós működésről. Az egyetlen felhasználót kiszolgáló gépet a cég 599, a kettős használatra tervezettet 799, míg az erőforrásokat három fő között megosztót 999 dollárért kínálja.

➲ www.opensensesolutions.com

phpBeans

A *Simian Systems Inc.* bejelentette a *phpBeans* előírásgyűjteményt, melyet követve, alkalmazva a fejlesztők valóban nagyvállalati szintű PHP-s alkalmazásokat készíthetnek – a cég állítása szerint erre eddig nem volt lehetőség. A *phpBeans* olyan módszert ír körül, melynek segítségével a különböző gépeken létező objektumok egymással átlátszó módon, távoli eljáráshívással léphetnek kapcsolatba, így a különféle programszintek egymástól könnyedén elkülöníthetők. A tervezet része – bár a fő szerepet a szabványos távoli eljáráshívási módszerek, és nem a tényleges programok játsszák – a *phpBeans Object Server*, egy objektumkonténer kiszolgáló, mely hitelesített tranzakciók lebonyolítását teszi lehetővé az általa tárolt objektumok és a csatlakozó ügyfelek között; a *phpBeans Client API*, mely egy nemcsak *PHP*, hanem más nyelvek alá is elérhető könyvtár, segítségével a programozók kéréseket indíthatnak a távoli *phpBeans Object* kiszolgálók felé; valamint a *phpBeans* protokoll, mely egyszerű, gyors és hatékony adatátviteli rétegként segíti a tranzakciók lebonyolítását. A *phpBeans* objektumkiszolgáló *GNU GPL* és kereskedelmi szerződéssel, míg az ügyfél *API GNU LGPL* szerződéssel érhető el, így mindegyik felhasználó megtalálhatja a számára leginkább megfelelő konstrukciót.

➲ www.phpbeans.com

Ön vett már IBM üzletágat?

A vezető kínai PC-gyártó, a *Lenovo Group Limited* felvásárolta az *IBM PC*-gyártó üzletágát. A körülbelül 1,75 milliárd dolláros üzletkötés nyomán a Lenovo a világ harmadik legnagyobb PC-gyártója lesz, éves forgalma a becslések szerint 12 milliárd dollár körül fog mozogni. A megegyezés szerint a két cég között nemcsak egyszerű adásvétel történt, hanem hosszú távú értékesítési, szolgáltatási és pénzügyi szövetséget hoznak létre. Az egyesült vállalat a világ 160 országában lesz jelen, *Kínában* a *Lenovo*, azon kívül pedig az *IBM* által korábban bevezetett márkanevek révén lesz ismert. A gyártás a *Lenovo*, a szolgáltatások biztosítása pedig az utóbbi időben hangsúlyosan a szolgáltatások felé forduló IBM feladata lesz.

➲ www.lenovogrp.com

Sokcsatornás MP3

A *Fraunhofer Intézet* oldalára felkerültek az első, a térbeli hangzást kínáló *mp3* formátummal kapcsolatos anyagok. Az egyelőre csak *Windows* és *Mac OS X* alá elérhető, ingyenesen letölthető próbacsomagok fejlesztői készletből, surround dekóderből, kódolóból, lejátszóprogramból állnak, illetve a windowsos változat *Winamp* beépülő modult is tartalmaz. Az új formátum kipróbálását néhány szintén szabadon letölthető dallal is segítik, az intézet ígérete szerint a mintakészlet hamarosan bővülni fog. Az új *mp3* formátum 5+1 csatorna hangjának közvetítésére alkalmas, az újfajta kódolású fájlok a csak a régebbi, sztereó változat kezelésére képes eszközökön is lejátszhatók, miközben méretük nem sokkal haladja meg a csupán két csatornát tároló állományokét. A mintaprogramok futtatásához legalább 1 GHz órajelű *Pentium III* vagy azzal egyenértékű processzorral, 128 MB memóriával és 5.1-es hangkártyával felszerelt gépre és a *DirectX 9.0* vagy újabb változatára van szükség.

➲ www.iis.fraunhofer.de/amm/download/mp3surround/index.html

Megújult LinuxSecurity.com

Megújult, kibővült a linuxos és nyílt forrású programokkal kapcsolatos biztonsági kérdésekkel foglalkozó *LinuxSecurity.com* webhely. A még 1996-ban indított, a *Guardian Digital* munkatársai által gondozott hely az elmúlt évek során az informatikai szakemberek és a nyílt közösség tagjai körében egyaránt kedvelt információforrássá vált, oldalain a világ minden tájáról származó írások, útmutatók, leírások jelennek meg, de a biztonsági frissítések beszerzésében, továbbá fórumok formájában az eszmecseréhez is segítséget kínál a látogatóknak.

➲ www.linuxsecurity.com

Médiatitkár a zsebben

Elvileg már ebben a hónapban kapható lesz az *Archos Linux* alapú személyi videófelvevő és -lejátszó készüléke. A *Pocket Media Assistant 400*, röviden *PMA400* jelzésű készülék 30 GB kapacitású merevlemezzel, *USB*, *Wi-Fi* és infravörös csatolóval, 320x240 képpont felbontású, 3,5"-os *TFT* kijelzővel és TV-kimenettel rendelkezik, operációs környezete *Qtopia* alapú, így a multimédiás szolgáltatásokon túl – kiegészítő billentyűzettel – személyi adatkezelésre és egyéb alkalmazások futtatására is alkalmas. A kisméretű, tudásához képest meglepően könnyű, 280 grammos készülék az *MPEG-4* és az *mp3* formátumot támogatja, előbbi révén közel DVD-minőségű mozgóképek tárolására és lejátszására képes. Ára 800 euró.

➲ www.archos.com

Zöld út a magyar UMTS hálózatoknak

A *Nemzeti Hírközlési Hatóság* kihirdette a hosszas késlekedés után tavaly nyáron kiírt, harmadik generációs *UMTS* mobitelefon-hálózatok kiépítésére vonatkozó pályázat eredményét. A Hatóság döntése szerint mindhárom Magyarországon már jelen lévő mobilszolgáltató – *Pannon GSM Távközlési Rt.*, *T-Mobile Magyarország Rt.*, *Vodafone Magyarország* – jogot nyert a bővítésre, míg a negyedik blokkot, amellyel egy újabb szereplő piacra lépését szerették volna ösztönözni, egyelőre senkinek nem ítélték oda. A harmadik generációs hálózatok beindítása 2006-ban várható, segítségükkel a mostaninál gyorsabb adatátvitelre, jobb hangminőségű telefonálásra – amit talán jobb lesz, ha a továbbiakban hangtelefonálásnak hívunk –, videotelefonálásra, multimédiás szolgáltatások igénybe vételére lesz lehetőségünk. A három cég a jogokért több évre elosztva összesen több mint 50 milliárd forintot fizet be az államkasszába.

➲ www.nhh.hu

Medgyesi Zoltán
(mz@rettesoft.hu)
A Linuxvilág hírszerkesztője. Szabadidejét legszívesebben a barátnőjével tölti, szeret autózni és bográcsban főzni.

Fal, amely összeköt – az FSN.hu tégla projektje

Az FSN.hu alapvető célja, hogy megfelelő infrastrukturális hátteret biztosítson a szabad szoftverekkel foglalkozó emberek és szervezetek számára. A hardvernek azonban megvan az a rossz szokása, hogy viszonylag gyorsan elavul...

Így nemrég az ***Összefogás a szabad szoftverek elterjesztéséért alapítvány (Free Software Network; FSN)*** is a géppark lecserélése mellett döntött. Nonprofit szervezet lévén ehhez a szabad szoftverek iránt elkötelezett emberek segítségét kérte. Papíron megterveztek egy olyan rendszert, amely képes kiszolgálni a magyar felhasználókat, az árát 1000 egyenlő részre osztották, és ezeket a „téglákat" igyekeznek „eladni". Az interjú idején a „téglaszámláló" 231-en állt... (➲ http://www.fsn.hu/?f=brick&setlang=hu).
A projektről és az FSN céljairól ***Nagy Attilát***, az alapítvány elnökét kérdeztük.

Tulajdonképpen miért kell segíteni a szabad szoftverek elterjedését? Az ember azt gondolná, hogy ha valamire ingyen van, azt az emberek maguktól is megtalálják. Szóval eredetileg honnan jött az ötlet, és pontosan kik állnak az FSN mögött?
A szabad szoftverek mögött egy igen erős meggyőződés, ideológia áll. A szabad szoftverek szerelmeseinek célja, hogy azt a tudást, amelyeket ezek a programok képviselnek, mindenki korlátok nélkül felhasználhassa a saját céljai elérésére. Mivel ezeket leggyakrabban non profit módon fejlesztik, a fizikai és az ismeretterjesztés a közösség feladata marad.
Az ***FSN*** mögött (***Free Software Network***, amelyet az ***Összefogás a szabad szoftverek elterjesztéséért alapítvány*** az internetes megjelenésénél használ) hat, szabad szoftvereket régóta szerető, használó ember áll: ***Nagy Attila (bra), elnök, Riskó Gergely (errge), titkár, Beregszászi Alex (alex), Micskó Gábor (trey), Pásztor György (gyu),*** és ***Petrikovics Balázs (gaab).***
Az alapítványunk elsősorban nem a megtalálásban segít, hanem az információhoz való gyors hozzáférésben, amelynek két prominens képviselője a ***hup.hu*** portál és az ***ftp.fsn.hu*** fájlszerver.

Miért jobb, ha Magyarországon is van egy olyan kiszolgáló, amin megtalálhatjuk a szabad szoftverek legfrissebb változatát? Miért nem használjuk egyszer?en az olyan nemzetközi helyeket, mint a sourceforge.net, vagy a freshmeat.org?
A ***Sourceforge*** és a ***Freshmeat*** funkciója némileg eltér az ***FSN*** által biztosított szolgáltatásokéitól. Az első egy fejlesztői webhely, míg a második a szabad szoftverek legújabb verzióira hívja fel a figyelmet. Annak, hogy Magyarországon is elérhetőek ezek a szoftverek, rengeteg előnye van. Egyrészt szinte mindenki által ugyanolyan gyorsan hozzáférhetőek, hiszen nem korlátozza a letöltést az egyes internet-szolgáltatók nemzetközi sávszélessége, illetve a tükrözéseken kívül saját tartalmat is igyekszünk biztosítani (beleértve a hazai projektek megjelenését is).

Hétköznapi igényekhez mérten igen impozáns teljesítménnyel rendelkező gépet akartok beszerezni. Nemzetközi viszonylatban, vagyis a többi szabad szoftverekkel kapcsolatos helyhez képest mennyire számít nagynak ez a rendszer?
Az ***FSN*** forgalma a szerverek elhelyezését biztosító ***Axelero Rt***-nél is meghatározó, azt külön monitorozzák és egyes esetekben szabályozzák. Ez azt hiszem jól mutatja, hogy a forgalmunk mind hazai, mind nemzetközi viszonylatban is jelentős. Mindezt úgy érjük el, hogy a szervereink nagy többsége 450 MHz-es ***Pentium III*** kategóriájú, amely azt hiszem ma már meglehetősen kicsinek számít. Ezen szeretnénk javítani azzal, hogy a terhelésnek megfelelően bővítjük az erőforrásokat. Az új gép nemzetközi mércével mérve közepesnek mondható. Külföldön általában több gépből álló szerverfarmot használnak, nem ritka, hogy 8-10 TB-os háttértárral és több gigabites hálózati kapcsolattal.

2005. január 3-án az 1000-ből még csak 196 tégla jött össze, az akció pedig február végéig tart. Bíztok benne, hogy meglesz a szükséges összeg? Vannak kiegészítő források?
Szinte minden lehetőségünket felhasználtuk már. Logókat kaptunk a közösségtől, amelyeket kitehettünk pár szakmai és nem szakmai oldalra is. A projekt szerepelt a ***Fix TV*** műsorán és a ***Fiksz rádióban*** is. Sokan felajánlották, hogy segítenek a kedvezményes beszerzésben. Sajnos azonban az eszközök beszerzéséhez pénzre van szükség, amely – egyelőre úgy tűnik – nem fog összejönni az eredeti határidőre, amit ennek megfelelően szeretnénk legalább egy hónappal kitolni. Valószínűleg így sem fog összejönni a teljes összeg, de talán egy picivel közelebb kerülünk hozzá. Kiegészítő források sajnos nincsenek. A gyártók és forgalmazók néhány eszközt tudnának olcsóbban adni, de csak ha valamennyi pénz (legalább 60-70%) összejön. Az adománygyűjtés után felkeressük őket, és meglátjuk, ki mit tud ennyiért adni.

A Linuxvilág stábja

Mi újság a rendszermag fejlesztése körül?

Markus Lidelt nevezték ki az *Intelligent Input/Output (I2O)* karbantartójának. Az *I2O Special Interest Group* által tervezett *I2O* olyan alkatrész szabvány, amely lehetővé teszi, hogy a kibemenet terhelés alól felszabadítsuk a *CPU*-t és ezzel megnöveljük az alkatrész I/O teljesítményét, miközben a futó rendszerre gyakorolt hatását is csökkentjük. *Markus* viszonylag új ember a *Linux* rendszermag fejlesztésben, 2004 márciusában jelent meg először rendszermag változásnaplóiban néhány kisebb *I2O* folttal a 2.6.1-es rendszermaghoz. Itt is, mint annyi más projektnél ez idő tájt de facto *Alan Cox* volt karbantartó és meglehetősen sok foltot fogadott el *Markustól* a következő néhány hónapban. 2004 júliusára a 2.6.8-as megjelenésekor viszont már *Markus* fogadta el az *I2O* csomagokat más fejlesztőktől. 2004 augusztusában, *Warren Togami Markust* jelölte hivatalos karbantartónak, így *Andrew Morton* jóváhagyása után, *Markus* 2004 szeptemberében frissítette a karbantartók fájlját és bejegyezte magát *I2O* karbantartóként. Nem sokkal ezt követően kezdeményezte az *I2O* kód újraírását/újraszervezését amelyet *Andrew* új, stabil rendszermagokban alkalmazható nagyobb változtatásokról szóló szabadabb rendszabályaival összhangban a *2.6.9-rc2* rendszermagba be is vettek.

A rendszermagfejlesztés gyakran teli van vitákkal és problémákkal. Mostanában a *Philips* webkamerákat támogató *pwc* meghajtó robbantott ki vitát a meghajtó karbantartójaként ismert *Nemosoft Unv*, és a *Linux Journal* rovatvezető és kernelkapitány *Greg Kroah-Hartman* között. A különféle alkatrészek támogatása érdekében a rendszermag szabályaival ellentétes módon régóta létezett egy hurok amellyel bináris modulokat lehetett a rendszermagba csatolni. Míg *Greg* sürgette az eltávolítását, *Nemosoft* pedig arra kérte *Linus*-t, hogy az egész meghajtót távolítsa el a rendszermagból. Most, hogy a meghajtó már a *GPL* oltalma alá esik, a szerzőnek nincs jogi alapja ilyesmit követelni, *Linus Torvalds* azonban úgy érezte teljesítenie kell a szerző kívánságát, különösen figyelembe véve, hogy a kérdéses kód karbantartó nélkül marad. Mint kiderült, ez nem volt éppen népszerű döntés néhány ember, például *Alan Cox*, szemében, akik kicsit konzervatívabban állnak hozzá a szabadalmi kérdésekhez. A *Nemosoft* kiadta a kódot, tehát nem veheti csak úgy vissza. Állítása igazolására *Alan* kijelentette, hogy ő is megkérhetné *Linus*-t, hogy távolítsák el az évek során összegyűlt összes *Alan* munkát, ami lényegében a rendszermag elég tetemes részét tenné ki és tönkretenné a teljes projektet. *Alan* álláspontja szerint ilyen esetekben egyszerűen nincs értelme teljesíteni a szerzők kérését. Hosszas vita után, *Luc Saillard* visszafejtette a kérdéses bináris modult és elküldte a *pwc* meghajtó új, kérdéses hurok nélküli változatát.

David Engebretsen úgy döntött tovább már nem vállalja a *PowerPC* karbantartói feladatait, így helyét *Paul Mackerras* és *Anton Blanchard* foglalta el, akik *Benjamin Herrenschmidt*, *Tom Rini* és még sok más szerzővel együtt rengeteg *PPC* foltot küldtek az évek során. *David* az *IBM*-nél munkája részeként tartotta karban a PPC változatot és az eredeti PPC64 Linux változat átírását végző csapatot vezette.

Jeff Garzik elkészítette a *blktool* eszközt, az eddig használt *hdparm* kezelhetőbb,általánosabb változatát . A *hdparm*-hoz hasonlóan, a *blktool* is pusztulást hozhat a merevlemezünkre ha helytelenül használjuk. Akárcsak a *hdparm*, a *blktool* is eléggé *IDE*-centrikus, bár *Jeff* a *SCSI*, *I2O* és *RAID* támogatáson is dolgozik. A *hdparm IDE*-centrikussága valószínűleg abból ered, hogy annak idején *Mark Lord*, az eredeti *IDE* alrendszer karbantartója készítette. A *hdparm* programhoz hasonló módon a *blktool* parancssoros felületen keresztül teszi elérhetővé a merevlemezünk apró részleteit, és segítségével a felhasználó finom módosításokat végezhet működés terén ami jelentős sebességnövekedést is okozhat. A *blktool* parancssora által használt pontos csatolófelület még képlékeny; kiváltképp miután *Alan Cox* úgy érzi, hogy amennyiben egy új eszközt fejlesztünk ki, annak javítania kell a *hdparm* eszközben felmerült hibákat különös tekintettel a parancssoros felületre. *Jeff* úgy tűnik kapható a különféle alternatívákra, így a végső eszköz valószínűleg többféle módszert is nyújt majd a felhasználóknak céljaik eléréséhez. A lényeg, hogy *Jeff* kezdeményezése támogatásra talált a kernelfejlesztők közt, és így a népek végül helyére pofozzák.

Zack Brown

Linux Journal 2005. január, 129. szám

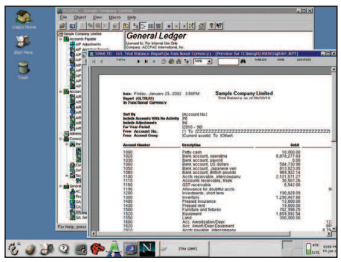
Új termékek

RTLinuxPro 2.1

A *FSMLabs* megjelentette az *RTLinuxPro* kettős rendszermagú, valós idejű operációs rendszer 2.1-es változatát. Az *RTLinuxPro 2.1* újdonsága többek közt a társ operációs rendszer továbbfejlesztett névtér-particionálása, amely a magasabb fokú megbízhatóságot és a hibakeresés és -elhárítás megkönnyítését szolgálja; a jobb *VME*-támogatás az ipari és a légi, valamint védelmi alkalmazások számára; az új processzortípusok kiterjesztett támogatása; valamint a kibővített *POSIX API*-k, amelyek a kód hordozását, a szabványos folyamatok, szálak és készülékek közötti adatcserét segítik elő. A beágyazott, több géptípust is felölelő fejlesztésekhez készült *RTLinuxPro Development Kit* az RTLinuxPro 2.1 kiadásával szintén frissült, újdonsága a *Visual SlickEdit IDE*.
www.fsmlabs.com

ACCPAC Advantage Series 5.3-as változat

Az *ACCPAC Advantage Series* számlázórendszer 5.3-as változata új tranzakcióelemző szolgáltatással bővült, segítségével a felhasználók



olyan kódokkal láthatják el a tranzakciókat, amelyek a tranzakciók teljes ideje alatt változatlanul maradnak. A címkék alapján mód nyílik jelentések készítésére, valamint a hagyományos pénzügyi jelentéseken túl részletesebb elemzéseket is lehet készíteni. Az 5.3-as kiadásba tucatnyi egyéb bővítés is került, többek közt átdolgozták a számlázó modulokkal kapcsolatos biztonsági szolgáltatásokat; az *ACCPAC* webalkalmazáson keresztül lehetővé vált a makrók létrehozása és futtatása; a beépített, az új rendelési modullal együttműködő készletkezelő révén pedig felgyorsult a feldolgozás és a szállítások teljesítése.
www.accpac.com

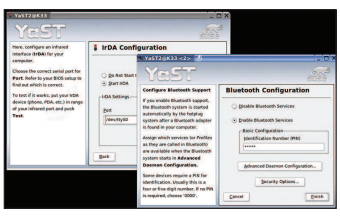
SuSE Linux Professional 9.2

A *SuSE Linux Professional 9.2* 32 bites x86 processzorokra és *AMD* és *Intel 64* bites lapkákra érhető el. A *SuSE 9.2* támogatja a *Bluetooth* vezeték nélküli kapcsolatokat; a *Bluetooth*-képes készülékeket

a *YaST* közreműködésével önműködően ismeri fel. A *Bluetooth*-kezelő program segítségével a felhasználók könnyedén csatlakozhatnak a *WLAN*-okhoz és az egyéb hálózatokhoz, illetve gond nélkül mozoghatnak közöttük. A *SuSE 9.2*-nek része a *KDE 3.3* és a *GNOME 2.6* asztali környezet; az *OpenOffice.org 1.1.3*; az *Evolution 2.0*; a *The GIMP 2.0*; az *Inkscape* vektorgrafikai és az *Nvu* weblapkészítő alkalmazás. Szintén a terjesztés része, ám zárt alkalmazás a *TextMaker* és a *PlanMaker*, az *SEP* biztonsági mentő program, valamint a *MainActor 5* videoszerkesztő bemutató változata.
www.suse.com/us

Yellow Dog Linux 4.0

A *Terra Soft Solutions* bejelentette a *Yellow Dog Linux (YDL) 4.0* változatát, mely 32 bites támogatást nyújt az *USB-G3, G4* és *G5 Power Mac* gépekhez, továbbá a *Freescale Board Support Package* révén a *Genesi Pegasos II ATX* alaplapok támogatása is megoldódott.



A *YDL 4.0 Fedora Core 2* alapú, *KDE 3.3* és *GNOME 2.6.0* környezetet tartalmaz; telepítője és maga a munkakörnyezet egyaránt megújult megjelenést kapott. A *YDL 4.0* terjesztésbe beépített alkalmazások között például az *OpenOffice 1.1.1*, a *Rhythmbox 0.8.3* és a *Mozilla 1.7* található meg, míg a fejlesztők munkáját a 32 bites, 2.6.8-as rendszermagra épült glibc 2.3.3 és *GCC 3.3.3* segíti. Az *YDL 4.0* fontos újdonsága a külső eszközök kiterjesztett *USB*-támogatása és a beépített *FireWire*-támogatás; utóbbival – kézi beállításokkal – akár a *FireWire* alapú rendszerindítás is megoldható. A *PowerBook* és a *G4/G5 Power Mac* gépeken a kétmonitoros használat is lehetővé vált. A vezeték nélküli kapcsolatok felépítése *Netgear WG511 54 Mb PCMCIA* kártyán keresztül lehetséges, a *D-Link* és *Linksys USB*-s kártyák támogatásának megvalósítását a közeljövőre tervezik.
www.terrasoftsolutions.com

Sugar Sales 2.0

A *LAMP* alaprendszerre épülő *Sugar Sales* és *Sugar Sales Professional 2.0* nyílt forrású fogyasztói kapcsolatkezelő (*CRM*) alkalmazások a *SugarCRM, Inc.* újdonságai. A *Sugar Sales Professional 2.0* új szolgáltatása az ajánlatkészítő és az árlista modul, melynek segítségével az értékesítő hozzáférhet a termékárakhoz, illetve módosíthatja azokat, továbbá testreszabott ajánlatokat készíthet és menthet ki *PDF* formátumba. Mindkét *CRM* alkalmazás utasításkövető képességekkel is rendelkezik, amelyek révén a felhasználók figyelemmel követhetik a marketingprogramokból, a külső kommunikációból és a felhasználói igényekből származtatott elvárásokat. Mindkét *CRM* alkalmazás webes naptárat, az értékesítési feladatokról szóló önműködő értesítéseket, testreszabható honlapokat és több előre megadott és dinamikus jelentést kínál.
www.sugarcrm.com

Ők mondták

Szóval, ha egy nálad okosabbat találsz, csak haladj tovább. A karbantartói felelősséged nagyjából olyasmire változik mint „Jó ötletnek hangzik – próbáljuk ki”, vagy „Ez jól hangzik, de mi a helyzet xxx-el?” A második verzió különösen hasznos, hiszen jó alkalom arra, hogy valami újat megtudjunk az „xxx”-ről, vagy kihúzhatjuk magunkat, hiszen rámutattunk valamire, amire az okosabb ember nem is gondolt. Mi mindkét esetben nyerünk.

Linus Torvalds

A rendszermag karbantartásról (*lwn.net/Articles/105375*)

Ha végiggondoljuk, valóban van értelme. A *Linux* és a nyílt forrású termékek olcsóbbak, ellenállóbbak és biztonságosabbak. Amikor a *Microsoft* azt állítja az ő termékük *TCO* (teljes bekerülési költség) értéke alacsonyabb, olyan mintha azt állítaná, hogy a *Föld* lapos. A helyesen gondolkodó informatikai igazgatók jól tudják, hogy a *Linux* és a nyílt forrású termékek olcsóbbak és aligha lehet bolonddá tenni őket szavakkal való bűvészkedéssel vagy gyanús, kereskedő-módosította *TCO* tanulmányokkal.

Del Elson

Open Source in Australia, CXO Today (*www.cxotoday.com*)

Ha a nyílt forrás megfelelő módon ellenőrzött nincs nála jobb. De a kód elérhetővé tétele önmagában még nem jelent garanciát.

Bruce Schneier

(*www.schneier.com/blog/archives/2004/10/schneier_securi.html*)

A jelenlegi tudományos kiadói modellek gazdaságilag nem tarthatók. A kutatók és a tanulók csak a szükséges tudományos műveltség törtrészéhez jutnak hozzá. Azonban már folynak a fejlesztések és tesztek a gyógymód és alternatív megoldások után.

University of California Office of Scholarly Communications

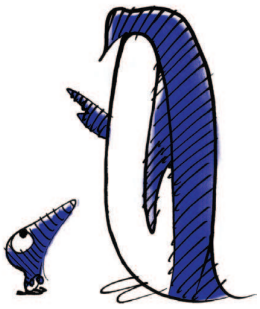
(*osc.universityofcalifornia.edu*)

Linux-index

1. Ennyi millió dollárt költöttek *Oklahoma* város új, nyilvános kulcsú biztonsági rendszert használó *Wi-Fi* rendszerére: 90
2. Az új rendszer által lefedett négyzetmérföldek száma: 650
3. A rendszer által a közönség számára biztosított hálózati elérhetőség mértéke: 0
4. Az *UTOPIA* azaz *Utah Telecommunications Open Infrastructure Agency* rendszeréhez tartozó városok száma: 14
5. Ennyi ezer háztartás tartozik az *UTOPIA* vonalhoz: 140
6. Az *UTOPIA* ennyi százaléka száloptika alapú: 100
7. Ennyit lehet megtakarítani *Linux* használatával a *Microsoft Windows* rendszerhez képest: 30
8. Egy 2000 alkalmazottat foglalkoztató nagyvállalat megtakarítása három év alatt az irodai alkalmazásokon, ezer Euroban: 525
9. Három év alatti megtakarítás 2000 alkalmazottat foglalkoztató nagyvállalat kiszolgálóin, ezer Euroban: 57
10. 2000 alkalmazottat foglalkoztató nagyvállalat megtakarítása a tartalomkezelésen három év alatt, ezer Euroban: 32
11. 2000 alkalmazottat foglalkoztató nagyvállalat megtakarítása az adatbázis-kezelőkön három év alatt, ezer Euroban: 21
12. A fenti tanulmányban vizsgált vállalatok száma: 50
13. Az új *Cray XD1 Opteron/Linux*-alapú szuperszámítógép legnagyobb gigaflop teljesítménye 12-processzoros vázösszeállításban: 58
14. Az új *Cray* legnagyobb processzorszáma, egy szekrénybe helyezett 12 vázzal: 144
15. A teljes gépszekrény csúcsteljesítménye gigaflopban: 691
16. Az új *Cray* ára millió dollárban: 2
17. Beágyazott *Linux* megoldásokhoz szánt lapkákon található *Linux*-kész *MIPS*-alapú *Sha hu (kis tigris)* rendszer sebessége Mhz-ben: 400
18. A *Sha hu* átlagos fogyasztása wattban: 1
19. *Kína* ennyi millió tanulót szeretne az informatikai iparba vonni: 200
20. *Kína* tanulónként ennyi számítógépet szeretne biztosítani tanulónként: 1

Források

1-3: *CLS Communications, over Business Wire (CLS Kapcsolattartás, üzleti vonalon)*
3-6: *UTOPIA*
7-11: *Research & Markets (Kutatás és Piac)*
12-16: *InternetNews*
17-20: *LinuxDevices*



A Linux Journal honlapján számtalan gond megoldásához találhattok további segítséget. A Sunsite tükröoldalait, a gyakori kérdéseket és az egyéb útmutatásokat a ➲www.linuxjournal.com honlapon olvashatjátok el. A rovatban közzétett válaszokat Linux-szakértők kis csapata készítette el. További kérdéseiteket szívesen fogadják (angol nyelven) a ➲www.linuxjournal.com/lj-issues/techsup.html címen, ahol csak egy kérdőívet kell kitöltenetek, de a bts@ssc.com címre levelet is írhattok. A levél tárgyában szerepeljen a „BTS" kulcsszó.

A hónap szakmai tanácsai

A /etc/shadow terjesztése

Egy segédprogramra vadászok, melynek érzésem szerint már léteznie kell valahol. Lássuk, mi a gondom. A kormányzati számítógépeken mostanában gyakrabban kell módosítani a jelszavakat, ide értve a root jelszavát is. Eddig olyan sokféle géppel és operációs rendszerrel rendelkeztünk, hogy a */etc/shadow* vagy a */etc/passwd SSH*-n keresztüli átmásolása az összes gépre gyakorlatilag kivitelezhetetlennek tűnt, ugyanis minden operációs rendszer alatt más bejegyzések tartoznak a roothoz. Ha egy gépen felülírjuk a root bejegyzését egy másik operációs rendszer írásmódja szerint, azzal csak kárt okozunk. Feltételezem, hogy vakon felülírva a fájlokat elsősorban a root héját és bejelentkezési útvonalát lehetne elrontani. Az utóbbi időben ugyanakkor sikerült megszabadulnunk a nem PC alapú – *SGI*, *Sun*, *HP* és egyéb – munkaállomásainktól, így remélhetőleg csak egyetlen operációs rendszerrel kell majd bajlódnunk, amikor megküzdünk a vírusok és a foltok problémájával. Vagyis végre lehetővé válik, hogy a root bejegyzést az egyes gépek */etc/shadow* vagy */etc/passwd* fájljába *SSH*-n keresztül írjuk bele.

Vajon létezik már erre eszköz? Gondolom, csak készült már olyan parancsfájl, amit némi átalakítással alkalmassá tudnék tenni a módosítások terjesztésére. A gépek egy része valamilyen – *DNS*-kiszolgálóval is ellátott – tartományba tartozik. A *DNS*-t nem futtatók *NIS* szolgáltatást használnak. A *DNS* alapúakkal nem nagyon foglalkoztam még, de azt tudom, hogy a *NIS*-t futtatóknál még mindig helyileg kell megváltoztatni a root bejegyzését. Eddig telneten vagy *SSH*-n keresztül egyenként beléptünk minden gépre, így adtuk meg a root új jelszavát – nem is nagyon volt más lehetőségünk. A gépeken naprakészen kell tartani a root fiókokat, különösen, ha valamiért esetleg egyfelhasználós módba akarnánk váltani.

Irene Paradis
➲ *irene.paradis@us.army.mil*

A *NIS* használata klasszikus megoldás erre a gondra. A root jelszava ugyanakkor egyetlen módszerrel kezelhető igazán jól, mégpedig a */etc/shadow* fájl módosításával. *RADIUS* kiszolgáló használatát is érdemes megfontolni.

Christopher Wingert
➲ *cwingert@qualcomm.com*

Az `rdist` alkalmas egy-egy fájl több gépre való másolására. Lásd: ➲ *www.magnicomp.com/rdist* Az is megoldás, hogy letiltod, „kicsillagozod" a root jelszavát, vagyis * karaktereket írsz a s/etc/shadow fájl titkosított jelszót tartalmazó mezőjébe, majd mindenhez `sudo`-t használsz.

Don Marti
➲ *dmarti@ssc.com*

Hogyan lehet kapcsolókat megadni a rendszermagnak?

A 2004 novemberi számban megjelent tanácsok között a *Fedora* telepítésével kapcsolatosan olvasható `linux noacpi, linux disableapic` és `linux noacpi disableapic` kapcsolók pontosan milyen célt szolgálnak? Két *AMD MP 2800+* processzort tartalmazó gépem rendszeresen összeomlik. Az utolsó memóriakiíratás során láttam valami *acpi* vagy *apic* vonatkozású dolgot – a legközelebb fel kellene még jegyeznem, hogy pontosan mi is jelenik meg. A cikk olvasása után ki akartam próbálni ezeket a kapcsolókat, de nem sikerült rájönnöm, hol találom őket.

Doug Baker
➲ *cfdbaker@qwest.net*

A tanács az volt, hogy a `noacpi` vagy a `ldisableapic` vagy a `noacpi disableapic` kapcsolót parancssori kapcsolóként adjuk át a rendszermagnak. Amikor a rendszertöltő, vagyis a *GRUB* vagy a *LILO* rákérdez, hogy melyik operációs rendszert vagy rendszermagot akarod elindítani, módod nyílik a kapcsolók hozzáadására. *LILO* használatakor nyomd meg a *Ctrl-X* billentyűkombinációt, ezzel a parancssorba lépsz, majd írd be a `linux noacp` parancsot. Feltételezem, hogy a *Linux* a *LILO* menüpontjainak egyike. Ha megoldást hoz a kapcsoló használata, akkor állandó jelleggel is hozzáadhatod a */etc/lilo.conf* fájlhoz.

Usman Ansari
➲ *usmansansari@yahoo.com*

A folyamat a *GRUB* rendszertöltő esetében – amely a *Fedora* alapértelmezett rendszertöltője – is nagyon hasonló. Lásd a nem hivatalos Fedora GYK-t: www.fedorafaq.org/#otherinstall.

Don Marti
➲ *dmarti@ssc.com*

A processzor tesztelése különböző terhelésekkel

Munkámból fakadóan gyakran tesztelek linuxos gépeket. Olyan módszert keresek, amellyel fo-

kozatosan lehet emelni a processzor terhelését nulláról száz százalékra, miközben követni lehet, hogy mi történik az egyes alkalmazásokkal. Ha megpróbálom fokozatosan növelni a processzorterhelést, az általában hirtelen ugrik nulláról száz százalékra. Ha rövid időkre megpróbálom felfüggeszteni a futtatást, akkor nulla és száz százalék között ugrál az igénybevétel mértéke. Tudtak valamilyen eszközt erre a célra?

Patrick Killelea
➲ *p@patrick.net*

Próbálj meg olyan programot futtatni, amely valamilyen a processzort erősen terhelő feladatot, például véletlen számok előállítását usleep hívásokkal kever. A `BUFSIZE` és az `USLEEP` értékét módosítva nekem sikerült különféle processzorterheléseket elérnem:

```
/* A fordítás a 'gcc -Wall terheles.c -o
terheles' paranccsal történik */
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#define BUFSIZE 1024
#define USLEEP 10000
char buf[BUFSIZE];
int main (int argc, char **argv)
{
        int f;
        f = open("/dev/urandom",
        ➥O_RDONLY);
        while (1) {
                read(f, &buf, BUFSIZE);
                usleep(USLEEP);
        }
        return 0;
}
```

Köszönöm *Greg Kroah-Hartmannek*, hogy letisztázta a fenti kódot. Lásd még: `man usleep`. Ha többprocesszoros gépen egy-egy processzort akarsz terhelni, akkor tekintsd át a *Robert Love* a *Linuxvilág 2003 szeptemberi* számában megjelent, *„Processzorhoz kötés”* című cikkében szereplő rendszerhívásokat.

Don Marti
➲ *dmarti@ssc.com*

Egyfelhasználós mód

Rendszertöltés közben hogyan válthatok egyfelhasználós, azaz első szintű futtatási módba?

Arthur Schroeder, showmeyr@yahoo.com

Írd át a *GRUB* rendszertöltő sorát, és add hozzá a `single` parancsot.

Christopher Wingert
➲ *cwingert@qualcomm.com*

A *LILO* és a *GRUB* parancssorába egyaránt be lehet írni a `single` parancsot, ha egy linuxos gépet egyfelhasználós módban akarsz elindítani. Ha valamiért mindig egyfelhasználós módban akarod indítani a gépet, akkor a *LILO* vagy a *GRUB* beállításainak módosításával is át tudod adni a `single` kapcsolót a rendszermagnak. A */etc/inittab* fájl átírása is megoldás. Saját *Red Hat 9.0* terjesztést futtató gépemen a fájl elején található az `id:3:initdefault` parancs, amelyben a 3-as számot kell 1-esre cserélni.

Usman Ansari
➲ *usmansansari@yahoo.com*

Lukács 5:37-38

Új *Dell Dimension 4600* gépemre próbálom telepíteni a *Red Hat Linux 7.1*-es változatát. A telepítő CD elindul, majd kiválaszthatom a telepítés típusát. Mindegy, mit választok, miután a számítógép megkezdte a hardverelemek felismerését, és túljutott a CD-ROM-meghajtón és a merevlemezeken, egyszerűen lefagy. A kikapcsoláson kívül semmit nem tudok kezdeni vele.

Joe Pietro
➲ *jm_pietro@hotmail.com*

Mielőtt túl sok idődet fecsérelnéd el, javaslom, hogy próbálkozz meg egy újabb terjesztéssel. A *Red Hat 7.1* már jó pár éves. Nagy valószínűséggel egy újabb terjesztéssel több szerencséd lesz. Én a *Fedora Core 2* terjesztést ajánlom. A *Fedora Core* a *Red Hat* leszármazottja, és hagyományosan jól fut a *Dell* gépeken. A ➲ *www.redhat.com* címről tudod letölteni.

Usman Ansari
➲ *usmansansari@yahoo.com*

A *Red Hat 7.1* terjesztéshez már nincs aktív forrás, és biztonsági frissítések sem készülnek hozzá. Lehet, hogy a géped megérezte a biztonsági hiányosságokat? A régebbi *Red Hat Linuxokhoz* a ➲ *edoralegacy.org* oldalon találhatsz támogatást. Ha rövid idő alatt ellenőrizni szeretnéd, vajon a géped megfele-

lően működik-e, valamint használható-e *Linux* alatt is, akkor először próbáld ki a CD-lemezről indítható, a ➲ *knoppix.org* oldalról elérhető *Knoppix* terjesztést.
Don Marti
➲ *dmarti@ssc.com*

USB-soros csatolók soros kapuinak beállítása

Van egy alkalmazásom, mely *USB*-soros átalakítókon keresztül távoli soros készülékekhez csatlakozik. Van mód arra, hogy az egyes *USB* eszközök meghatározott *USB* soros kapuként kerül jenek számbavételre, függetlenül az *USB* kapuk csatlakoztatásának sorrendjétől? Azt kellene például elérni, hogy az *x USB* kapu mindig a */dev/usb/ttyUSBy* legyen. Mivel az alkalmazást több mint 200 helyen fogják használni, és előfordulhat, hogy egy-egy USB-soros csatolót másikra, esetleg újabbra ki kell cserélni, az *USB* eszközök sorozatszámára alapuló megoldás nem az igazi.
Jeff Dennison

Ha 2.6-os rendszermagot használsz, az udev segítségével tudsz ilyen párosításokat létrehozni. Az egyes *USB*-soros egységeknél kell keresned valamilyen egyedi azonosítót, ezek alapján létre tudod hozni az eszközök elnevezését megadó szabályokat. Amint írod, a sorozatszámok használata nem a legjobb megoldás. Próbálkozz az *USB* eszköz topológiájának követésével, vagy bármi más egyedi dologgal, hiszen az egyediség a dolog kulcsa. Ha 2.4-es rendszermagot használsz, akkor csak sok szerencsét kívánhatok. A */proc/tty/drivers/usb-serial* könyvtárt bebarangolva megpróbálhatod kideríteni, hogy melyik készülék melyik */dev/ttyUSB* csomóponthoz csatlakozik, de nem lesz könnyű dolgod. Legalább találtál még egy nyomós indokot, amiért érdemes lenne áttérned a 2.6-os rendszermagsorozatra.
Greg Kroah-Hartman
➲ *greg@kroah.com*

Fordítási kapcsolók megadása Gentoo alatt

Kezdőként, *stage3 .tar* fájl használatával, élő CD-lemezről próbálom telepíteni a *Gentoo* terjesztést. Sikerült elindítanom ezt a fázist, és próbálom optimalizálni a terjesztést. Gondolom, hogy a *GCC make* esetében különféle kapcsolókat kellene megadnom. Egyelőre csak a kezdéshez, illetve az alapok megértéséhez várok segítséget. Mit tanácsoltok?
Rebelrouser, ➲ *Rebelrouser@blueyonder.co.uk*

Ha nem tudod, mit változtass meg, inkább maradj az élő CD-lemezen szereplő beállításoknál. Ezek a */etc/make.conf* fájlban találhatók. Erről és a *Gentoo* helyes telepítéséről a terjesztés telepítési útmutatójában találsz további tájékoztatást.
Greg Kroah-Hartman, ➲ *greg@kroah.com*

A Fedora telepítője lefagy

Fedora terjesztést próbálok telepíteni. Ennek során eljutok a képernyő beállításához, ahol 256 színt választok, majd az *OK* gombra kattintok. Ebben a pillanatban a képernyő megmerevedik, olvashatatlanná válik. A parancssort sem tudom elérni. Kérlek, segítsetek!
Chris, ➲ *fiston63@hotmail.com*

A grafikus kártya és az *X* telepítéskori beállítását ki is hagyhatod. Ha a telepítés befejeződött, indítsd újra a gépet, és próbálkozz meg most az *X* beállításával. A kártyád típusát az `lspci -vvv` paranccsal derítheted ki. Ha a kártya támogatása hiányzik, látogass el a gyártó weboldalára, és keress hozzá illesztőprogramot.
Usman Ansari
➲ *usmansansari@yahoo.com*

Be lehet csapni az Oracle telepítőjét?

Tudja valaki, hogyan lehetne átverni az *Oracle 10g* telepítőjét, és elhitetni vele, hogy egy *Slackware* rendszer *Red Hat*? Így rá lehetne venni, hogy legalább megpróbálja a telepítést. Ha nem, akkor vajon hogyan jön rá, hogy nem Red Hat rendszeren futtatják?
Blake Tullysmith
➲ *bdt@vipretech.com*

Futtasd az `strace` eszközt a telepítőn:

```
# strace oracle-installer
```

Így meg tudod állapítani, hogy a program mit vizsgál meg, mielőtt elutasítaná a telepítést.
Christopher Wingert
➲ *cwingert@qualcomm.com*

Linux Journal 2005. január, 129. szám

Változatkezelés az Arch használatával: bevezetés

Ha a CVS-t akarjuk felváltani, vagy komolyan gondolkozunk egy változatkezelő rendszer bevezetésén, íme egy hatékony eszköz, amely nyomon követi a változtatásokat és kézben tartja a projektjeinket.

Az ***Arch*** rohamosan válik az egyik leghatékonyabb eszközzé a szabad szoftvereket fejlesztő programozók eszköztárában. Ez a cikk annak a három részes cikksorozatnak az első tagja, amely bemutatja az ***Arch*** használatát az osztott fejlesztés során, valamint megismerteti az olvasót a megosztott archívumok és az ***Arch projektekhez*** alkalmazható önműködő parancsfájlok használatával.
Ebből a cikkből megtudhatjuk, hogyan szerezhetjük meg a kódot egy nyilvános Arch-archívumból, hogyan tölthetjük vissza a változásokat és hogyan készíthetünk helyi fejlesztési ágat a projektről kapcsolat nélküli használatra. Megismerhetünk továbbá néhány technikát a helyi és távoli archívumokkal való hatékonyabb munkára is.

A változatkezelés története

A változatkezelés egy projekt változásainak nyilvántartásával foglalkozik. A szabad szoftverek fejlesztése során alapvető fontosságú a projekten végzett munka elemzésének, a fejlesztési ágak összehasonlításának, a változtatások ismétlésének vagy visszavonásának lehetősége. A lehetséges közreműködők nagy száma és a változtatások gyors kibocsátása rövid időn belül életre hívta ezeknek a változtatásoknak a kezelésére szolgáló eszközöket.
A változáskezelés legősibb eszköze a szalagos tároló volt. Egy projekt korábbi változatait a szalagon tárolt biztonsági mentésekből kellett előhúzni és soronként összehasonlítani az új változattal. A szalagról történő beolvasása nem egy gyors folyamat, így ez semmilyen szempontból nem nevezhető hatékony megoldásnak.
Ennek a hátráltató tényezőnek a kiküszöbölésére sok fejlesztő kéznél tartotta a fájlok régebbi változatait is összehasonlítás céljából, és ez a lehetőség hamarosan a fejlesztőeszközökben is megjelent. A fájlalapú változatkezelés, amelyet például az ***Emacs*** karakteres szerkesztő is használ, számozott biztonsági másolatokat őriz a fájlokról, így ha meg akarjuk vizsgálni a változtatásokat, könnyedén összehasonlíthatjuk a *foo.c~7~* fájlt a *foo.c~8~* fájllal. Néhány korai kereskedelmi operációs rendszerben még a fájlrendszer szintjén is megjelentek ezek a számozott biztonsági másolatfájlok.
Közel két évtizeden keresztül a szabad programok külső fejlesztői által előszeretettel alkalmazott formátum a foltfájl volt, amit különbségfájlként is emlegettek. Két fájlt összehasonlítva egy erre a célra fejlesztett különbségképző program állította elő azt a listát, amely a két fájl eltéréseit emelte ki. A kimenetként előálló különbségfájl által tartalmazott eltérések jóváhagyásához csak egy foltozó programon kellett lefuttatni a fájlt.
A 90-es évektől a ***CVS (Concurrent Versions System)***, párhuzamos változatkezelő rendszer) vált a fejlesztők egy központi csoportja által alkalmazott alapértelmezett változáskezelő rendszerré. Egy egyszerű ágkövető és egyesítő rendszer teszi lehetővé a felhasználó számára a különböző fejlesztési ágakkal történő kísérletezést, majd a sikeres próbálkozások főprojektben történő rögzítését.
A ***CVS***-nek megvannak a maga korlátai, amelyek sok projekt számára egyre inkább terhet jelentenek. Először is a rendszer egyáltalán nem tárolja az olyan metaadatok megváltozását, mint a fájlokhoz kötődő jogosultságok vagy a fájl nevének a megváltozása. Ilyen hiányosság az is, hogy a bejegyzések nem kapcsolhatók össze, így igen nehézkes a több fájlon vagy könyvtáron átívelő változások követése. Végül majdnem minden távoli ***CVS***-táron végrehajtott művelet egy újabb kapcsolat megnyitását igényli a kiszolgáló felé, megnehezítve ezzel a kapcsolat nélküli munkát.
A ***Subversion Projekthez*** hasonló erőfeszítések nagy utat tettek meg a ***CVS***-ben fellelhető hiányok kiküszöbölése felé vezető úton. A ***Subversion*** lényegét tekintve egy ***CVS++***, amely támogatja a metaadatok változásának rögzítését és az elemi bejegyzéseket. Továbbra is szükség van azonban egy olyan központi kiszolgálóra a hálózaton, amelyhez az összes változáskezelésben részt vevő ügyfélgép csatlakozik.

A megosztott változáskezelő rendszerek

Az utóbbi néhány évben a változáskezelő rendszereknek egy új generációja látott napvilágot, amelyek mindegyike az osztott modellt alkalmazza. Az osztott változáskezelő rendszerek szakítanak a központosított tárhely elvével, ehelyett az egyenrangú gépek alkotta elrendezést részesítik előnyben. Minden fejlesztő fenntartja a maga tárolóhelyét, a rendelkezésre álló eszközök pedig lehetővé teszik a végrehajtott változtatások egyszerű átadását a hálózaton lévő gépek között.
A ***Monotone***, a ***DARCS*** az ***Arch*** és a hasonló projektek olyan környezetben tettek szert nagy népszerűségre, ahol a szabad forráskódú fejlesztés a jól összekapcsolt egyetemeken kívül zajlik, és a noteszgépek sokkal elterjedtebbek.

Jelen pillanatban az osztott rendszerek egyik legígéretesebb darabja a ***GNU Arch***. Az ***Arch*** úgy bonyolítja a kapcsolat nélküli használatot, hogy a felhasználókat helyi másolatok létrehozására ösztönzi és hatékony eszközöket kínál a projektek archívumok közti mozgatására. Az ***Arch***-ból hiányzik bármiféle kizárólagos kiszolgálói folyamat, az archívum kezelésére pedig a fájlrendszer-műveletek egy hordozható részhalmazát használja. Az archívumok egyszerűen olyan könyvtárak, amelyeket a hálózaton az általunk választott fájlrendszer-protokoll segítségével teszünk elérhetővé. Mi több, az ***Arch*** támogatja az archívumok ***HTTP***, ***FTP*** vagy ***SFTP*** protokollokon keresztül történő elérését. Az egyik nagy előnye annak, hogy nincs egy kitüntetett démon, hogy nem kap egy új program jogokat a kiszolgáló gépünkön. Így a biztonsággal kapcsolatos kérdések csak az ***SSH*** démonnal vagy a webkiszolgálóval lesznek kapcsolatban, amin a legtöbb rendszergazda amúgy is rajta tartja a szemét. Egy másik előny, hogy az ***Arch*** használatakor a legtöbb feladat megoldásához nincs szükség a root jogosultságaira. A fejlesztők elkezdhetik csak a saját gépeiken használni és archívumokat tehetnek közzé anélkül, hogy akár csak telepíteni kellene az ***Arch***-ot a webkiszolgáló gépen. Ez hatással van az alkalmazás módjára is: míg a ***CVS*** vagy a ***Subversion*** használata egy felülről lefelé irányuló döntés, amely a teljes projektcsapatra vonatkozik, az ***Arch***-ot alkalmazhatja csupán egy-két fejlesztő is a csapatban, míg át nem áll a többi tag is.

Hogyan juthatunk a tla-hoz?

Az ***Arch*** eredetileg ***Tom Lord hackerlab*** programkönyvtáraihoz tartozó burkoló- és héjprogram-gyűjtemény volt. Abban az időben a programot larch néven emlegették és egy kicsit esetlenül lehetett használni. Az ügyfél ***C*** nyelven teljesen át lett írva, a neve pedig ***tla*** lett ***(Tom Lord's Arch)***. A felülete még mindig nem tökéletes, de arra megfelel, hogy egy képzett fejlesztő normál módon használja a napi munkájához. A ***tla*** csomagjai a legtöbb ***GNU/Linux*** rendszercsomaghoz elérhetőek (források: ➲ ***www.linuxjournal.com/article/7752***).

Egy csak olvasható projekt letöltése

Miután feltelepítettük a ***tla***-t, nem árt, ha valamilyen letöltött kóddal ki is próbáljuk. Az ***Arch*** az adatainkat egy könyvtárban tárolja, amit archívumnak nevezünk. Egy archívumon belül az adataink egymásba ágyazott kategóriákban helyezkednek el: ezek a projektek (ez az egész munkának a neve), az ágak (a fejlesztés vagy valamilyen egyéb leíró elem egy bizonyos ága, és a változatok (egy egyszerű szám, amelyet annak a jelölésére használhatunk, hogy egy bizonyos szál fejlesztésében milyen messze sikerült előrehaladnunk). A kódhoz jutás első lépése egy nyilvános archívum regisztrálása, így az ***Arch*** egy nevet tud rendelni az archívum helyéhez:

```
$ tla register-archive http://www.lnx-bbc.org/arch
```

Ezt követően a `tla archives` parancs futtatása után látnunk kell az `lnx-bbc-devel@zork.net--gar` archívumot Ha kíváncsiak vagyunk, hogy itt milyen projektek tárolása történik, a teljes lista lekérésére használhatjuk a ***tla abrowse*** parancsot.

```
$ tla abrowse lnx-bbc-devel@zork.net--gar
lnx-bbc-devel@zork.net--gar
```

```
lnx-bbc
  lnx-bbc--research
    lnx-bbc--research--0.0
      base-0 .. patch-10

  lnx-bbc--stable
    lnx-bbc--stable--2.1
      base-0 .. patch-29

scripts
  scripts--gargoyle-bin
    scripts--gargoyle-bin--1.0
      base-0 .. patch-7
```

Ebből a listából az látható, hogy a `lnx-bbc-devel@zork.net--gar` archívum két projektet tartalmaz `lnx-bbc` és `scripts` néven. Az `lnx-bbc` két ággal rendelkezik: `research` (kísérleti) és `stable` (stabil). Az `lnx-bbc--research` ágnak csak egy változata van (0.0), amelynek tíz változtatását tartalmazza az archívum. Az `lnx-bbc--stable` egy változattal (2.1) és 29 változással szerepel.
Mivel most már rendelkezünk az `LNX-BBC` regisztrált archívumával a helyi listánkban, nincs akadálya annak, hogy letöltsük az `LNX-BBC` stabil ágának egy másolatát:

```
$ tla get \
lnx-bbc-devel@zork.net--gar/lnx-bbc--stable lnxbbc
```

Amikor befejeződött a letöltés és a foltok alkalmazása, kell lennie egy ***lnxbbc/*** könyvtárunknak tele mindenféle fájllal. Tegyünk úgy, mintha megváltoztatnánk a kódot: lépjünk be az ***lnxbbc/*** könyvtárba és írjunk be valahova egy új megjegyzést.

Változtatások hozzáadása

Miután megváltoztattuk a fájlt, a `tla what-changed` parancsnak ki kell írnia egy `M robots.txt` sort, jelezve, hogy a `robots.txt` tartalma megváltozott. A változás részleteit a `tla what-changed --diffs` paranccsal kérhetjük le, amely kilistáz egy különbségfájlt készen arra, hogy visszaküldjük a projekt fejlesztőcsapatának.

```
--- orig/robots.txt
+++ mod/robots.txt
@@ -1,3 +1,5 @@
+# Welcome, robots!
+
 User-agent: *
 Disallow: /garchive/
 Disallow: /cgi-bin/
```

Ennek a módszernek a hátránya, hogy a különbségfájl nem jelzi a metaadatokban bekövetkezett változásokat: az eltávolított és új fájlok nem fognak megjelenni, amikor egy másik fejlesztő lefuttatja a különbség-foltunkat. Ahhoz, hogy a projekt kezelőinek egy bonyolultabb változtatást át tudjunk adni, egy változáskészletet (changeset) kell létrehoznunk. Az ***Arch***-ban egy változáskészlet a teljes könyvtárszerkezetet jelenti a fájlok változásainak, a foltoknak, az új és eltávolított fájloknak a teljes nyilvántartásával. A közreműködés

legjobb módszere az, ha létrehozunk egy változáskészlet-könyvtárat és ezt összecsomagoljuk a továbbításhoz:

```
$ tla changes -o ,,new-robot-comment
$ tar czvf my-changes.tar.gz ,,new-robot-comment/
```

Az *Arch* nem veszi figyelembe a két veszővel, egyenlőségjellel és még néhány különleges karakterrel kezdődő fájlneveket. Azzal, hogy elhelyezzük a „ jelet a változáskészletünk könyvtárnevének az elején, elkerüljük, hogy az *Arch* kifogásolja, hogy az új könyvtárunk még nem létezik az archívumban. A gyakorlatban hasznos lehet az elektronikus levélcímünk vagy valamilyen más azonosítónk feltüntetése a tar-csomag fájlnevében és a változáskészlet könyvtárának a nevében.

Az archívum frissen tartása

Időnként szükségünk lesz a projekt legfrissebb változtatásainak letöltésére. Ez nem jelent bonyolultabb feladatot, mint a `tla update` parancs futtatását a letöltött másolatban. Az *Arch* először egy `tla undo` parancs végrehajtásával biztonságba helyezi a helyi változtatásainkat, mielőtt az új változáskészleteket alkalmazná. Miután feltette az összes foltot, lefuttatja a `tla redo` parancsot a helyi változások visszaállítására.
A korábban bemutatott minden `tla` parancshoz szükség van egy élő hálózati kapcsolatra az archívumot tároló `lnx-bbc.org` rendszerhez. A kapcsolat nélküli használathoz létre kell hoznunk egy helyi archívumot és ezen belül egy fejlesztési ágat.

Egy archívum létrehozása

Mielőtt dolgozni kezdenénk egy írható-olvasható archívumban, azonosítanunk kell magunkat a `tla` számára:

```
$ tla my-id "J. Random Hacker <jrh@zork.net>"
```

A levélcímünk megadása után elérkezett az idő, hogy archívumot készítsünk a projektjeink számára. Az *Arch* több archívum létrehozását is lehetővé teszi, de egy archívumon belül is tetszőleges számú projekt és fejlesztési ág tárolására van lehetőség.
Az archívumok neve két részből áll, amelyet kettős kötőjellel választunk el egymástól: az első rész az elektronikus levélcímünk, a második pedig valamilyen azonosító. Sok fejlesztő azonosítóként az évszámot adja meg és minden évben új archívumot kezd:

```
$ tla make-archive -l jrh@zork.net--2004 ~/ARCHIVE
$ tla my-default-archive jrh@zork.net--2004
```

A `my-default-archive` paranccsal megadva az alapértelmezett archívumunkat a helyi archívumon végrehajtandó műveletek begépelését könnyíthetjük meg.

Fejlesztési ág létrehozása

Az *Arch* arra ösztönzi a felhasználókat, hogy a fejlesztési ágak használatával ágaztassák el és fésüljék össze a projekteket. Az ágak jelentik az elsődleges módszert a kód egyik archívumból a másikba történő mozgatására akár hálózaton keresztül is.
A fejlesztési ágakat használhatjuk a projekt egy teljesen új fejlesztési irányát tartalmazó teljes kód változásának nyomon követésére, de alkalmazhatjuk arra is, hogy a projektünk egy átmeneti másolatát hozzuk létre a noteszgépünkön, amelyen hálózati kapcsolat nélküli környezetben is dolgozhatunk.
A nyilvánossá tett ágak jelentik a fejlesztéssel kapcsolatos párbeszéd elsődleges eszközét is az *Arch*-ot használó fejlesztők között. Ahelyett, hogy levélben küldözgetnének nagy méretű tar-csomagokat egymásnak, a közreműködő valószínűleg inkább egy ágat hoz létre a helyi változások tárolására, és arra kéri fel a fejlesztőket, hogy fésüljék bele ezeket a változtatásokat a fő projektbe. Itt ragyog fel leginkább az *Arch* decentralizált és demokratikus természete: bárki csatlakozhat a fejlesztéshez anélkül, hogy ehhez a központi fejlesztőcsapattól valamilyen különleges jogosultságot kellene kapnia.
Mielőtt elágaztatnánk az `lnx-bbc` projektet, helyet kell biztosítanunk a projekt számára az archívumunkban. A projekt azonosítója az archívum nevéhez hasonlóan épül fel: a kategória (vagy projektnév), két kötőjel, az ág neve, két kötőjel és a verziószám. Valószínűleg *Tom Lord LISP*-szakértői tapasztalata vezetett ezeknek a kötőjeleknek a használatához:

```
$ tla archive-setup lnx-bbc--robot-branch--0.0
```

Ezzel létrejön az `lnc-bbc` nevű kategória a `robot-branch` nevű fejlesztési ággal `0.0` verziószámmal. A `jrh@zork.net--2004/` archívum-nevet nem kell megadnunk a projekt neve előtt, mivel ez az alapértelmezett archívumunk.

A fejlesztési ágak címkézése

Végül elérkezett az idő arra, hogy felcímkézzük az águnkat. Ez annyit jelent, hogy a `robot-branch` egy címkével kezdődik, amely az `lnx-bbc-devel@zork.net--gar` archívumban lévő egy projekt adott változatára mutat, a helyi változtatások pedig ettől a ponttól kezdődnek:

```
$ tla tag \
  lnx-bbc-devel@zork.net--gar/lnx-bbc--stable--2.1 \
  lnx-bbc--robot-branch--0.0
```

Ha itt futtatjuk a `tla abrowse` parancsot, az archívumunkról a következő képet kell mutatnia:

```
jrh@zork.net--2004
  lnx-bbc
    lnx-bbc--robot-branch
      lnx-bbc--robot-branch--0.0
        base-0
```

Az új fejlesztési ág használatba vétele

Most már készen állunk arra, hogy letöltsük az új águnk egy másolatát:

```
$ tla get lnx-bbc--robot-branch robot-branch
```

Itt beléphetünk a `robot-branch` könyvtárba és módosíthatunk egy-két dolgot:

```
$ chmod 444 index.txt
$ tla mv faq.txt robofaq.txt
$ echo "ROBOT TIME" > robot-time
```

```
$ tla add robot-time
$ tla rm ports.txt
```

A `tla mv` parancs olyan módon módosítja egy fájl nevét, hogy az ***Arch*** képes legyen a fájl változásának nyomon követésére. Fontos, hogy ezt a parancsot használjuk a szabványos `mv` helyett. A `tla add` parancs előkészíti egy fájl hozzáadását az archívumhoz, a `tla rm` pedig egy fájl eltávolításának tervét rögzíti. Ezek a változások most már megjeleníthetők a helyi fejlesztési ágban:

```
$ tla commit
```

Elindul a kedvenc karakteres szerkesztőnk (a korábban a `$EDITOR` környezeti változóban megadottaknak megfelelően) egy végrehajtási jegyzőkönyv sablonját kínálva. Miután kitöltöttük a bejegyzéseket, a mentéssel és kilépéssel véglegesítjük a változtatásokat.
A `tla abrowse` parancsot kiadva láthatóvá válik, hogy most már két változata létezik az archívumunkban lévő `robot` ágnak, a ***base-0*** és a `patch-1`:

```
jrh@zork.net--2004
  lnx-bbc
    lnx-bbc--robot-branch
      lnx-bbc--robot-branch--0.0
        base-0 .. patch-1
```

Egy projekt összefésülése két különböző archívumból

Természetesen, mialatt a saját águnkon dolgozunk, az eredeti archívumban is folytatódhat a munka. A `tla update` parancs futtatása csak a helyi ágból olvassa ki a változtatásokat, az eredeti projektből nem. A központi archívumból a `star.merge` paranccsal tudjuk a változtatásokat átemelni:

```
$ tla star-merge \
lnx-bbc-devel@zork.net--gar/lnx-bbc--stable--2.1
```

Ütközések esetén (amikor a saját águnk és a központi projekt is tartalmaz változtatásokat ugyanazokra a kódsorokra vonatkozóan) az ***Arch*** az eredeti foltozási eljárást használja, amelynek során létrehozza a megfelelő `.orig` és `.rej` kiterjesztésű fájlokat. Érdemes egy keresést végrehajtani a visszautasításokat tartalmazó fájlokra, mielőtt a star-merge parancs változtatásait véglegesítenénk.

Az archívum-műveletek felgyorsítása

Észrevehettük, hogy az egyes változatok vagy `base-0` vagy `patch-#` néven szerepelnek, ahol a # jelenti a `base-0`-ra alkalmazandó változtatások sorszámát. Az ***Arch*** napló-rendszerű archiválási formátumot alkalmaz, így rögzíti a projekthez kötődő bármilyen információ változását. Ez azt is jelenti, hogy a nagyméretű, sok változattal rendelkező projektek esetén bizonyos feladatok végrehajtása hosszú időt vehet igénybe. A műveletek meggyorsítására pillanatképet készíthetünk egy adott változatról. Az ***Arch*** pillanatképei egy közzétett változatról készült egyszerű tömörített tar-csomagok. Amikor valamilyen műveletet végrehajtunk, az ***Arch*** megkeresi a legmagasabb sorszámmal rendelkező pillanatképet és a szükséges változtatásokat ettől kezdve hajtja végre:

```
$ tla cacherev
```

Ennek befejezése után futtathatjuk a `tla chachedrevs` parancsot, hogy lássuk, milyen pillanatképekkel rendelkezünk az archívumunkban:

```
lnx-bbc--robot-branch--0.0--base-0
lnx-bbc--robot-branch--0.0--patch-1
```

A könyvtárak

Mivel nem mindig rendelkezünk megfelelő jogosultsággal az archívumban a pillanatképek készítéséhez, hasznos lehet helyi képeket létrehozni a fájlműveletek felgyorsítása érdekében. Az ***Arch*** egy másik fajta pillanatképet is támogat, amelyet könyvtárnak (library) nevez, s amely a különböző ágakból származó közzétett fájlokat tartalmazza. Ez különösen a távoli archívumoknál hatékony, mivel így még egy változat alapképét sem kell letöltenünk ahhoz, hogy alkalmazzuk a változáscsomagunkat:

```
$ mkdir ~/LIBRARY
$ tla my-revision-library ~/LIBRARY
$ tla library-config --greedy ~/LIBRARY
$ tla library-add \
  lnx-bbc-devel@zork.net--gar/lnx-bbc--stable--2.1
```

Ez a könyvtár nem éppen kis méretű, a fenti példához tartozó csomag 78 MB méretű volt 2004 júniusában, de egy lassú kapcsolat esetén a dolog bőven megéri a fáradságot. Ráadásul a hordozható gépek gyakran lassú ATA merevlemezes egységgel rendelkeznek, az archívummal kapcsolatos műveletek pedig nagy terhelést jelenthetnek, mivel az IDE meghajtók sok processzoridőt igényelnek.
Egy önműködően frissülő Arch-könyvtár gyorsabbá és reaktívabbá teszi változáskezelő műveleteinket még a helyi archívumok esetén is.

Folytatás következik

A sorozat következő cikkében azt fogjuk megtanulni, hogyan készíthetünk nyilvánosan elérhető tükrözéseket, amellyel a központi fejlesztők a fejlesztési ágainkból visszaemelhetik a változásokat. Azt is megtanuljuk, hogyan válogathatunk egy sűrűn változó ág változáskészleteiből, és hogyan tudjuk a változáskészleteinket biztonsági célból kriptográfiai módszerrel megjelölni a ***GnuPG*** segítségével.
A sorozat harmadik, befejező részében az ***Arch*** központi fejlesztéseket támogató eljárásairól lesz szó. Megtanuljuk, hogyan kezelhetünk megosztott hozzáférésű archívumokat az ***OpenSSH SFTP*** protokolljával, és hogyan írhatunk parancsfájlokat az archívum feladatainak önműködő végrehajtására.

Linux Journal 2004. november, 127. szám

Nick Moffit, Linux szakértő, Sun Francisco Baynegyedében él. A GNU/Linux LNX-BBC Bootable Business Card csomagjának szerkesztő mérnöke, valamint a GAR szerkesztőrendszer szerzője. Amikor nem a rendszereivel foglalkozik, a városi tömegközlekedés történetét tanulmányozza.

Linux a Linksys Wi-Fi útválasztóin

Ennek a megbízható és költséghatékony platformnak a meghódítása lehet az első lépés egy vezeték nélküli projekt sikere felé. Ha például egy nagyobb területet akarunk lefedni, összeköthetjük a hozzáférési pontokat, a flash-memóriában nagyobb munkaterületet alakíthatunk ki, mesterségesen fölcsavarhatjuk a teljesítményszabályzót, és így tovább.

A vezeték nélküli hálózati megoldások tömegcikké váltak. Ma már egy ***802.11***-es szabványnak megfelelő vagy más néven ***Wi-Fi*** eszköz teljesen hétköznapi terméknek számít. Több ezer rivális gyártó verseng látszólag megegyező termékeivel a ***Wi-Fi*** piacon a pénzünkért. Ilyen forrongó területen természetes, ha a gyártók a legolcsóbb megoldásokat keresik. És mit választanak ennek érdekében? Természetesen a ***Linuxot***.

Napjainkra a Linux vált az olcsó, többfunkciós, vezeték nélküli hálózatépítés elsődleges operációs rendszerévé. A terület egyik legnagyobb szereplője, a ***Linksys*** a következő generációs ***802.11g Wi-Fi*** eszközeivel a ***Linux*** felé fordult. Amikor 2003 elején a ***Cisco*** megvásárolta a ***Linksyst***, hozományként megkapta a linuxos eszközöket, velük pedig egy a kiadatlan ***GPL***-kódok felett folytatott elhúzódó vitát. A nyílt forráskód híveinek néhány hónapos lobbizása után a ***Cisco*** megenyhült és nyilvánosságra hozta a forráskódot.

A ***Linksys WRT54G*** termékét (*1. kép*) az alacsony ár és a beépített hardver teszi különösen érdekessé.

A ***WRT54G***-nek része egy négykapus ***Ethernet*** elosztó, egy ***Ethernet WAN*** kapu, támogatja az új nagy sebességű 54 MB/s ***802.11g*** vezeték nélküli protokollt, és megfelel a korábbi ***802.11b*** eszközökkel szemben támasztott követelményeknek is.

Mégis az teszi igazán érdekessé a ***WRT54G***-t, amit hiányolhatunk belőle. A motorháztető alatt egy 125 MHz-es ***MIPS*** processzor duruzsol 16 MB RAM-mal. Ennek teljesítménye bőven elég lenne néhány komolyabb alkalmazás futtatására is, miért ne adjunk tehát hozzá néhányat?

A fejlesztőkörnyezet üzembe helyezése

A ***Linksys*** honlapján található legfrissebb forráskód 145 MB-os és teljes eszközláncot biztosít a ***MIPS***-re történő keresztfejlesztéshez.

Kövessük a ***WRT54G/src*** alkönyvtár ***README*** nevű fájljának utasításait a közvetett hivatkozás és a ***PATH***-kiegészítések létrehozásához, majd lépjünk be a ***router*** alkönyvtárba és futtassuk a `make menuconfig` parancsot.

1. kép A 100 dollár alatti áron elérhető Linksys WRT54G a 16 MB memóriájával és 125 MHz-es processzorával egy alkalmas Linux-platform, amely támogatja a 802.11b és g protokollokat.

Az első fordításunkhoz hagyjuk meg az alapbeállításokat, és kattintgassunk végig a lehetőségeken a saját beállítófájljaink létrehozásához. Lépjünk feljebb egy könyvtárszinttel a ***WRT54G/src*** alkönyvtárba, majd adjuk ki a `make` parancsot. Ennyi az egész. A ***WRT54G/image*** könyvtárban létrejött egy ***code.bin*** fájl, amely egy tömörített ***cramfs*** fájlrendszert és egy 2.4.20-as ***Linux*** rendszermagot tartalmaz.

Most következik a dolog rémisztő része: hogyan tudjuk ezt az új ***firmware***-t a ***Linksys***-re juttatni? Két eljárás is létezik erre, az egyik a ***tftp***, a másik pedig a web-alapú ***firmware***-frissítő csatlakozófelület. Azt javaslom, hogy első alkalommal a webes frissítéssel próbálkozzunk. Adjuk meg a ***Linksys*** dobozunk címét – ennek alapértelmezett értéke ***198.168.1.1*** –, és jelentkezzünk be. Válasszuk ki az ***Administration (Felügyelet)*** menü ***Firmware Upgrade (Firmware Frissítés)*** menüpontját, majd töltsük fel a ***code.bin*** fájlunkat. Az útválasztó újraindul. Gratulálok, épen most „moddoltuk” a ***Linksys***-dobozunkat.

A Ping-trükk

A *WRT54G*-n lévő *Linux* létezésének felfedezéséhez egy a *Diagnostics (Hibakeresés)* menü *ping* eszközében lévő hiba vezetett el. Az 1.42.2-nél korábbi változatok megengedték tetszőleges kód futtatását a *ping* ablakából, amennyiben *back-tick* operátorok közé tettük. Ha egy korábbi *firmware*-rel rendelkező készülékünk van, próbáljuk meg begépelni az

```
ls -l /
```

parancsot a *ping* ablak IP-cím mezőjébe. És íme, csodálatos módon megjelenik a gyökérkönyvtár tartalma.
A *ping*-trükk a kíváncsiaknak lehetővé teszi, hogy a forráskód módosítása nélkül fedezzék fel a készüléküket. A felfedezésnek ez a módja azonban lassú és unalmas. Igazából egy a készüléken futó parancshéjra lenne szükségünk. A *ping*-trükk forráskódban történő kiterjesztésével egy olyan egyedi *firmware*-képfájl hozható létre, amely hálózatos felületen keresztül is rendelkezik a *Linux* héj teljes hatékonyságával. A parancshéj elkészítésének leírását a hálózaton keresztül elérhető forráscímek között találjuk meg.
De miért állnánk meg itt? Az eredeti *firmware carmfs* fájlrendszere 200K szabad helyet hagy a f*lash*-memóriában. Helyet találhat itt magának egy sereg hasznos alkalmazás, akár egy *telnet* program, egy *Secure Shell (Biztonságos parancshéj)*, esetleg egy *VPN* ügyfél vagy kiszolgáló.

A wl parancs

Az egyik hasznos parancs, amit a *Linksys* csak bináris formában szolgáltat a `wl`. A `wl` parancs több tucat olyan beépített parancsot tartalmaz, amellyel a vezeték nélküli beállításokat vezérelhetjük, beleértve a népszerű teljesítményt szabályozó beállítást is. A `wl` parancs paraméter nélküli beírásával egy teljes listát kapunk a képességeiről.
A *WRT54G* alapértelmezett teljesítménybeállítása 28 milliwatt, és ez a beállítás kívülről nem is állítható. A *ping*-trükk vagy egy parancshéj segítségével azonban használhatjuk a `wl` parancsot, paraméterként megadva utána a `txpwr` alparancsot és az 1 és 84 milliwatt közti teljesítményértéket. Ez az érték a következő újraindításig megemeli vagy csökkenti az alapértelmezett teljesítménybeállítást.
A teljesítmény növelése, vagy a beépített antenna cseréje növelheti a kisugárzott teljesítményt, amivel megsérthetjük az erre vonatkozó helyi szabályokat. Ha kicseréljük az antennákat és csökkentjük a teljesítményt, akkor úgy tudjuk jelentősen megnövelni az egység hatótávolságát, hogy ezzel nem sértjük az engedélyezett sugárzási teljesítményre vonatkozó helyi előírásokat.
A *WRT54G* két külső antennát támogat, és önműködően választ a kettő közül attól függően, hogy melyikkel fogta be a legutóbbi aktív adatcsomagot. Amikor egy új, nagyobb hatékonyságú antennát szerelünk fel, nyilván nem ezt a beállítást szeretnénk használni, hanem minden alkalommal a nagyobb teljesítményű antenna használatára szeretnénk a készüléket utasítani. Ezt a bejövő adatok esetén a `wl txant` paranccsal, küldésre pedig a `wl antdiv` paranccsal tehetjük meg. A 0 paraméter a bal oldali antennát jelöli ki, az 1 pedig a jobb oldalit, ha elölről nézzük a készüléket.

Az SSH (biztonsági parancshéj) telepítése

Egy vállalkozó szellemű felhasználó a teljes *OpenSSH* eszközkészletet átültette a *Linksys* készülékébe. Sajnos az *OpenSSH* bináris állományának mérete miatt sok alapvető *Linksys* funkciót is el kellett távolítani a megfelelő méretű szabad hely létrehozásához, ráadásul a kapott RAM-igény is a rendelkezésre álló memória határán volt. Igazából egy kis memóriaigényű *SSH*-kiszolgálóra lenne szükség, s ezt az igényt a *Dropbear* kiszolgáló szépen teljesíti is. *Matt Johnson* a *Dropbear SSH*-démont kifejezetten olyan, szűkös memóriával rendelkező eszközökön való futtatásra tervezte, mint amilyen a *Linksys*. Az eredeti *Linksys Linux* megvalósításából hiányzik egy sereg olyan alapvető fájl, ami a többfelhasználós *Linux* rendszereken nélkülözhetetlen. Ezek közül kettő – a */etc* könyvtárban lévő *passwd* és *groups* – a *Linux* alkalmazások többségének futtatásához mindenképpen szükséges. Ahhoz, hogy futtatni tudjuk a *Dropbear* kiszolgálót, ezeket a fájlokat hozzá kell adnunk a *flash*-csomaghoz. Egy jelszóval nem rendelkező root bejegyzést tartalmazó *passwd* fájl és egy megfelelő *groups* fájl létrehozásával már majdnem alkalmas is a környezet a *Dropbear* futtatására. Ezek a fájlok a *flash* képfájl */etc* könyvtárába kerülnek és a *Linksys*-en csak olvashatóak.
Futtatás közben a *Dropbear* ezeken kívül igényli a nyilvános kulcshoz való hozzáférést, amely az SSH kapcsolatfelvételhez és hitelesítéshez szükséges, valamint a *known_hosts* fájlt, amely az elfogadandó ügyfélgépek nyilvános kulcsait tartalmazza. A `dropbearkey` programmal egy pillanat alatt létrehozhatjuk a saját kulcsunkat, de a kulcs tárolása a *Linksys*-en már egy kicsit trükkösebb.
A *WRT54G* egy hash-térképet tartalmaz a kulcsok név-érték párjaival, amelyek egy *nvram*-nak nevezett nem felejtő memóriában tárolódnak. A kapott `nvram` segédprogram és *API* lehetővé teszi, hogy írjuk és olvassuk ezt a memóriaterületet. A *Dropbear* saját kulcsát és a saját *.ssh* könyvtárunkban tárolt *id_rsa.pub* fájlban lévő nyilvános kulcsunkat az nvram-ban tároljuk és rendszerindításkor innen másoljuk a memóriában lévő virtuális lemezre.
Lefordítjuk a *Dropbeart* a kulcsfájl-hitelesítés beállításával, és máris biztonságos módszerünk van a *Linksys*-re történő bejelentkezéshez. Ha jelszavas belépésre van szükségünk, egy folttal kiegészítve a *Dropbear* kódját rávehetjük, hogy a rendszerjelszót olvassa ki az *nvram*-ból, és így lehetővé válik a jelszavas bejelentkezés is.

A Flash-memória tömörítésének növelése

Az *SSH* és a `telnetd` programokhoz hasonló eszközök telepítése után hamarosan tapasztalni fogjuk, hogy a *Linksys* firmware képfájlja feszegetni kezdi az eszközben lévő falshmemória tárolókapacitásának határait. Szükségünk lenne egy olyan fájlrendszerre, ami jobb tömörítést nyújt, mint a *cramfs* és egyúttal megfelel a *Linksys Linux* rendszermag elvárásainak is.
Az alapértelmezett *cramfs* fájlrendszer 4K méretű blokkokba tömöríti az adatokat, ez a 4K mérethatár azonban korlátozza az elérhető tömörítési arányt. Ha találnánk egy olyan fájlrendszert, ami nagyobb adatblokkokat használ, de a hozzárendelése megfelel az operációs rendszer oldalméretének, sokkal több adatot és programot tudnánk tárolni a firmware-ben.

Philip Lougher ***sqashfs*** fájlrendszere 32K méretű adatblokkokat használ a tömörítéshez és megfelel a 2.4 és 2.6-os rendszermagoknak. Ha szerét ejthetnénk a ***Linksys firmware cramfs***-ről ***squashfs***-re történő átültetésének, talán elég helyünk lenne arra, hogy egy ***VPN*** ügyfelet és kiszolgálót is telepítsünk a rendszerre.

A ***Linksys*** rendszermagja egy 2.4.20 verziójú rendszermag, amelyet a ***Broadcom*** írt át. A ***Broadcom*** cég vezető szerepet tölt be a 802.11g áramköri lapkák gyártásában és a ***WRT54G***-ben lévő központi egység és rádióáramkör is a munkáját dicséri. A ***squashfs tar***-fájlja tartalmaz foltokat a 2.4.20-tól a 2.4.22-ig terjedő rendszermagokhoz, de sajnos ezek közül egyik sem alkalmazható teljesen a ***Broadcom*** rendszermag-fájához, ezért egy kis manuális beavatkozásra is szükség lesz. A legkevesebb hibát tartalmazó folt a 2.4.22-es változat, ez az alkalmazásakor csak egy kódrészletet hiányol. A folt fájljának elolvasásával és a hiányzó darab megkeresésével kézzel módosíthatjuk a kódot. A ***Sveasoft*** honlapján találhatunk egy ***WRT54G***-hez készült ***squashfs***-foltot is.

A következő lépés a ***Broadcom*** rendszermag indítókódjának a szerkesztése és egy ***squashfs*** ellenőrzőrész hozzáadása. A do_mount.c fájl szinte ugyanazt a kódot tartalmazza és jól használható, amikor az ***arch/mips/brcm-boards/bcm947xx*** könyvtárban lévő ***startup.c*** fájlt frissítjük.

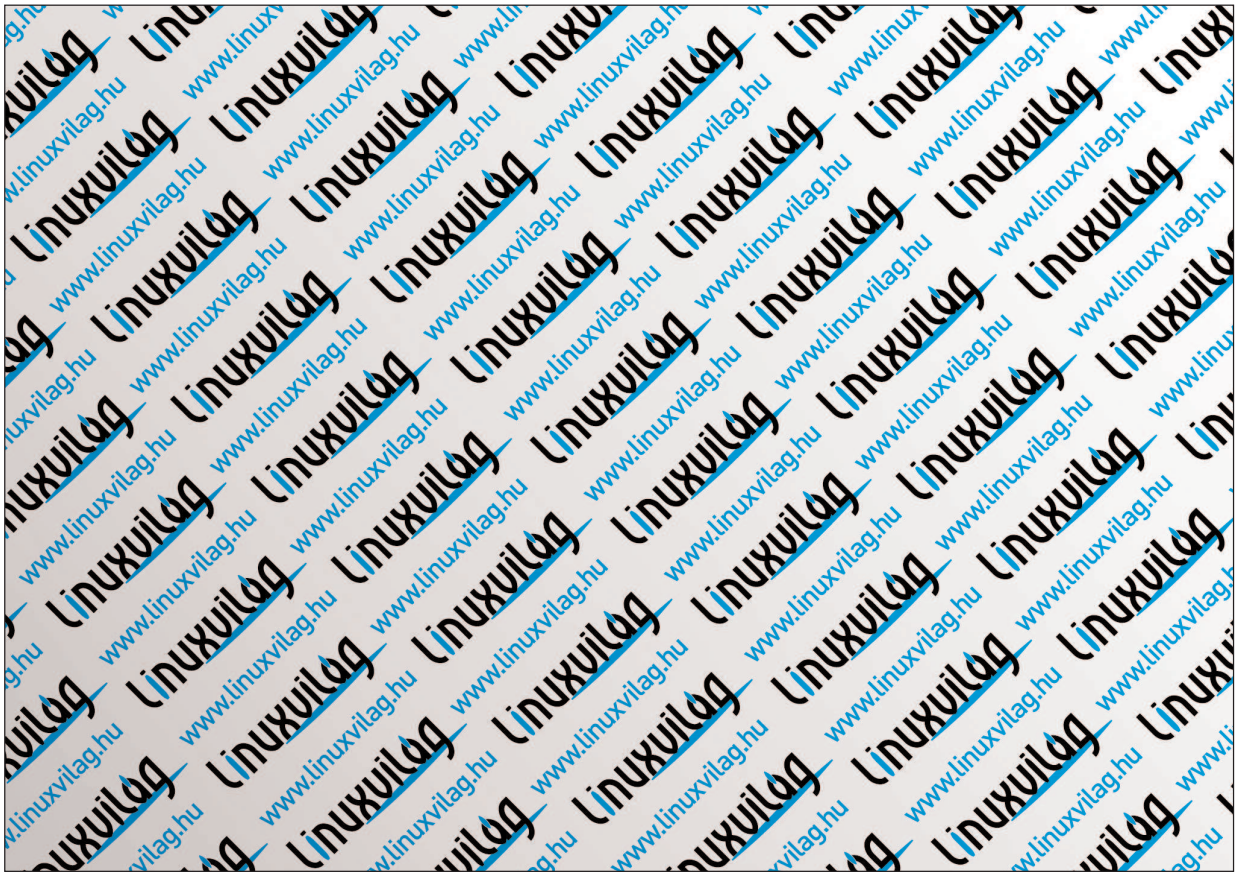
A rendszermag megfoltozása után az útválasztó ***Makefile***-ját kell megfoltozni oly módon, hogy hozzon létre egy ***squashfs***-képfájlt, és a Linux rendszermag beállításaiba a ***squashfs*** támogatását is bele kell foglalni. Az elért eredmény bőven megéri a befektetett munkát: az újrafordítás után mintegy 500K-val több szabad helyet kapunk az eredeti ***cramfs*** fájlrendszerhez képest.

A vezeték nélküli elosztórendszer

A normál ***WRT54G*** egy vezeték nélküli hozzáférési pont (AP), ami azt jelenti, hogy képes más vezeték nélküli ügyfelekkel való kapcsolattartásra, azonban más vezeték nélküli hozzáférési pontokkal már nem. A wl paranccsal elérhető, hogy a ***Wireless Distribution System (WDS, vezeték nélküli elosztórendszer)*** segítségével más hozzáférési pontokhoz kapcsoljuk, vagy hogy vezeték nélküli ügyfélként működtessük.

A ***Wireless Distribution System*** egy olyan ***IEEE*** műszaki leírás, amely lehetővé teszi, hogy egy nagy távolságokat áthidaló hálózatban összekapcsoljuk a vezeték nélküli hozzáférési pontokat. Habár ezért némi teljesítményveszteséggel kell fizetnünk, a kapott kiterjesztett vezeték nélküli hálózat sokkal nagyobb hatótávolságú, mint amilyen a külön álló hozzáférési pontok alkalmazásával elérhető lenne.

Két hozzáférési pont ***WDS***-sel történő összekapcsolásához ismernünk kell a készülékek egyedi ***MAC***-azonosítóit. Jelentkezzünk be az egyes készülékekre, és futtassuk a `wl wds [MAC-cím]` parancsot megadva a másik készülék adatát. Ennek hatására egy új, ***wds0.2*** eszköz jelenik meg a készülékeken, amelyhez most már IP-cím rendelhető. Ha elvégeztük az IP-címek hozzárendelését és a két készülék közti útvonalválasztást is beállítottuk, akkor a ***ping*** paranccsal ellenőrizhetjük is az összeköttetést a két készülék között.



A Linksys WRT54G forrásfa

Amikor kibontjuk a *Linksystől* származó *GPL* forrást, a *WRT54G* főkönyvtár alatt egy könyvtárszerkezet jön létre. A következőkben ismertetem ennek fontosabb részeit.

A *tar*-csomag főkönyvtára a */WRT54G*. A fő *Makefile* a */release/src* könyvtárban található. A forrás kicsomagolása után olvassuk el az itt található *README* fájlt a fordításhoz szükséges tudnivalókért.

A *Linksys* egységgel kapott minden program a */release/src/router* könyvtárból került lefordításra. Ha további programokat szeretnénk hozzáadni, azt itt tehetjük meg és módosítanunk kell az itt található *Makefile*-t. A *Linux* rendszermag forrásfáját a *Broadcom* cég módosította, amely a *WRT54G*-ben található áramköri lapkák és processzor gyártója. A rendszermag módosításainkat és foltjainkat itt, a */release/src/linux/linux* könyvtárban kell létrehoznunk.

Egy közvetett hivatkozást kell létrehoznunk az */opt* könyvtárból az itt lévő */tools/brcm* könyvtárra. A *brcm* alatti két könyvtárat hozzá kell adnunk a *PATH* változónkhoz. Erről részletesen a *README* fájlban olvashatunk.

A foltok és a frissített forráskód letölthető a *Sveasoft* oldaláról.

Minden egyes ***WDS***-kapcsolat a hálózaton belüli adatforgalom megkettőződésével jár. Mivel a ***802.11g*** protokoll félduplex jellegű, ez a hálózati átvitel feleződését jelenti. Amennyiben a hozzáférési pontok 54MB/s sebességen működnek, ez nem nagy áldozat, amíg a kapcsolatok száma három vagy annál kevesebb.

Ügyfél-módú összeköttetés

Az összekapcsolás egyszerűbb formája az egyik készülék ügyfélként való beállítása és egy hozzáférési ponthoz való csatlakoztatása. Ezt ***Ethernet*** hídnak hívják és számos eszközt kifejezetten ezzel a céllal működtetnek.
Az ügyfélmódot a Linux rendszermagban kell kiválasztani és ezzel a beállítással kell a rendszermagot lefordítani. Ha ezzel készen vagyunk, a rendszermagot a ***Broadcom*** egy bináris moduljával kell összeépíteni, amely tartalmazza mind az AP-, mint az ügyfélmódot. A `wl ap 0` ügyfélmódba kapcsolja a készüléket, míg a `wl join [SSID]` egy másik hozzáférési ponthoz köti. Ha az ügyfélen az útvonalválasztás alapértelmezett átjárójának a a hozzáférési pont IP-címét adjuk meg, az ügyfél azonnal a hozzáférési pontra irányítja az adatokat és a híd aktívvá válik. Több AP-ügyfél pár is beállítható, ami egy másik lehetőséget kínál a fent vázolt ***WDS***-módszer mellett.

A nyílt forráskód hatékonysága

A ***Linux*** megtalálta a helyét a szuperszámítógépektől a beágyazott rendszerekig minden területen, így a ***Linksys*** eszközökben is. A ***Linuxra*** történő átállás a kiélezett versenyt folytató piaci szereplőkkel szemben támasztott nagy teljesítmény és alacsony költségigény eredménye. Sok hasonló vezeték nélküli útvonalválasztó, így a ***Belkin F5D7230-4***, a ***Buffalotech WBR-G54*** és az ***ASUS WL-300g*** és ***WL-500g*** használ ***Linuxot*** a firmware-jében, és a lista napról napra bővül. Sajnos egyik cég sem tesz eleget a ***GPL*** követelményeinek, és nem szabadítja fel a forráskódot. A jogi kifogások mellett ezek a termékek lehetőségeikben és szolgáltatásaikban is rövid időn belül messze le fognak maradni a ***Linksys*** nyílt forrású termékei mögött. A ***Linksys*** firmware-csomagjai naponta jelennek meg meglepő új lehetőségekkel. Az írásom idején léteznek a ***Linksys WRT54G***-hez olyan firmware-fordítások, amelyek támogatják a ***VPN***-eket, teljesítményszabályozást, antenna-kiválasztást, ügyfél- és ***WDS***-módot, sávszélesség-felügyeletet és még rengeteg mindent, amely különféle forrásokból érhető el. A világháló és a nyílt forráskód összekapcsolásával, otthon vagy kis irodákban alkalmazható útvonalválasztóból hatékony sokfunkciós eszköz hozható létre.
Figyelmeztetés: kísérleti firmware használatával tönkretehetjük a készülékünket és a ***Linksys*** garanciáját is elveszíthetjük. Ha alkalmi felhasználók vagyunk és csak otthoni vagy irodai vezeték nélküli hozzáférésre van szükségünk, jobb, ha nem kísérletezünk a leírtakkal, hanem ehelyett a ***Linksys*** gyári firmware-jét használjuk. Ha viszont hajlandóak vagyunk még a készülékünket is kockára tenni egy kis kísérletezésért, sokkal többet kihozhatunk a dobozból, mint amit a csomagolásán lévő leírásban olvashatunk – köszönet érte a ***Linuxnak*** és a nyílt forrású fejlesztésnek.

Linux Journal 2004. augusztus, 124. szám

James Ewing (james.ewing@sveasoft.com) több mint 20 éve vállalkozó és szoftverfejlesztő. Kaliforniából egy évtizede kötözött át Svédországba, ahol jelenleg próbálja az idejét igazságosan elosztani a felesége, két gyermeke és a Muppet Show svéd szakács szerepének gyakorlása közt.

KAPCSOLÓDÓ CÍMEK

- http://openwrt.ksilebo.net/
- http://sourceforge.net/projects/openwrt/
- http://www.linksys.com/products/product.asp?prid=508&scid=35
- http://www.batbox.org/wrt54g-linux.html
- http://dg834.grandou.net/
- http://openbrick.org/
- http://www.soekris.com/
- http://meshcube.org/english/specs.html

Hálózatkezelés NSA Security-Enhanced Linux alatt

Egy gyakorlati példa segítségével küzdjük le az SELinux bonyolultsága miatti félelmeinket, és egyszerűbb kiszolgálóinkat is vértezzük fel SELinux alapú védelemmel.

Írásomban áttekintem, az ***SELinux*** segítségével hogyan növelhető a hálózati rendszerek biztonsága, valamint ismertetem hálózati vonatkozású védelmi vezérlőinek felépítését és megvalósítását. Ezután átveszünk egy az ***SELinux*** házirendjének egy egyszerű hálózati alkalmazás zárolására való használatát szemléltető példát.

Az SELinux szerepeinek, típusainak és tartományainak áttekintése

Az ***SELinux*** erőteljes általános védelmet nyújt a hálózati rendszereknek. Lehetővé teszi, hogy a rendszereket „szűkre szabva" a szolgáltatásoknak csak a működésükhöz feltétlenül szükséges jogosultságokat adjuk meg. A lehető legkevesebb jogosultság alapelvének ilyen jellegű megvalósításával meg lehet előzni a biztonsági határok hibás vagy rosszindulatú kód, vagy hibát vétő vagy rosszakaratú felhasználó miatt előforduló átlépését.

Egy külső személyek számára szolgáltatásokat nyújtó webkiszolgálót például számos módon lehet védeni:

- Szükségtelen szolgáltatások letiltása
- A kiszolgáló program ***chroot börtönben (jail)*** való futtatása
- Helyi csomagszűrés ***iptables*** segítségével
- Jogosultságkezelés ***sudo*** segítségével
- Beállító fájlok zárolása

A fentiekkel több réteggel érjük el a biztonság növelését, tulajdonképpen a mélybe nyúló védelem alapelvét követjük. Az ***SELinux*** a rendszert egy további biztonsági réteggel bővíti, a ***kötelező hozzáférés-vezérléssel (mandatory access control, MAC)***. A normál ***SELinux*** a ***MAC***-et ***típusszabályokkal (type enforcement, TE)*** és ***szerep alapú hozzáférés-vezérléssel (role-based access control, RBAC)*** valósítja meg, mindezt egy központi kezelésű biztonsági házirend vezérli, melynek betartatásáról a rendszermag gondoskodik. A hagyományos ***UNIX*** biztonságtól eltérően a normál felhasználók semmilyen ellenőrzési jogot nem kapnak az ***SELinux*** biztonsági házirendjére (erre utal a kötelező jelző), míg a rendszergazda, a root feladatköre több, egymástól elkülönülő felügyeleti szerepre is feloszthatő, amit a feladatok és a felelősség szétosztásának nevezünk.

A hagyományos, ***önkényes hozzáférés-vezérlést (discretionary access control, DAC)*** a ***TE*** modell megszigorítja, ugyanis az operációs rendszer egyes objektumaihoz – folyamatokhoz, fájlokhoz és hálózati erőforrásokhoz – típusokat rendel, majd pontosan körülírja a közöttük folyó párbeszédekre vonatkozó szabályokat. (Egy folyamat típusát általában tartománynak nevezik.) Mindezekkel az eszközökkel kifinomult hozzáférés-vezérlés valósítható meg, és a lehető legkevesebb jogosultság elve az operációs rendszer általános értelemben vett megerősítésén túl is messze kiterjeszthető.

Az SELinux hálózati hozzáférés-vezérlési alrendszere

Az ***SELinux*** a 2.6-os rendszermag ***LSM (Linux Security Modules, Linux biztonsági modulok)*** és ***Netfilter API***-jaira épül. Az ***LSM*** és a Netfilter egyaránt egy a rendszermag stratégiai pontjain elhelyezett csatlakozási pontokból, „kampókból" álló hozzáférés-vezérlési keretrendszer. A rendszermagon belüli adatáramlások a csatlakozási pontokon keresztül a biztonsági modulok, például az ***SELinux*** felé irányulnak, ezek elvégzik a hozzáférés-vezérlési számításokat, majd közlik az eredményt a csatlakozási ponttal. A csatlakozási pont a biztonsági modul utasítása alapján vagy engedélyezi az adatáramlást, vagy megtiltja azt.

Az ***SELinux*** egyik fontos tervezési alapelve, hogy az operációs rendszer objektumainak szintjén ellenőrzi a hozzáférést. Ahelyett, hogy egy biztonsági figyelő egyszerűen csak megmondaná, adott program adott kapcsolókkal és átadott értékekkel elindíthat-e egy rendszerhívást vagy sem, az ***SELinux*** a program futása során annak teljes biztonsági környezetét átlátja, továbbá hozzáfér az elérni kívánt objektum és a végrehajtani kívánt művelet biztonsági címkéjéhez is. A rendszergazda által futtatott `ls` parancs például teljesen más, mint a normál felhasználók által használt.

Egy ***SELinux*** engedély általános formátuma a következő:

- `művelet (forrás környezet)`
- `(cél környezet):(cél objektumosztályok)`
- `engedélyek`

Egy példa az SELinux házirendjéből:

```
allow bluetooth_t self:socket listen;
```

A fentiek értelmében a `bluetooth_t` tartomány hallgatózási (listen) engedélyt kap a saját biztonsági környezetével (kontextusával) felcímkézett foglalatokhoz (socket). Egy a `bluetooth_t` tartományban futó folyamat tehát egy általa

birtokolt foglalaton jogosult a listen() hívás alkalmazására. A `self` (önmaga) csak egy egyszerű jelölés a cél környezet egyenlővé tételére a forrás környezettel. A foglalatokkal kapcsolatos házirendben gyakran láthatunk ilyen parancsokat, ugyanis a foglalatok általában a létrehozó folyamattal azonos biztonsági környezet címkét kapnak.

A hálózati objektumok címkézése

SELinux alatt az objektumok biztonsági címkéje a következő formátumot követi:

```
felhasználó:szerep:típus
```

Például:

```
root:staff_r:staff_t
```

Ez egy olyan folyamat biztonsági környezete, amelyet a `staff_r` szerepen keresztül, a `staff_t` tartományban a `root` futtat. A 80-as kapuhoz tartozó címke a következő:

```
system_u:object_r:http_port_t
```

A `system_u` felhasználó és az `object_r` szerep a rendszerobjektumok esetében alapértelmezett. Semmi értelme nem volna egy kapuhoz valódi felhasználónevet vagy szerepet rendelni, hiszen senki sem birtokolja, és nem kezdeményez az ***SELinux*** elbírálását igénylő műveleteket sem.

Foglalatok

A foglalatok címkézése a hozzájuk tartozó fájlleíró (i-node) alapján történik. Egy foglalat lehet általános, vagy tartozhat az alábbi foglalat alosztályok valamelyikébe:

- ***UNIX*** folyam (***UNIX*** stream)
- ***UNIX*** datagram
- ***TCP***
- ***UDP***
- Nyers (***ICMP*** vagy egyéb nem ***TCP***/***UDP***) (***Raw***)
- ***Netlink*** család
- Csomag (Packet)
- Kulcs (***pfkeyv2***) (***Key***)

A biztonsági házirendben a foglalatok alosztályait is meg lehet különböztetni egymástól, így magas fokú rugalmasságot lehet elérni, és kifinomult vezérlést lehet megvalósítani a különféle hálózati protokollok felett:

```
allow lpd_t printer_port_t:tcp_socket name_bind;
```

A szabály értelmében csak az `lpd_t` tartományban létrehozott ***TCP*** foglalatok kötődhetnek a `printer_port_t` típusú kapukhoz.

Kapuk

Az ***IPv4*** és az ***IPv6*** kapuk címkézése implicit módon, a rendszermagon belül, a házirend által megszabottak szerint történik. A kaputípusok címkézésének formátuma a következő:

```
portcon protokoll { kapuszám | kaputartomány }
        környezet
```

Az alábbi szabály a szabványos nyomtatókapu címkézéséhez adja meg a biztonsági környezetet:

```
portcon tcp 515 system_u:object_r:printer_port_t
```

Hálózati csatolók

Minden hálózati csatoló (network interface, netif) a házirendben megadottak szerint kap biztonsági környezetcímkét. A hálózati csatolók címkézése a következőképpen történik:

```
netifcon csatoló környezet
alapértelmezett_üzenet_környezet
```

Az `alapértelmezett_üzenet_környezet` érték a csatolón keresztül beérkező üzenetek címkézésére szolgálna, ám jelenleg használaton kívül van.
Két példa házirend netif címkéjére:

```
netifcon eth0 system_u:object_r:netif_intranet_t
➥[...]
netifcon eth1 system_u:object_r:netif_extranet_t
➥[...]
```

Csomópontok

SELinux alatt a csomópont egy ***IPv4*** vagy ***IPv6*** címet vagy hálózati maszkot jelent. Lehetővé teszi állomás- és hálózatcímek házirenden keresztül való biztonsági címkézését. A csomópontok címkézése a következő formátumot követi:

```
nodecon cím maszk környezet
```

Példa helyi cím címkézésére:

```
nodecon 127.0.0.1 255.255.255.255
        system_u:object_r:node_lo_t
nodecon ::1
➥ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
        system_u:object_r:node_lo_t
```

Hálózati csatlakozási pontok és engedélyek

A hozzáférés-vezérlési csatlakozási pontokat minden foglalatokkal kapcsolatos rendszerhíváshoz megvalósították, így minden foglalat alapú hálózati protokoll működését ellenőrizni lehet ***SELinux*** házirenddel. A csatlakozási pontok némelyike csupán a rendtartást szolgálja, ám nagyobb részük egy vagy több hozzáférés-vezérlési engedély ellenőrzését végzi. Általános foglalatvezérlő meglehetősen sok van, ezért a csatlakozási pontok, a foglalatokra vonatkozó rendszerhívások és az engedélyek közötti kapcsolatokat az *1. táblázatban* foglaltam össze.
Belül a `socket()` rendszerhívás két csatlakozási pontra oszlik. Az `selinux_socket_create` csatlakozási pont annak ellenőrzésére alkalmas, hogy a folyamat létrehozhatja-e a megadott típusú foglalatot. Az `selinux_socket_post_create` felügyeleti csatlakozási pont a foglalathoz rendelt fájlleíró biztonsági címkéjének megadását és foglalattípusának meghatározását teszi lehetővé.
Az ***SELinux*** engedélyek a rendszerhívások és az egyéb műveletek biztonsági szemszögből való elvonatkoztatásához is hozzájárulnak. Vegyük észre például, hogy

1. táblázat *A csatlakozási pontok, a foglalatokra vonatkozó rendszerhívások és az engedélyek kapcsolatai*

Csatlakozási pont	Rendszerhívás	SELinux engedély
`selinux_socket_create`	`socket`	`create` (létrehozás)
`selinux_socket_post_create`	`socket`	n. a.
`selinux_socket_bind`	`bind`	`bind` (kötődés)
`selinux_socket_connect`	`connect`	`connect` (kapcsolódás)
`selinux_socket_listen`	`listen`	`listen` (hallgatózás)
`selinux_socket_accept`	`accept`	`accept` (fogadás)
`selinux_socket_sendmsg`	`sendmsg, send, sendto`	`write` (írás)
`selinux_socket_recvmsg`	`recvmsg, recv, recvfrom`	`read` (olvasás)
`selinux_socket_getsockname`	`getsockname`	`getattr` (jellemzők lekérdezése)
`selinux_socket_getpeername`	`getpeername`	`getattr` (jellemzők lekérdezése)
`selinux_socket_setsockopt`	`setsockopt`	`setopt` (kapcsolók megadása)
`selinux_socket_getsockopt`	`getsockopt`	`getopt` (beállítások lekérdezése)
`selinux_socket_shutdown`	`shutdown`	`shutdown` (leállítás)

2. táblázat *A fájlokra jellemző csatlakozási pontok és engedélyek*

Csatlakozási pont	Rendszerhívás	SELinux engedély
`selinux_file_ioctl`	`ioctl`	`ioctl`
`selinux_inode_getattr`	`fstat`	`getattr` (jellemzők lekérdezése)
`selinux_inode_setattr`	`fchmod, fchown`	`setattr` (jellemzők beállítása)
`selinux_file_fcntl`	`fcntl`	`lock` (zárolás)
`selinux_file_lock`	`fcntl, flock`	`lock` (zárolás)
`selinux_file_permission`	`write, write, read`	`append, write, read` (hozzáfűzés, írás, olvasás)

a `getsockname()` és a `getpeername()` rendszerhíváshoz egyaránt a `getattr` engedély tartozik. Az ***SELinux*** ezek biztonsági szerepét egyenlőnek veszi. Hasonlóan, minden `sendmsg()` és `recvmsg()` alapú rendszerhívás biztonságfelügyeleti szempontból egyszerű írásnak vagy olvasásnak számít. Akit a részletek is érdekelnek, a csatlakozási pontokat megvalósító kódot a 2.6-os rendszermag forrásában, a ***security/selinux/hooks.c*** fájlban találja.

Mivel a foglalatok is fájlok, a fájlokra vonatkozó hozzáférésvezérlés egy részét is megöröklik. A 2. *táblázat* a fájlokra jellemző, de a foglalatokra is érvényes csatlakozási pontokat tartalmazza.

A UNIX tartományra vonatkozó engedélyek

Linux alatt a ***UNIX*** tartomány foglalatait elvonatkoztatott névtérben, a fájlrendszertől függetlenül lehet létrehozni. Az elvonatkoztatott névtérben lévő ***UNIX*** tartománybéli foglalatok közötti adatcserék, illetve ezek irányultságának ellenőrzésére további csatlakozási pontokat valósítottak meg. Az `selinux_socket_unix_stream_connect` csatlakozási pont a `connectto` engedély meglétét ellenőrzi, amikor egy ***UNIX*** tartománybéli foglalat adatfolyam (***stream***) kapcsolatot próbál létesíteni egy másikkal.

Az `selinux_socket_unix_may_send` csatlakozási pont a `sendto` engedély meglétét követeli meg ahhoz, hogy az egyik ***UNIX*** tartománybéli foglalat datagramot küldhessen egy másiknak.

A ***UNIX*** tartománybéli foglalatok egy további ***Linux*** alatti szolgáltatása az egyenrangú felek hitelesítése a `SO_PEERCRED` – foglalatokra vonatkozó – kapcsolóval. A kapcsoló az egyenrangú fél felhasználó-, csoport és folyamatazonosítójának lekérdezését eredményezi. ***SELinux*** alatt egy-egy egyenrangú fél biztonsági környezetét egy új, szintén foglalatokra vonatkozó művelettel, a `SO_PEERSEC` kapcsolóval is megtudhatjuk. A `getsockopt(2)` hívást ezzel a kapcsolóval indítva a `selinux_socket_getpeersec` csatlakozási pontra adhatjuk át a vezérlést, mely a felhasználó által megadott pufferbe másolja a biztonsági környezetet. Ezt a megoldást helyi folyamatok közötti kommunikációra, például ***megnövelt biztonságú adatbusz (Security-Enhanced DBUS)*** megvalósítására használják.

Netlink vezérlők

A ***Netlink*** foglalatok üzenet alapú felhasználó/rendszermag adatcserét tesznek lehetővé. Például a rendszermag irányítótáblájának és az ***IPSec*** működésének beállítására használhatók.

A *Netlink* adatcsere aszinkron jellegű, az üzenetek küldése és fogadása eltérő biztonsági környezetben is lehetséges. *Netlink* csomag továbbításakor a küldő biztonsági engedélyei egy képességhalmaz formájában a csomag mellé kerülnek, fogadáskor pedig megtörténik az ellenőrzésük. Így lehetővé válik, hogy például a rendszermag forgalomirányító kódja ellenőrizze, hogy az irányítótábla frissítését küldő felhasználónak valóban van-e joga ilyen frissítés küldésére.
Az *LSM* tervezet részeként a képességeket kezelő kódok a rendszermagból átkerültek egy biztonsági modulba, így az *LSM*-ek szükség esetén többféle biztonsági modell megvalósítására is alkalmasak.
Az *SELinux* modul az `selinux_netlink_send` csatlakozási pontot használja arra, hogy kizárólag a `NET_ADMIN` képességet a rendszermag által küldött *Netlink* csomagokba bemásolja. Az `selinux_netlink_recv` csatlakozási pont a biztonsági szempontból fontos üzenetek fogadásakor jut szerephez. Az *SELinux* ezt a csatlakozási pontot alkalmazza annak ellenőrzésére, hogy a `NET_ADMIN` képesség másolása a csomag elküldésekor megtörtént-e, valamint a küldő folyamat valóban rendelkezett-e ezzel a képességgel.
Egyre több *Netlink* család kerül megvalósításra, és az *SELinux* a biztonsági szempontból kritikus *Netlink* foglalatokhoz alosztályokat ad meg. Így a foglalatok feletti felügyelet az egyes *Netlink* családokra egyedileg állítható be, amivel elérhető például az irányítási üzenetek és a rendszer naplózási üzeneteinek megkülönböztetése.
Az *SELinux* az `selinux_netlink_send` csatlakozási pont révén annak meghatározására is képes, hogy a *Netlink* foglalatok egyes típusain az üzenetek olvasási vagy írási műveletek, majd rendre alkalmazza az `nlmsg_read` vagy az `nlmsg_write` engedélyeket. Ezáltal rendkívül kifinomult házirend állítható össze, például adott tartomány számára engedélyezni lehet az irányítótábla kiolvasását, miközben meg lehet tiltani annak frissítését.

IPv4 és IPv6 vezérlők

Az *SELinux* számos vezérlőt tartalmaz a *TCP*, az *UDP* és a *nyers (raw)* foglalat alosztályokhoz. A `node_bind` engedély szabja meg, hogy egy foglalat kötődhet-e egy megadott típusú csomóponthoz. Értelemszerűen csak helyi IP-címnél érdemes használni, és általa adott démon adott IP-címhez való kötődését lehet meggátolni.
A `name_bind` engedély azt szabja meg, hogy egy foglalat kötődhet-e adott típusú kapuhoz. Csak akkor jut szerephez, ha a kapu száma a helyi kaputartományon kívülre esik.
A helyi kaputartomány az, ahonnan a rendszermag a kapuszámok önműködő kiosztását végzi (például, amikor egy kimenő *TCP*-kapcsolat forráskapujának számát kiválasztja), megadására a `net.ipv4.ip_local_port_range` rendszerhívás használható. Egy átlagos rendszeren a tartomány a következő:

```
$ sysctl net.ipv4.ip_local_port_range
net.ipv4.ip_local_port_range = 32768    61000
```

A `name_bind` meghívására tehát csak akkor kerül sor, ha egy foglalat egy e tartományon kívülre eső kapuhoz kötődik. Az *SELinux* az 1024 alatti számú kapuk esetében mindig megvizsgálja az engedélyeket, függetlenül a sysctl beállítástól. Mindkét kötődés vonatkozású vezérlő az `selinux_socket_bind` csatlakozási pontról kerül meghívásra, mely viszont a ***bind(2)*** rendszerhíváson keresztül jut a vezérléshez.
A `send_msg` és a `recv_msg` engedély annak szabályozására alkalmas, hogy egy foglalat egy adott típuson vagy kapun keresztül küldhet vagy fogadhat-e üzeneteket.
Számos vezérlő határozza meg, hogy *TCP*, *UDP* vagy nyers foglalaton keresztül lehet-e megadott típusú netif és csomópont objektumok felé és felől csomagokat küldeni és fogadni; ezek a `tcp_send`, a `tcp_recv`, az `udp_send`, az `udp_recv`, a `rawip_send` és a `rawip_recv`.
Mindezeket az üzenet alapú vezérlőket az `selinux_sock_rcv_skb` csatlakozási pont hívja meg a beérkező üzenetekre, ez ugyanis a hálózati verem első olyan pontja, ahol a csomagok egyértelműen hozzárendelhetők a fogadó foglalatokhoz. A kimenő csomagok kezeléséhez az *SELinux* bejegyez egy *Netfilter* csatlakozási pontot, és az IP rétegben elfogja a csomagokat. A kimenő csomagokhoz csatolt tulajdonlási adatok ebben a fázisban is megmaradnak.
A fenti vezérlők annyiban mind protokollfüggetlenek, hogy *IPv4* és *IPv6* protokollokkal egyaránt működnek.

Hálózati házirend

Az elmélettel foglalkoztunk eleget, lássunk tehát egy valódi *SELinux* házirendet egy egyszerűbb hálózati alkalmazáshoz. Mivel helyszűkében vagyunk, az igazi hálózatkezelés pedig sosem egyszerű, csak egy egyszerű *TCP* visszhang-ügyfél számára fogunk házirendet készíteni.
Az ügyfél forráskódja a cikkhez tartozó internetes források között szereplő weboldalon érhető el. Rövid annyit róla, hogy létrehoz egy *TCP* foglalatot, kapcsolódik a távoli állomás visszhang kapujához, kiír valami szöveget, majd visszaolvassa.
Saját munkaállomásom két *Ethernet* csatolóval rendelkezik, a példában az `eth0` egy intranetbe tartozik, és a kiszolgáló, melyhez csatlakozok, a ***10.3.1.2*** IP-címet viseli.
Amit a biztonsági házirendtől elvárunk:

- Az ügyfél az operációs rendszernek csak a valóban szükséges erőforrásaihoz férjen hozzá.
- Az ügyfél kommunikációs lehetőségei csak a ***10.3.1.0/24*** alhálózatra eső `inetd` kiszolgálókra és az `eth0` csatolóra terjedjenek ki.

Házirend

Az alábbiakban egy a fentieknek megfelelő, megjegyzésekkel ellátott házirend szerepel. Használatához telepíteni kell a terjesztésünkhöz készült *SELinux* házirend forráscsomagokat, majd legfelső szintű könyvtárába kell lépnünk (saját gépemen ez a */etc/selinux/strict/src/policy*).
Hozzuk létre a domains/program/echoclient.te nevű fájlt, majd adjuk hozzá az *1. kódrészletben* szereplő házirendbejegyzéseket.
A *net_contexts* fájlhoz a következő címkemegadásokat kell hozzáadni:

```
# eth0 címke
netifcon eth0 system_u:object_r:netif_intranet_t
          system_u:object_r:unlabeled_t
  # A belső hálózat címkézése.
```

1. kódrészlet echoclient.te

```
# Egyszerű visszhangügyfél (echoclient) házirend
# a linuxvilágos cikkhez
# Fájl: domains/program/echoclient.te

# Az echoclient_t típus megadása tartományként
type echoclient_t, domain;

# Az echoclient_exec_t megadása futtatható típusú
# fájlként.
type echoclient_exec_t, file_type, exec_type;

# Ez egy makró, ez fogja engedélyezni
# a megfelelően címkézett futtatható fájlnak
# az átlépést az echoclient_t tartományba
# a staff_t tartományból.
domain_auto_trans(staff_t, echoclient_exec_t,
                  echoclient_t)

# Megadjuk, hogy mely szerepek léphetnek be az
# echoclient_t
# tartományba.
role staff_r types echoclient_t;

# Ez a makró teszi lehetővé a tartománynak
# a megosztott
# könyvtárak használatát.
uses_shlib(echoclient_t);

# Azoknak az engedélyeknek a biztosítása, amelyek
# a program futtatásához staff_t-ként, SSH-n
# keresztül való bejelentkezésnél szükségesek,
# így válik lehetővé a hibakereső és -jelző
# üzenetek kiírása a felhasználó termináljára.
allow echoclient_t sshd_t:fd use;
allow echoclient_t staff_devpts_t:chr_file {
                                getattr read write };

# Hálózati beállítások
# Ezek a tartomány által igényelt foglalat
# engedélyek.
# Érdemes megjegyezni, hogy csak a TCP
# foglalatokra érvényesek.
allow echoclient_t echoclient_t:tcp_socket {
            connect create read shutdown
            ➥write };
# Engedélyezzük a programnak, hogy TCP üzeneteket
# küldjön a visszhangkapu
# felé, illetve ilyeneket fogadjon tőle.
# Egy normál házirendben a kapu
# inetd_port_t címkét kap, és az inetd által
# kezelt kapuk csoportjába tartozik. A
# net_contexts fájlban szereplő házirendet
# módosítva a szabály
# egyetlen kapura is szűkíthető.
allow echoclient_t inetd_port_t:tcp_socket {
                                recv_msg send_msg };
# Az intranetes csatolón keresztül csak a TCP
#forgalmat engedélyezzük.
allow echoclient_t netif_intranet_t:netif {
                                tcp_recv tcp_send };

# A belső IP-címekkel csak TCP alapú
# kommunikációt
# engedélyezünk.
allow echoclient_t node_internal_t:node {
                                 tcp_recvtcp_send };
```

```
nodecon 10.3.1.0 255.255.255.0
        system_u:object_r:node_internal_t
```

A ***types/network.te*** fájlba kerülő sorok:

```
# A netif_intranet_t megadása hálózati
# csatoló típusként.
type netif_intranet_t, netif_type;
```

Fájlkörnyezet megadása a futtatható fájl számára egy új, ***file_contexts/program/echoclient.fc*** nevű fájlban:

```
# Alapértelmezett fájlkörnyezet a címkézéshez
/tmp/echoclient –
➥system_u:object_r:echoclient_exec_t
```

Fordítsuk le és töltsük be a házirendet:

```
$ make load
```

Ezzel a házirend elkészült. Látszólag rengeteg munka volt vele, ám ha már otthonosan mozgunk a különféle házirend-fájlok között, és jobban megismertük a szükséges házirend-bejegyzések típusait, minden könnyebbé válik. Ekkor már az olyan eszközöket is könnyebben tudjuk majd használni, mint például az `audit2allow`, amely a napló tiltási üzenetei alapján engedélyező szabályokat hoz létre. A mindennapi házirendkészítéshez jobb valamilyen magasabb szintű, grafikus eszközt használni, jelen esetben azonban a dolgok működésének lépésről lépésre való bemutatása volt a cél.

Próba

Fordítsuk le és lássuk el címkével a futtatható ügyfél-programot:

```
$ make echoclient
cc      echoclient.c    -o echoclient

$ restorecon /tmp/echoclient
```

Ellenőrizzük a címke helyességét:

```
$ getfilecon /tmp/echoclient
/tmp/echoclient
➥system_u:object_r:echoclient_exec_t
```

Erre a célra az `ls -Z` parancsot is használhatjuk.
Lássuk, működik-e a rendszer. Rootként jelentkezzünk be SSH-n keresztül staff_r szerepbe, majd:

```
$ id -Z
root:staff_r:staff_t
$ /tmp/echoclient 10.3.1.2
Sending message: 'Hello, cliche'
Received message: 'Hello, cliche'
```

Működik!
A házirendet auditallow szabályokkal bővítve azt is figyelemmel követhetjük, hogyan folyik az engedélyek megadása.
Ellenőrizzünk is néhány házirendbéli szabályt, vajon valóban működik-e.

1) Próbáljunk egy az intraneten kívülre eső IP-címmel kapcsolatot létesíteni. A címet helyileg irányítsuk, így nem fordulhat elő, hogy véletlenül az internet felé küldünk el egy csomagot:

```
$ ip ro add 196.40.74.92 via 10.3.1.2 dev eth0
$ /tmp/echoclient 196.40.74.92
```

A program *TCP* időtúllépést kap, és csomag küldésekor az alábbi, tiltásról értesítő naplóüzenet jön létre:

```
avc:  denied  { tcp_send } for  pid=10831
  exe=/tmp/echoclient saddr=10.3.1.1 src=32822
  daddr=196.40.74.92 dest=7 netif=eth0
  scontext=root:staff_r:echoclient_t
  tcontext=system_u:object_r:node_t
  tclass=node
```

Mint vártuk, az echoclient_t tartomány nem kapott engedélyt arra, hogy *TCP* csomagot továbbítson */node_t/* csomópontnak (ez az alapértelmezett általános csomópont környezet).

2) Próbáljunk másik csatolón keresztül kommunikálni. A visszhang kiszolgáló IP-címét irányítsuk a hurok felületen keresztül, így a csomagok erre kerülnek elküldésre:

```
  $ ip ro add 10.3.1.2 via 127.0.0.2 dev lo

  $ /tmp/echoclient 10.3.1.2
avc:  denied  { tcp_send } for pid=10828
  exe=/tmp/echoclient saddr=10.3.1.1 src=32821
  daddr=10.3.1.2 dest=7 netif=lo
  scontext=root:staff_r:echoclient_t
  tcontext=system_u:object_r:netif_lo_t
  tclass=netif
```

Most is helyes eredményt kaptunk. Az `echoclient_t` tartomány nem kapott engedélyt arra, hogy `netif_lo_t` hálózati csatolón keresztül továbbítson csomagot.
Az `echoclient` program a házirend értelmében rendkívül szűkre szabott jogosultságokkal fut. Amit nem engedtünk meg kifejezetten, az tilos számára. A programban esetlegesen felmerülő hibák, a felhasználó hibázásának vagy rosszindulatának hatása a házirend révén erőteljesen korlátozható.
Egyszerű példánk alapján világossá válik, hogy ***SELinux*** házirenddel hogyan lehet elérni a hálózati biztonság terén kitűzött célokat. Egy valós környezetben alkalmazott házirendnek számos különleges lehetőséget kell biztosítania, ezekkel a helyszűke és az érthetőség miatt nem foglalkoztunk, de példaként érdemes az ***ICMP***-üzenetek és a ***DNS***-lekérdezések engedélyezését megemlíteni. Mindenkinek javaslom a saját terjesztéséhez készült házirendforrások tanulmányozását, illetve a grafikus házirendkészítő alkalmazások kipróbálását.

Jövőbeli fejlesztések

SELinux alatt valószínűleg meg fog valósulni a címkézett hálózatkezelés valamilyen formája. Ennél a megoldásnál maga a hálózati forgalom kerül címkézésre – ilyesmire leginkább bizalmas adatokat kezelő katonai és kormányzati rendszerekben láthatunk példát. Az ***SELinux*** egy korábbi változata IP-beállításokat használt a csomagok címkézésére, ám ezt a megoldást kivették belőle, még mielőtt a rendszermag fő ágába bekerült volna, ugyanis a szükséges csatlakozási pontok túlságosan mélyrehatók voltak. Egy másik megoldás lenne az ***SELinux*** és az ***IPSec*** egybeépítése, és a ***biztonsági társulások (Security Associations, SA)*** címkézése a csomagok helyett. Egy adott ***SA***-n érkező csomagot implicit módon lehetne címkézni az ***SA*** környezetével. A korábbi ***Flask Project*** keretein belül már készült minderre egy prototípus, az itt szerzett eredmények kiváló útmutatóként szolgálhatnak.
Az ***SELinux*** szorosabb egybeépítése más hálózatbiztonsági összetevőkkel, például a titkosítással vagy a tűzfalakkal, további kutatások tárgya.

Köszönetnyilvánítás

Köszönettel tartozom ***Russell Cokernek*** írásom átnézéséért és értékes megjegyzéseiért.

Linux Journal 2005. január, 129. szám

James Morris (jmorris@redhat.com) az ausztráliai Sydney városából származó rendszermag-programozó, jelenleg Bostonban, a Red Hatnél dolgozik. Az SELinux, a Networking és a Crypto API rendszermag karbantartója; LSM fejlesztő és az Emeritus Netfilter Core Team tagja.

KAPCSOLÓDÓ CÍMEK

- www.nsa.gov/selinux
- www.lurking-grue.org/gettingstarted_newselinuxHOWTO.html
- people.redhat.com/kwade/fedora-docs/selinux-faq-en
- www.gentoo.org/proj/en/hardened/selinux

Maradjunk naprakészek terjesztésünk biztonsági frissítéseivel

A kezdő Linux rendszergazdák számára a programok naprakészen tartása az első lecke. Jeremy ennek a népszerű frissítő eszközök segítségével való elvégzését tárgyalja, kattintásról kattintásra.

Ha a *Linuxot* asztali vagy kiszolgáló környezetben meg akarjuk őrizni első számú játékosnak, akkor ügyelnünk kell arra, hogy a biztonsági foltok segítségével naprakészek maradjunk. Hiába tesszük meg a szükséges biztonsági intézkedéseket a hálózat és a vas szintjén, ha egyetlen apró rés miatt az egész rendszer sebezhetővé válik. Minden felhasználónak, legyen szó akár üzleti, akár nonprofit, akár otthoni használatról, tisztában kell lennie azzal, rendszerét és alkalmazásait miként frissítheti, és rendszeresen meg is kell ezt tennie.

A rendszer sértetlenségének megőrzéséhez két lépésre van szükség: tudni kell, mikor kell frissíteni, és valóban végre is kell hajtani a frissítést. Az első feladatot a terjesztéshez tartozó biztonsági témájú levelezőlista figyelésével teljesíthetjük. A második elvégzésére számos mód kínálkozik, végrehajtásához grafikus és parancssori eszközöket egyaránt használhatunk. Néhány terjesztés önműködő frissítő segédeszközt is tartalmaz, melynek segítségével könnyebben figyelemmel követhetjük rendszerünk állapotát.

Az adott alkalmazás újabb változatának telepítését frissítésnek vagy foltozásnak is nevezhetjük, a két művelet lényege ugyanaz. Ugyanakkor arra is érdemes figyelni, hogy a frissítés során ne telepítsünk olyan csomagváltozatot, amelyet eredetileg nem akartunk feltenni. A csomagok fejlesztői változatai általában külön sorszámsorozatból kapják változatszámukat. Ha a változatszám túlságosan különbözik, vizsgáljuk meg, milyen egyéb változatokat tudunk elérni.

Írásomban a linuxos rendszerek naprakészen tartására használható eszközök grafikus és parancssori változatával egyaránt foglalkozok. Részletesebben a ***Debian 3.0 (Woody)***, a ***Mandrake 10.0***, a ***SuSE 9.1*** és a ***Fedora Core 2*** terjesztésről lesz szó.

Mikor frissítsünk?

Honnan tudhatjuk, hogy mikor kell frissítést végeznünk? A legjobb módszer az, hogy feliratkozunk a terjesztésünkkel kapcsolatos biztonsági értesítők levelezőlistájára. Az internetes források között az itt tárgyalt terjesztésekre és a megfelelő biztonsági levelezőlistákra vezető hivatkozások egyaránt megtalálhatók. Ezek általában kis forgalmú listák, általuk a biztonsági jellegű foltokról vagy frissítésekről értesülhetünk. Általában közvetlen hivatkozásokat is tartalmaznak a frissített csomagok letöltéséhez, a csomagok helyességének megőrzését pedig ***MD5*** ellenőrző összegekkel segítik. Ennél a módszernél a csomagokat kézzel kell telepítenünk, és a függőségeket is magunknak kell feloldanunk. A frissítések megjelenéséről úgy is értesülhetünk, hogy írunk egy a frissítéseket rendszeresen lekérdező parancsfájlt. ***SuSE 9.1*** és ***Fedora Core 2*** alatt meglévő programjainkat könnyedén, grafikus felületről tudjuk frissíteni. Debian és Mandrake alá szintén léteznek jól áttekinthető grafikus eszközök, ezeket parancsfájlokkal arra is rávehetjük, hogy az éjszaka közepén töltsék le a frissítéseket, amelyek telepítését a számunkra kényelmes időpontra tolhatjuk el. Muszáj megemlítenem a programok felügyelet nélküli frissítésének veszélyeit. Nálam például az ***Apache*** webkiszolgáló pontos beállítása komoly munkát jelent. Frissítéskor mindig szembesülök a kérdéssel: szeretném-e felülírni meglévő beállító fájljaimat? A legtöbbször a `diff` segítségével átnézem, milyen változtatásokra számíthatok, és a beállító fájl felülírását általában nem engedélyezem. Ha létfontosságú alkalmazásokat is futtatunk, mindig fordítsunk kellő gondot a változtatások rögzítésére, a beállító fájlokról pedig készítsünk biztonsági mentéseket.

RPM alapú terjesztések

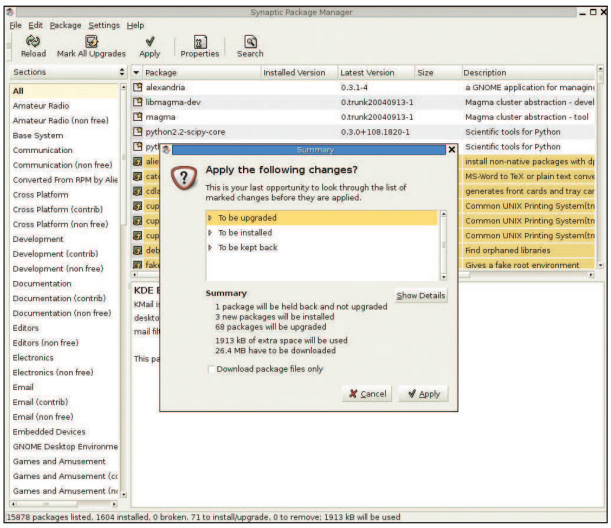
Az ***RPM*** parancssori eszköz megbízható, kézi módszert kínál a biztonsági frissítések telepítésére. Az rpm parancsnak rengeteg kapcsolója van, ám a csomagok frissítéséhez csupán a következő utasítást kell kiadnunk:

```
# rpm -Uv csomag.rpm
```

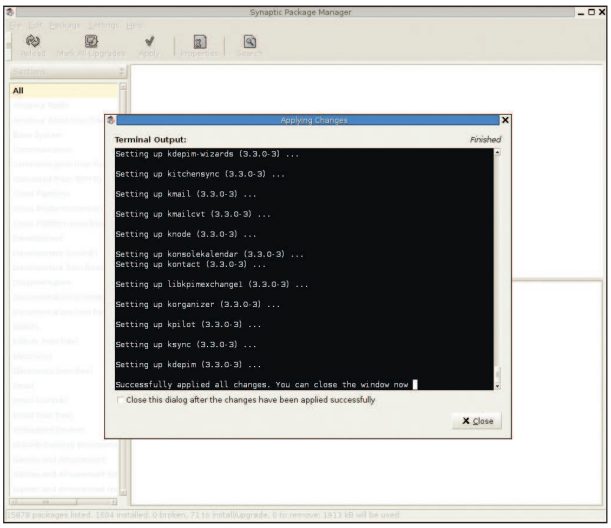
Az ***RPM*** fájl helyi fájl is lehet, de ***FTP***-n vagy ***HTTP***-n keresztül elérhető fájlt is megadhatunk. Ha a biztonsági levelezési listán közvetlen URL-eket kapunk a frissített csomagokra, akkor a parancssori frissítés roppant egyszerűvé válik. Az `rpm` parancssori eszközről az ***RPM*** webhelyen vagy a megfelelő `man` oldalon találunk további tájékoztatást.

Debian alapú terjesztések

A ***Debian*** és az egyéb, ***Debianra*** épülő terjesztések csomagkezelője a `dpkg`. Neve a ***Debian GNU/Linux package***



1. ábra A módosítandó alkalmazások a Synaptic listájában



2. ábra A Synaptic az összes frissítés telepítése után

manager, ***Debian GNU/Linux csomagkezelő*** kifejezésből származik. A `dpkg` ***GYK*** szerint a név mára elveszítette jelentőségét, hiszen a `dpkg`-t nem ***Debian*** rendszerek is használják, sőt, a ***Linuxtól*** eltérő operációs rendszerek alá is létezik. Ez a csomagkezelő lényegében aládolgozik például a ***APT***-nek (***Advanced Packaging Tool, fejlett csomagkezelő eszköz***) és a grafikus eszközöknek, például a ***Synapticnek***. Az ***RPM***-hez hasonlóan a dpkg is milliónyi parancssori kapcsolót ismer, ám mi most csak a frissítésre használhatóval foglalkozunk, amelynek alkalmazásakor a kiadandó parancs a következőképpen fest:

```
# dpkg -i csomag.deb
```

A `-i` kapcsoló a csomag telepítésére utasítja a `dpkg`-t. Ha a csomagból korábbi változat is telepítve van, a `dpkg` először eltávolítja azt, majd felteszi az újabb változatot. Az `rpm`-től eltérően a `dpkg`-nak a csomag telepítés előtti letöltéséhez a wget-re vagy curl-re is szüksége van.

Debian 3.0 (Woody)

Debian alatt csomagkezelő feladataink túlnyomó részét valószínűleg az ***APT*** segítségével fogjuk elvégezni. Az ***APT*** egy listát vezet az elérhető csomagokat tároló forrásokról. Ha egy forrás listájában újabb csomagváltozat szerepel, akkor az ***APT*** letölti a csomagot, majd átadja a vezérlést a `dpkg`-nek. Először ellenőrizzük, hogy a biztonsági frissítések forrása szerepel-e a sources.conf fájlban. A fájlban a következő sornak kell szerepelnie:

```
deb http://security.debian.org/ stable/updates main
```

A `stable` szó helyett lehet, hogy a `woody` szerepel – teljesen mindegy. A ***sources.conf*** fájl átnézése-átírása után frissítenünk kell a rendelkezésre álló csomagok listáját. A frissítés és a foltozás végrehajtásához az apt-get parancsot kétszer kell kiadnunk:

```
# apt-get update
# apt-get upgrade
```

Ekkor csak azoknak a csomagoknak a frissítésére kerül sor, amelyek más csomagok módosítását nem igénylik. Ha a függőségekkel is rendelkező csomagokat is frissíteni akarjuk, a következő parancsokat kell kiadnunk:

```
# apt-get update
# apt-get -u dist-upgrade
```

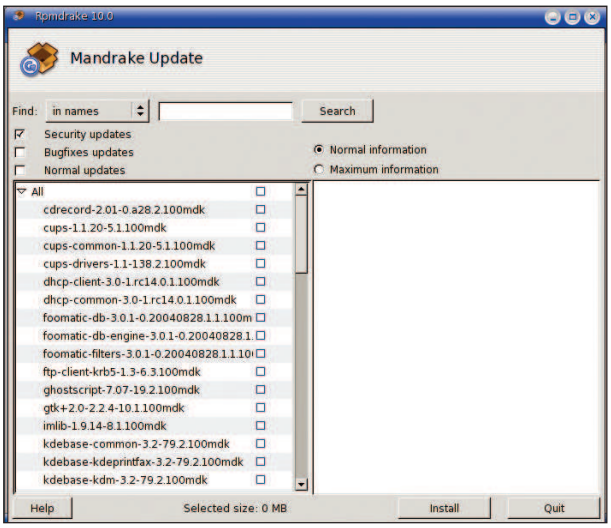
A `-u` kapcsoló segítségével pontosan láthatjuk, hogy mely csomagok kerülnek frissítésre, újonnan telepítésre vagy eltávolításra. A fenti parancsokat `crontab` segítségével is futtathatjuk, valamint arra is rávehetjük gépünket, hogy mindig töltse le a legújabb csomagokat, de ne telepítse őket. A `crontab` fájlba az alábbihoz hasonló parancsot kell beírnunk:

```
(apt-get update && apt-get -dy upgrade)
➥| mail -s "`hostname` frissítés" root
```

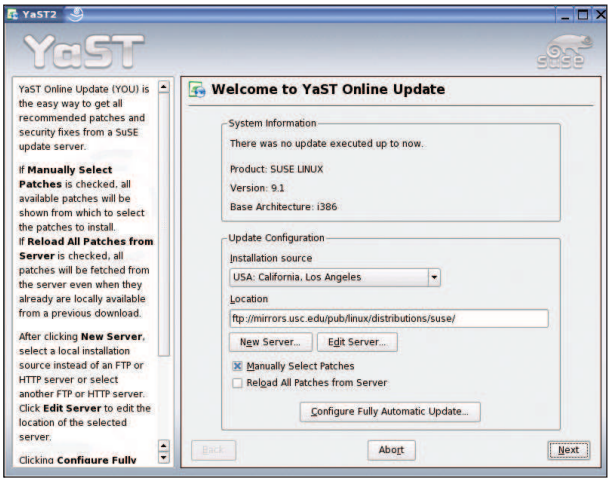
A parancs letölti a legújabb csomagok listáját, és ha ez sikeres, akkor letölti a frissítést igénylő csomagokat is. Az eredményről elektronikus levélben tájékoztatja a root felhasználót. Szükség szerint használjuk saját felhasználónevünket vagy levélcímünket. Ha értesítő levelet kaptunk a frissítésekről, a következő parancsot kell kiadnunk:

```
# apt-get upgrade
```

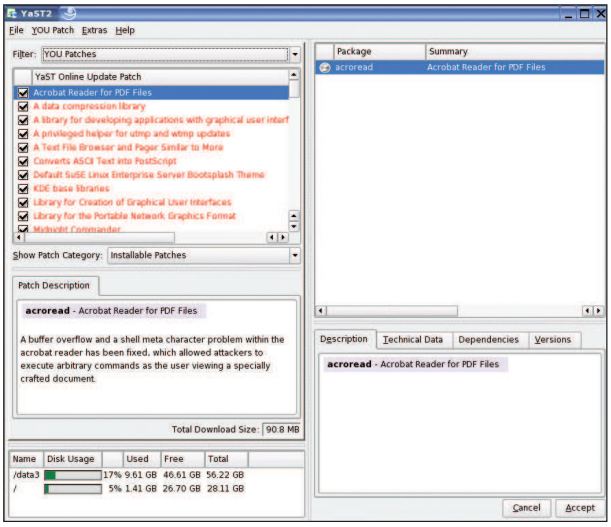
Ekkor megtörténik a korábban letöltött csomagok telepítése, melynek folyamatát konzolról vagy terminálról tudjuk figyelemmel kísérni. Bizonyos csomagok telepítésekor további beavatkozásra is szükség lehet, ezért érdemes lehet tartózkodni a teljes mértékben automatizált módszer használatától.
A grafikus oldalt szemlélve a ***Debiant*** használók számára a ***Synaptic*** teljes értékű felületet biztosít a dpkg-hoz.
A ***Synaptic*** futtatásához ablakkezelőnk ***Debian*** menüjéből az ***Apps›System›Synaptic Package Manager (Alkalmazások› Rendszer›Synaptic csomagkezelő)*** parancsot kell választanunk. A ***Synaptic*** működése sokban hasonlít az ***APT***-éhez. Ha frissíteni szeretnénk a rendelkezésre álló csomagok listáját, kattintsunk az ablak bal felső részén található ***Reload (Újratöltés)*** gombra. Egy ablakban megjelenik a frissített csomaglista tartalma. Amikor a ***Synaptic*** végzett a csomaglisták letöltésével, az összes rendelkezésre álló frissítést megtekinthetjük. A frissítést igénylő csomagokat egy zöld négyzet és egy felfelé mutató nyíl jelzi. Az újonnan elérhető csomagoknál a négyzet-



3. ábra Az rpmdrake listája a rendelkezésre álló csomagfrissítésekről



4. ábra A YaST2 Online Update tükörválasztó folyamata



5. ábra A YaST2 Online Update listája az elérhető csomagfrissítésekről

ben egy sárga csillag látható. A már telepített csomagoknál a négyzet zöld, a még nem telepítetteknél pedig fehér színű.

Ha az összes csomagfrissítést le szeretnénk tölteni és fel szeretnénk telepíteni, akkor kattintsunk az ***Apply (Alkalmaz)*** gombra. Az ezután megjelenő ablakból megtudhatjuk, hogy mely csomagok kerülnek frissítésre, telepítésre, visszatartásra vagy eltávolításra (*1. ábra*). A visszatartott csomagok esetében további, részletesebben meg nem adott függőségekkel kell számolni. Ha rákattintunk az ***Apply (Alkalmaz)*** gombra, megkezdődik a frissítések letöltése. A letöltés után a frissítések telepítését egy terminálszerű szövegterületen követhetjük, az esetleges kérdésekre is itt adhatunk választ. Ha végeztünk, kattintsunk a ***Close (Bezár)*** gombra. (*2. ábra*)

Mandrake 10.0

A ***Mandrake 10.0*** telepítésekor az első bejelentkezés előtti záró lépések egyike a fontosabb frissítések lekérdezése. Ha nulláról indulva telepítjük a terjesztést, akkor mindenképpen szakítsunk időt erre a műveletre. Mit tegyünk azonban, ha Mandrake rendszerünk már telepítve van, és be kellene foltoznunk egy biztonsági rést?

A ***Mandrake 10.0*** használóinak egy csinos, grafikus felületű csomagkezelő alkalmazás áll rendelkezésére, ez az rpmdrake. A ***KDE*** csillag menüjére kattintva, majd a ***System›Configuration›Packaging›Mandrake Update (Rendszer›Beállítások›Csomagkezelés›Mandrake frissítés)*** parancsot választva indíthatjuk el, de a parancssorból root felhasználóként az `rpmdrake` parancsot is kiadhatjuk. Válaszoljuk meg a felbukkanó kérdéseket, és máris megjelenik a biztonsági okból frissítést igénylő csomagok listája. (*3. ábra*) Ha mindegyiket frissíteni szeretnénk, kattintsunk az ***All (Mind)*** sorban található négyzetre, majd az ***Install (Telepít)*** gombra – aztán bontsunk egy sört, és dőljünk hátra. Az összes frissítés letöltése és telepítése után párbeszédpanel értesít a művelet sikeres elvégzéséről. Ilyen egyszerű az egész.

A parancssoros ***urpmi*** csomag nálam a ***Mandrake 10.0*** alapelemeként települt. Az `urpmi` működése sokban hasonlít az ***APT***-éhez, és szintén lehetővé teszi több forrás használatát a csomagok frissítésére. A forrás lehet CD-lemez, helyi ***RPM*** könyvtár, vagy ***FTP*** vagy ***HTTP*** alapú internetes erőforrás. A biztonsági javítások telepítéséhez az alábbihoz hasonló parancsot kell futtatnunk:

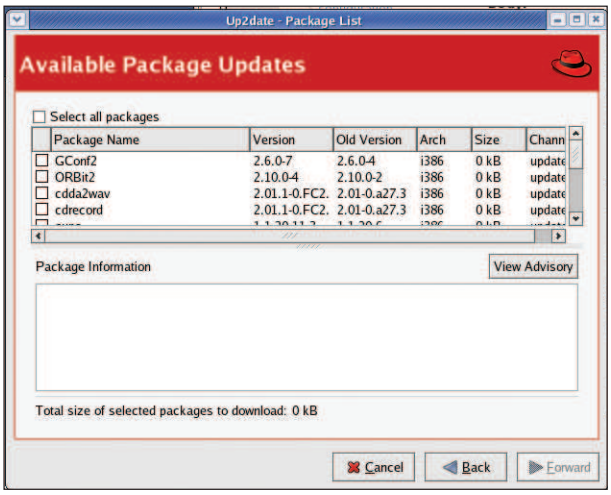
```
# urpmi.addmedia –update updates
➥ftp://pelda.com/Mandrake10.0/RPMS
➥with ../base/hdlist.cz
```

Ezzel egy ***FTP***-tükör biztonsági frissítéseit adhatjuk hozzá a források listájához. Természetesen az ***ftp://*** ***URL***-t egy valós hely címével kell helyettesítenünk. Az ***Easy urpmi*** weboldalról könnyen átlátható, webes felületről választhatjuk ki a hozzánk legközelebb eső tükörhelyet, gépünk típusát és azt, hogy mely forráskészletekből kívánjuk letölteni a frissítéseket.

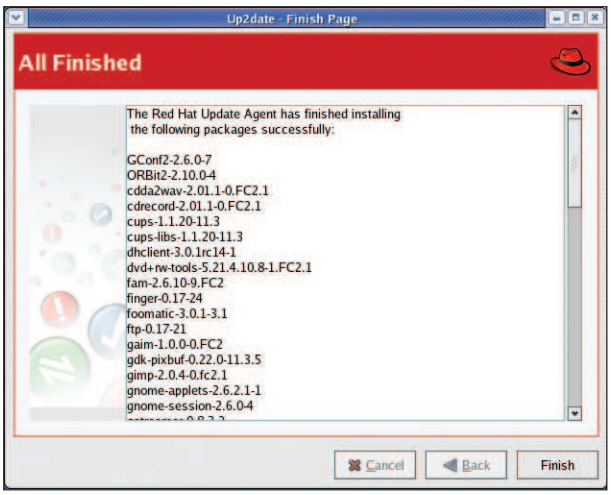
Ha le szeretnénk tölteni az elérhető csomagok listáját, majd telepíteni szeretnénk az összes csomagfrissítést, a következő két parancsot kell kiadnunk:

```
# urpmi.update -a
# urpmi –auto-select
```

Ezután megkezdhetjük a frissített csomagok és a függőségek miatt szükségesnek ítélt csomagok telepítését.



6. ábra Az Up2date listája a rendelkezésre álló csomagokról



7. ábra Az Up2date végzett a letöltéssel és a telepítéssel

SuSE 9.1

A ***SuSE 9.1*** hasonló módszert kínál a frissítések telepítésére, melyben a ***YaST2 Online Update (YOU)*** grafikus eszköz segít bennünket. A ***SuSE*** ikonra, majd a ***System›YaST (Rendszer›YaST)*** parancsra kattintva indíthatjuk el. Írjuk be a root jelszavát, majd kattintsunk a ***Software (Szoftver)*** és az ***Online Update (Online frissítés)*** parancsra. Válasszuk ki a telepítési forrást, vagy adjunk hozzá kézzel egy új kiszolgálót. (*4. ábra*) A ***YOU***-t úgy is beállíthatjuk, hogy minden nap megadott időpontban önműködően töltse le és/vagy telepítse a frissítéseket. A ***Next (Tovább)*** gombra kattintva letölthetjük azokat az adatokat, amelyek alapján felmérhető a frissítést igénylő csomagok köre. A lista frissítése után megkapjuk a csomagok listáját, a foltok leírását és a lemezhely-használattal kapcsolatos adatokat. (*5. ábra*) A foltok listájában vörös vonalak jelzik a biztonsági frissítéseket, kék vonalak az ajánlott frissítéseket és feketék az elhagyhatókat. A rendszerfrissítést az ***Accept (Elfogad)*** gombbal indíthatjuk el. Ha a frissítés befejeződött, kattintsunk a ***Finish (Befejez)*** gombra, ezt követően még sor kerül néhány rendszerszolgáltatás beállítására. A ***YOU*** mellett, ha akarjuk, a parancssorból az `rpm` parancsot is használhatjuk.

Fedora Core 2

A ***Red Hat*** frissítő ügynöke, az `up2date` számos ***Red Hat*** terjesztésben szerepelt, és a ***Fedora Core 2***-ben is megtalálható. Ha ***Fedora Core 2*** alatt le szeretnénk kérdezni az elérhető frissítéseket, akkor kattintsunk az egér jobb gombjával a rendszertálcán található piros felkiáltójelre, majd válasszuk a ***Check for updates (Frissítések lekérdezése)*** parancsot. A legújabb frissítések letöltését és telepítését a piros felkiáltójelre az egér jobb gombjával kattintva és a ***Launch up2date (up2date indítása)*** parancsot választva indíthatjuk el. Adjuk meg az alapbeállításokat. Az `up2date` első futtatásakor a program megkérdezi, hogy telepíteni akarjuk-e a ***Red Hat GPG*** kulcsaláírását. Én a saját gépemen igennel válaszoltam. A ***Channels (Csatornák)*** menüben két csatornára vagy frissítésforrásra iratkozhatunk fel, ezek a ***fedora-core-2*** és az ***updates-released-fc2***. Az `up2date` csatornái hasonlók az ***APT*** vagy az `urpmi` forrásaihoz. Következőként megadhatjuk az átlépni kívánt csomagokat. Nálam az eleve megjelenő csomag egy rendszermagfrissítés volt. A ***Forward (Tovább)*** gombra kattintva megkapjuk az elérhető frissítések listáját. (*6. ábra*) Ha az összes frissítést telepíteni akarjuk, akkor jelöljük be a ***Select all packages (Minden csomag kijelölése)*** jelölőnégyzetet.

Újra a ***Forward*** gombra kattintva elindíthatjuk a csomagok letöltését. Ezen a ponton megismételném a sörözéssel és a pihenéssel kapcsolatos ajánlatomat. Ha a letöltés befejeződött, újfent kattintsunk a ***Forward*** gombra, ezzel elindítjuk a telepítést. Ha a telepítés is véget ért, a program megjeleníti, hogy pontosan mely csomagokat és milyen számú változatban telepítette. (*7. ábra*)

A ***Fedora Core 2*** ugyancsak ***RPM*** alapú, vagyis terminál alól használhatjuk az rpm parancsot is.

Egy további ismert csomagkezelő felület a ***Yellow dog Updater Modified***, röviden ***Yum***. A ***Yum*** sokban hasonlít az ***APT***-hez, ám sokban el is tér tőle, a különbségeket a szerző részletesen ismerteti a ***Yum*** weboldalon. A ***Yum*** az `urpmi` és az ***APT*** esetében látottakhoz hasonló csomagforrásokkal dolgozik, és a csomagok telepítését az ***RPM***-re bízza. Az ***anaconda*** telepítő ***Python*** kötéseket használ az ***RPM*** elérésére, így a ***Python*** támogatásának fennmaradására bátran számíthatunk.

Összefoglalás

Van egy mondás a baseballban: „Annyira vagy jó, amennyire az utolsó ütésed.” A számítógépek világában ezt úgy fogalmazhatnánk meg, hogy egy gép annyira biztonságos, amennyire az utolsó frissítés révén azzá vált. Egy jó tűzfal vagy egy mágneskártyás beléptetős bejárat kiváló kezdő lépés a biztonság megteremtése felé, ám ne feledjük, ha elmulasztjuk a frissítések telepítését, és elavult ***Apache***- vagy ***OpenSSH***-változatot futtatunk, teljes rendszerünk veszélybe kerülhet.

Linux Journal 2005. január, 129. szám

Jeremy Turner több mint öt éve használja a Linuxot. Szenvedélye, hogy segítsen másokat a nyílt alkalmazások megismerésében. PHP-ben programoz, egy kórusban első tenor, rengeteg baseballt néz, elektronikus leveleit pedig rendszeresen olvassa (jeremy@linuxwebguy.com).

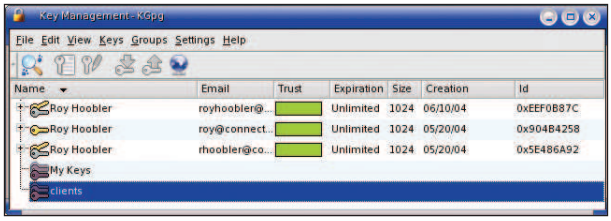
Levéltitkosítás egyetlen kattintással

A titkosított levelek küldését, fogadását és ellenőrzését grafikus eszközökkel végezve a biztonságos levelezés mindennapjaink természetes részévé válhat.

Jómagam egy hordozható gépet használok ***Linuxszal***, és nem szeretném, hogy mások is el tudják olvasni a leveleimet, ha netán rossz kezekbe kerülne. Leveleimet figyelik is, és az se tetszene, ha a rendszergazda belelátna a személyes dolgaimba. A ***GnuPG*** kiváló titkosítási lehetőséget kínál, és bárki számára elérhető. A ***KDE KGPG*** és ***KMail*** alkalmazásával a dolgok még egyszerűbbekké válnak. Írásomban a ***KGPG*** elektronikus levelek és fájlok titkosítására való használatát tárgyalom. Lehet, hogy a téma kicsit összetettnek fog tűnni, ám mire a végére érünk, bárki üzemképes rendszerrel rendelkezhet – ráadásul mindehhez elég lesz egy óra. Ha valakinek kérdése maradna, nyugodtan írjon a cikkben szereplő címemre, legalább ki tudja próbálni új, biztonságos levelezőrendszerét.

Mi az a GnuPG?

A ***Gnu Privacy Guard (GnuPG)*** az ***OpenPGP*** szabvány egy megvalósítása. A szabvány ***Philip Zimmerman*** munkája nyomán és ***PGP (Pretty Good Privacy)*** programjából fejlődött ki. A ***PGP*** 1991 tájékán jelent meg, jelenleg zárt alkalmazás. Az ***OpenPGP*** szabványok 1997-ben készültek el, a ***GnuPG 1.0***-s változata pedig 1999-ben látott napvilágot.
A ***GnuPG*** teljes egészében parancssori program, és meglehetősen bonyolult a kezelése. Szerencsére léteznek a használatát megkönnyítő eszközök, ezek közül a ***KDE***, a ***KGPG*** és a ***KMail*** szerepéről lesz szó.
A ***GnuPG*** és a ***PGP*** egymással kompatibilisek. Aki már használja a ***PGP***-t, azon belül is az ***IDEA*** algoritmust, annak némi munkával jár az áttérés, a többiek számára viszont semmilyen nehézséget nem okoz. Ha valakinek ***PGP 2.x*** változattal kell adatcserét folytatnia, esetleg ezt a programot szeretné lecserélni, akkor tanulmányozza át a cikkhez tartozó forrásokat, a ➲ www.linuxjournal.com/article/7863 címen.
Az adatvédelmi és adatbiztonsági házirendek összeállításához és betartatásához minden szervezetben vasfegyelemre van szükség. Amikor katonai hírszerzőként dolgoztam, a biztonságot nagyon komolyan vették. Ha valaki nem követte a házirendeket, annak súlyos következményei voltak. Minden szervezetben ki kell jelölni egy biztonsági igazgatót, akinek megfelelő hatalmat kell biztosítani az irányelvek követésének ellenőrzésére. Mint minden jól kidolgozott gyakorlati megoldásnál – mint például a ***CVS*** használata a kódok tárolására –, ha az embereket megtanítjuk, rászoktatjuk az érzékeny adatok titkosítására, akkor az eljárás hétköznapivá válik. A szervezet méretétől függően a házirend hatályba léptetése egy-négy hét alatt végbe vihető.



1. ábra A kulcskezelő eszköz segítségével a GnuPG kulcsok között név és levélcím alapján is kereshetünk

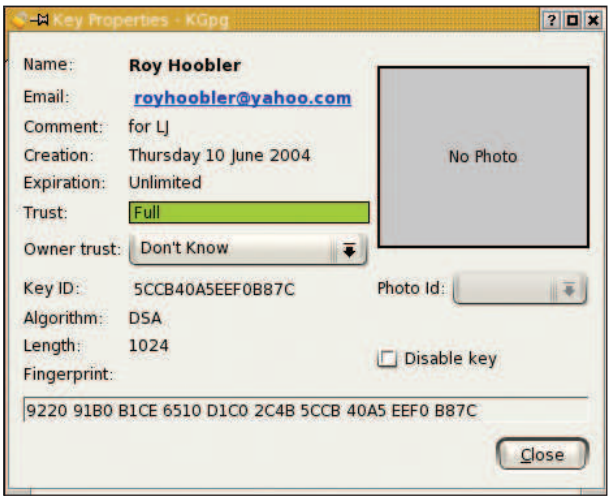
Saját kulcsunk előállítása

Példaként egy üzenetet fogok titkosítani, majd elküldöm azt munkahelyi címemről az ***rhoobler@comcast.net*** címemre. Mindkét fiókot ***KMail*** segítségével érem el. Ezután küldeni fogok egy titkosított választ a munkahelyi címemre. Természetesen előbb beállítom és elérhetővé teszem a ***comcast.net*** fiók kulcsait.
Miután telepítettük a ***KGPG***-t, a rendszertálcán kell hozzá tartoznia egy ikonnak. Ha nem lenne ilyen, akkor terminál alól, a `KGPG -k` paranccsal tudjuk elindítani. A `-k` kapcsolót ne hagyjuk el, ugyanis ez hozza elő a kulcskezelő felhasználói felületet. (*1. ábra*) A -k kapcsoló nélkül indítva a ***KGPG*** a háttérben, szolgáltatásként fut – ilyenkor jelenik meg a tálcán az ikonja. A felhasználói felület a tálcán található ikonra kattintva is megjeleníthető.
Első lépésként előállítom comcast.netes fiókom nyilvános és magánkulcsát. A kulcskezelőn belül válasszuk a ***Keys›Generate Key Pair (Kulcsok›Kulcspár előállítása)*** parancsot, amely egy meglehetősen egyszerű párbeszédpanelt jelenít meg. Ha az ***Expert Mode-ot (Szakértői mód)*** választjuk, a ***GnuPG*** terminálban indul. Most írjuk be nevünket, elektronikus levélcímünket, illetve a kívánt megjegyzést. A biztonsági házirendtől függően szükség lehet arra is, hogy a kulcsok lejáratát is beállítsuk.
A következő – rendkívül fontos – lépés a ***GnuPG*** jelszó megadása. Ha elhagyjuk, akkor egyetlen üzenetet sem tudunk majd megnyitni, a ***KGPG*** mindig jelszót fog kérni tőlünk, amikor bármit is megpróbálunk elolvasni. Várjunk

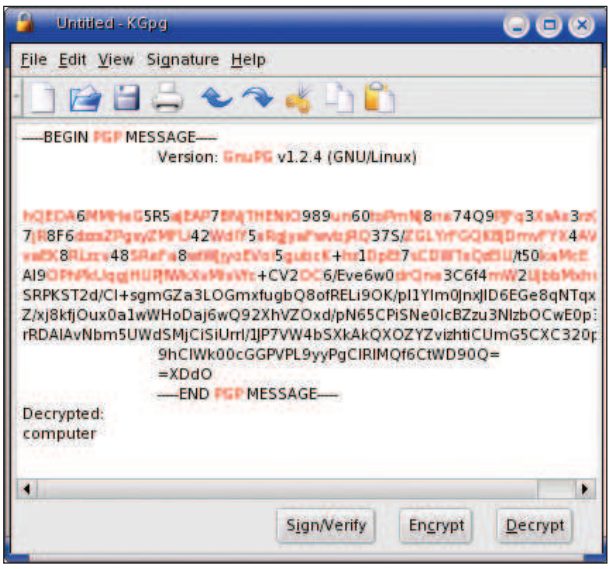
néhány másodpercet, amíg a *GnuPG* elkészíti kulcsainkat. A biztonság növelése érdekében javaslom egy ***visszavonási tanúsítvány (revocation certificate)*** elkészítését is (a fájl neve olyasféle legyen, mint az ***rhooblerrev.asc***). Ha ellopják vagy megtörik a gépünket, akkor ezt a tanúsítványt szétküldve értesíthetjük ismerőseinket nyilvános kulcsunk érvénytelenné válásáról.
Ennyi. Van nyilvános és titkos kulcsunk az ***rhoobler@comcast.net*** címhez. Két dolog van még hátra: a nyilvános kulcs és a titkos kulcs kimentése, utóbbi művelet elhagyható. A kulcsok kimentése külön-külön történik. A nyilvános kulcsnál az ***rhoobler.asc*** fájlnevet választottam, a titkos kulcs esetében pedig az ***rhooblerprivate.asc***-t. Vigyázzunk! Ha valaki megszerzi titkos kulcsunkat, és kitalálja jelszavunkat, akkor el tudja olvasni titkosított leveleinket, valamint a nevünkben tud titkosított, aláírt leveleket küldeni! A titkos kulcsot és a visszavonási kulcsokat kimentésük után írjuk CD-lemezre, helyezzük őket biztonságos helyre, majd töröljük a visszavonási (***rhooblerrev.asc***) és a titkos kulcsot (***rhooblerprivate.asc***) a merevlemezről.
A *GnuPG* a következő fájlokat helyezi el az egyes felhasználók *.gnupg* könyvtárába. A fájlokat csak az egyes felhasználók tudják írni és olvasni:

- *gpg.conf*: a *GPG* általános beállításai
- *pubring.gpg*: a nyilvános kulcsok listája
- *secring.gpg*: a biztonságos (magán) kulcsok listája
- *trustdb.gpg*: adatbázisfájl, az egyes személyek közötti megbízási viszonyokat adja meg

A kényelem kedvéért most az ***rhoobler@comcast.net*** nyilvános kulcsát egy alapértelmezett kulcskiszolgálóra mentem ki. A kulcskezelő eszköz segítségével válasszuk ki a kulcsot, kattintsunk rá az egér jobb gombjával, majd válasszuk az ***Export Public Key(s) (Nyilvános kulcs(ok) kimentése)*** parancsot. Újabb egyszerű párbeszédpanel jelenik meg, három lehetőséggel. Én az alapértelmezett kulcskiszolgálót választottam, majd az ***OK*** gombra kattintottam. A kulcskiszolgálót a ***Settings (Beállítások)*** menüben adhatjuk meg, alapesetben ez a ***subkeys.pgp.net***, ami – legalábbis nálam – mindig működött. Megtehetjük azt is, hogy fájlba mentjük ki a kulcsot, majd elektronikus levélben továbbítjuk, esetleg a weboldalunkon tesszük közzé. Abból, hogy bárki megismerheti a nyilvános kulcsunkat, semmi gondunk nem származhat, ám módot kell adnunk a kulcs eredetiségének ellenőrzésére – erről később lesz szó. Aki rendelkezik a nyilvános kulcsunkkal, az képes lesz fájlokat úgy titkosítani, hogy azokat csak mi tudjuk majd megnyitni.
Eljutottunk tehát odáig, hogy tudunk fájlokat titkosítani és titkosított leveleket küldeni. Mindennek azonban csak akkor van értelme, ha van kivel megosztanunk adatainkat. A próba kedvéért én átsétáltam a másik gépemhez, majd ugyanezeket a lépéseket végrehajtva létrehoztam a kulcsokat a munkahelyi levélfiókomhoz.
A következő művelet az ***rhoobler@comcast.net*** nyilvános kulcsának beemelése a kulcskiszolgálóról a kulcskezelő segítségével, majd a földgömb ikon vagy a menü ***File›Key Server Dialog (Fájl›Kulcskiszolgáló párbeszéd)*** parancsának kiválasztása. Adjuk meg az elektronikus levélcímet, majd emeljük be a kulcsot. Mielőtt használni kezdenénk a kulcsot, válasszuk ki a főablakban, majd válasszuk a menü ***Keys›Sign Keys (Kulcsok›Kulcsok aláírása)*** parancsát.
Ha nagyobb csoportot akartam volna összeállítani, akkor létrehozhattam volna egy saját kulcskiszolgálót, aláírhattam volna a kulcsokat és továbbadhattam volna őket. Egy másik megoldás, hogy mindenki elküldi levélben másoknak saját kulcsát, majd a való életben is találkoznak, amikor ellenőrzik és aláírják egymás kulcsát. Így jön létre a bizalmi hálózat. Én például aláírom ***Berci*** kulcsát, ő pedig ***Katiét***. Ha levelet kapok ***Katitól***, akkor kulcsát nyugodtan hozzáadhatom megbízási viszonyokat tároló adatbázisomhoz.
A kulcsoknak van egy ujjlenyomatuk, és ha nem vagyok biztos abban, hogy egy kulcs hiteles-e, akkor megnézhetem az ujjlenyomatát, majd felhívhatom a tulajdonosát, és segítségével ellenőrizni tudom a kulcsot. Az ujjlenyomat a kulcskezelő segítségével tekinthető meg, ehhez válasszuk ki a kulcsot, majd válasszuk a menü ***Edit Key (Kulcs szerkesztése)*** parancsát. (*2. ábra*)



2. ábra Ha ellenőrizni akarjuk egy levél hitelességét, akkor hívjuk fel a küldőt, és egyeztessük vele a kulcs ujjlenyomatát



3. ábra A vágólap visszafejtése

Fájlok titkosítása

A *KGPG*-t szinte észrevétlenül beépítették a *KDE*-be és a *KDE*-s alkalmazásokba. Ennek előnyeit leginkább a *Konqueror* böngésző használatakor élvezhetjük. A *KGPG* telepítése után kattintsunk az egér jobb gombjával a kívánt dokumentumra, majd az ***Actions (Műveletek)*** menüben létrehozhatjuk a dokumentum titkosított változatát. Lehetőség nyílik az eredeti példány megsemmisítésére is, aminek főleg akkor van értelme, ha a titkosítatlan változat megtartása valamiért nemkívánatos. A fájlok titkosításakor több kulcsot is kiválaszthatunk, így több különböző személy is el tudja olvasni a dokumentumot. Ha az eredeti fájlt megsemmisítjük, akkor ügyeljünk arra, hogy a titkosítás során saját kulcsunkat is kiválasszuk. Nálam, ha például csak az ***rhoobler@comcast.net*** kulcsot választom ki, akkor én leszek az egyetlen, aki vissza tudja fejteni a fájlt.

Elektronikus levél küldése

Végre készen állunk arra, hogy titkosított leveleket küldjünk. A *KMail* (vagy a *Kontact*) segítségével írjuk meg üzenetünket. Nálam a címzett az ***rhoobler@comcast.net***. Kattintsunk a *Lock* ikonra, vagy válasszuk az ***Options›Encrypt (Beállítások›Titkosítás)*** parancsot a menüből. Amikor a ***Send (Küldés)*** gombra kattintunk, egy párbeszédpanel jelenik meg. Ha nem látjuk a címzett kulcsát, kattintsunk a ***Refresh (Frissítés)*** gombra. Ha nem írtuk alá a kulcsot, nem fog megjelenni, ekkor vissza kell lépnünk, a kulcskezelő segítségével alá kell írnunk a kulcsot, majd rá kell kattintanunk a Refresh gombra. Fejezzük be az üzenetet, majd kattintsunk a *Send* gombra.
A *KMail* esetében az üzenetek visszafejtése beépített szolgáltatás. Ha titkosított üzenetet kapunk, csak be kell írnunk a jelszót, és máris megnyílik az üzenet. Ha titkosított üzenetet küldünk, és a címzett címe szerepel a kulcsgyűrűnkben, akkor a levél titkosítása és elküldése önműködően történik. Az üzenetet egyszerre több embernek is elküldhetjük titkosítva, feltéve, hogy mindannyiuk nyilvános kulcsával rendelkezünk. További lehetőség a fájl *KGPG* segítségével végzett titkosítása, majd mellékletként való továbbítása.
A *KMail* önműködően visszafejti a mellékleteket; ehhez egyébként a megtekintés, és nem a megnyitás parancsot kell választanunk. Ha webes levelező ügyfelet használunk, akkor töltsük le a fájlt, majd a *Konquerorral* tudjuk visszafejteni vagy megnézni a tartalmát. Ha olyan webes levelezőt használunk, mint például a *Yahoo mail*, akkor másoljuk ki a vágólapra a titkosított üzenetet, az egér jobb gombjával a tálca ikonjára kattintva válasszuk a ***Decrypt clipboard (Vágólap visszafejtése)*** parancsot, majd illesszük be a vágólap tartalmát a *KGPG* szerkesztőjébe. Hasonló módon lehet elvégezni az üzenetek titkosítását is.

Elektronikus levél aláírása

A titkosításnál népszerűbb alkalmazás a levelek aláírása. Ilyenkor maga a szöveg nem kerül titkosításra, ám az aláírás igazolja az üzenet hitelességét. Ha minden levelemet aláírom, mert ez a házirend, és valaki kap tőlem egy aláíratlan vagy hibás aláírású levelet, akkor nyugodtan feltételezheti, hogy hamis, és értesítheti a megfelelő személyt. A *KMail* a leveleket különféle színekkel jelöli meg, a színekből tudhatjuk, hogy az egyes üzenetek megbízható féltől érkeztek-e. Sárgák az aláírt, zöldek az aláírt és megbízható levelek.

Csoportok létrehozása és egyéb lehetőségek

A *KGPG* további érdekes szolgáltatása a kulcscsoportok létrehozásának lehetősége. Lehet például egy Felügyelet csoportom, amelybe három-négy kulcsot helyezek el. Üzenet küldésekor ki tudom választani a csoportot. Később, ha az egyik címzett továbbítja a levelet a csoport egy másik tagjának, akkor az már készen fog állni az elolvasásra.
A ***Configure KGPG (KGPG beállítások)*** alatt az ***ASCII Armor (ASCII védelem)*** beállítást is érdemes engedélyezni, ahogy valószínűleg ez is az alapbeállítás. Ilyenkor az aláírások és a titkosítás tisztán szövegesen tárolódnak, így könnyebben lehet továbbítani, nyomtatni, másolni és beilleszteni az anyagot. Az ***ASCII*** védelem kikapcsolásakor bizonyos fájlok bináris formátumot kapnak, ami esetleg gondokat okozhat.

Összegzés

Amint lesz időm, ígérem, minden titkosított bejövő levelemet vissza fogom fejteni, és válaszolni is fogok. Mivel a *KGPG* a *KDE* része, a legtöbb *Linux* terjesztésben eleve megtalálható. Néhány kulcsot létrehozni és kipróbálni a lehetőségeket mindössze egy-két órát igényel.
A *GnuPG* és a *KGPG* telepítése után a kulcsok és a titkosítás használata a mindennapok gyakorlatában sem okozhat gondot, ha ügyelni akarunk a biztonságra. Eddigi munkáim során mindig nagy figyelmet fordítottak az összeköttetések és például az *SSL* feletti tranzakciók biztonságára, ám a fájlok és az elektronikus levelek valahogy kiestek ebből a körből. *KDE* használatakor a biztonságos levelezőrendszer megteremtése nem okozhat különösebb nehézséget. Érdemes úgy kezdeni, hogy először csak a felettünk álló vezetőknek szánt leveleket titkosítjuk; vagy jó ötlet lehet olyan magánmappát nyitni a hálózaton, amely kizárólag titkosított dokumentumokat tárol. Ha ezeket a biztonsági házirendeket már elfogadtattuk, a szélesebb körű titkosítást is könnyebb lesz megvalósítani.

Linux Journal 2005. január, 129. szám

Roy Hoobler a Connect Computing, Inc. tulajdonosa. (www.connectcomputing.com) Független tanácsadóként, a zárt szoftverek világában szerzett tíz éves tapasztalatával vállalkozása jelenleg kisvállalkozásokat segít a Linux használatában és nyílt forrású üzleti alkalmazások megvalósításában.

A gyökér fájlrendszer titkosítása

Ha adataink fizikai biztonságát nem tudjuk garantálni, akkor nincs más lehetőségünk, mint a fájlrendszer titkosítása. Bár írásomban egy PowerPC alapú rendszer átalakítását fogom taglalni, az alapelvek más géptípusokon is változatlanok.

A *Linuxvilág* „Titkosított saját könyvtárak megvalósítása” (2003 október) című cikkében már ismertettem, hogyan lehet átlátszó módon titkosítani a kezdőkönyvtárakat. Ez alkalommal egy másik megoldást szeretnék ismertetni, a gyökér fájlrendszer titkosítását. Szó lesz a *GNU/Linux* rendszertöltési folyamatáról és a szükséges programokról, adok némi további útbaigazítást, megismerkedünk az *Open Firmware*-rel és néhány további figyelembe veendő tényezővel. A fogalmakat egy a *Fedora Core 3* előzetes kiadását futtató, *New World PowerPC* alapú *Apple iBook* alapján ismertetem. Természetesen az itt szereplő fogalmak és eljárások bármely készüléken, géptípuson és operációs rendszeren érvényesek, alkalmazhatók. Feltételezem, hogy rendelkezünk egy különálló *USB*-s flash memória egységgel, továbbá a gép belső programja képes a róla végzett rendszerindításra.

Feltételezem továbbá, hogy az olvasó már jártasságot szerzett a forrásfoltok telepítésében és a programok fordításában. A *Fedora Core 3 Test 3* terjesztés esetében az `mkinitrd` és az `initscripts` csomagokat kell megfoltozni a titkosított gyökér fájlrendszer támogatásával. Nem árt, ha a lemezkötetek kezelésével és a fájlrendszerek létrehozásával is tisztában vagyunk. A *Linux* terjesztések alapszintű telepítésével itt nem áll módomban foglalkozni.

Mielőtt rátérnénk a műszaki jellegű kérdésekre, meg kell ismernünk egy magasabb szintű fogalmat, a megbízás fogalmát. A megbízás kifejezést leginkább titkosítási és hitelesítési témák kapcsán hallhatjuk. Az elektronikus kulccsal rendelkező készülékeket alapesetben megbízhatókként kezeljük. Például, ha beírom a *PIN*-kódomat egy pénzkiadó automatába, akkor feltételezem, hogy nem fogja harmadik félnek kiadni azt. Hasonlóan, amikor beírok egy titkosítási kulcsot a számítógépembe, feltételezem, hogy nem fogja másnak átadni. Megbízom a számítógépemben, titkunk a kettőnké marad.

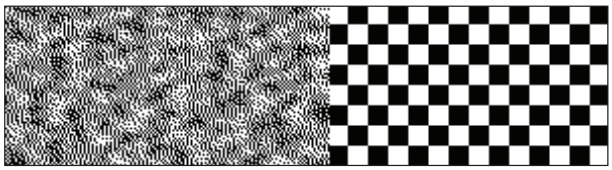
No de valóban megbízhatunk a számítógépünkben? Hacsak nem visszük mindenhova magunkkal, nyilván nem! Akkor is igaz ez, ha a lemezek titkosítva vannak. Vegyük a következő forgatókönyvet: míg alszunk, valaki ellopja a gépünket. A tolvaj – noha a visszafejtő kulcs hiányában egyelőre nem tud mi kezdeni vele – másolatot készít gépünk tartalmáról. Ezután a gép titkosított tartalmát lecseréli, majd visszateszi a gépet a helyére. Amikor másnap reggel felébredünk, a gép, ahogy minden nap, kéri tőlünk a kulcsot. Ez alkalommal azonban, amint megadjuk a kulcsot, elküldi azt a tolvajnak. Mivel most már adatainkkal és a kulcsunkkal egyaránt rendelkezik, el tudja olvasni dokumentumainkat.

Természetesen a példa kicsit erőltetett, de rendkívül szemléletes. A lényege az, hogy nem bízhatunk meg a számítógépünkben, hiszen nem tarthatjuk minden pillanatban szem előtt. Bármilyen jó is tehát a titkosítási rendszerünk, bizalmi alap nélkül épül fel.

Ha biztosítani akarjuk a számítógép rendszertöltési folyamatának megbízhatóságát, akkor el kell különítenünk tőle. A gondolat nem új, hiszen az autónknak is csak a kulcsát hordjuk magunkkal, magát az autót a parkolóban hagyjuk. A titkosítási kulcs természetes analógiája az autókulcsnak. A titkosítási kulcs könnyebben védhető, így a számítógépet nem kell mindenhová magunkkal vinnünk. Ennél egy kicsit tovább is lépünk, és a probléma megoldása céljából a számítógép indításához szükséges programot is a kulcson helyezzük el. A kulcs szerepét a flash meghajtó játssza. A rendszerindításért felelős program és a titkosítási kulcs együttes védelmével komolyan csökkenthetjük a rendszertöltő folyamat meghamisításának kockázatát.

Ehhez érteni kell a számítógép indításának folyamatát, ugyanis a titkosított gyökér fájlrendszer feloldása a rendszertöltő folyamat nélkülözhetetlen része. A jelenlegi üzembiztos, vagyis 2.6-os rendszermagok képesek az *initramfs* segítségével végzett rendszerindításra. (Lásd: *LWN.net*, *„Initramfs Arrives”*) Az *initramfs* egy `cpio` archív, a rendszermag most már tudja, hogyan bonthatja ki ezt egy *RAM*-lemezre. A kibontott fájlrendszer egy parancsfájlt tartalmaz, mely hagyományosan egy a gyökér fájlrendszer befűzéséhez szükséges rendszermagmodulokat betöltő parancsfájlt tartalmaz. Esetünkben ez a parancsfájl végzi majd a titkosított gyökér fájlrendszer visszafejtését is. A témáról bővebb tájékoztatást a *Linux* rendszermagforrások részeként elérhető *buffer-format.txt* és *initrd.txt* fájlokban lehet találni.

Linux alá számos fájlrendszer-titkosító felület létezik. *Jari Ruusu Loop-AES* megoldása is egy ezek közül. Számos *cryptoloop* változat is készült, ezek titkosított hurokeszközt biztosítanak. Írásomban a jelenlegi, 2.6-os rendszermagok



1. ábra Ha a fájlrendszer létrehozása előtt nem véletlenszerűsítjük a lemezrészt, akkor a támadó láthatja, mennyire van tele

2. ábra A lemezrész véletlenszerűsítésével eltitkolhatjuk annak kihasználtságát

által nyújtott ***dm-crypt*** felülettel foglalkozok. A ***Fedora Project*** jelenleg ezt a felületet helyezi előtérbe, és a ***dm-crypt*** modulok is szerepelnek a ***Fedora*** rendszermagcsomagjai között. Szükséges továbbá egy állandó jelleggel csatolt ***cryptsetup***. Segítségével leegyszerűsödik a ***dm-crypt*** eszközök kezelése. Végül, a rendszerindító fájlrendszer kezelésére a `parted` és a `hfsutils` szolgál.

Sajnos a ***Fedora Core anaconda*** telepítője alapesetben jelenleg nem támogatja a titkosított fájlrendszerre telepítést. Korlátain úgy léphetünk túl, hogy egy lemezrészt szabadon hagyunk, telepítjük a ***Fedorát***, megformázzuk titkosítottra a szabad lemezrészt, majd a telepítő által felmásolt fájlokat átmozgatjuk a titkosított részre. Az egyszerűség kedvéért feltesszük, hogy a ***Fedora*** telepítését két lemezrész használatával végezzük: a ***/dev/hda4*** a ***/home***, a ***/dev/hda5*** pedig a / elérési úttal van befűzve. Mivel a ***/home*** csak a ***Fedora*** telepítése után kerül feltöltésre, a ***/dev/hda4*** lesz a tartalék lemezrész, a ***/dev/hda3*** pedig a cseretár.

Fűzzük be a ***/dev/hda4*** meghajtót a ***/home***, a ***/dev/hda5*** meghajtót pedig a / elérési út alá, és telepítsük a ***Fedora Core 3*** terjesztést. A root-on kívül ne adjunk hozzá felhasználókat, mert a ***/home*** tartalma a későbbiek során törlésre kerül.

Ezen a ponton kapunk egy teljes értékű ***Linux*** rendszert. Mielőtt létrehoznánk a titkosított fájlrendszert, véletlenszerűsítenünk kell az általa elfoglalandó lemezrészt. Ezzel megelőzhetjük a lemez tartalmával kapcsolatos adatok kiszivárgását. Az *1. ábrán* egy elvonatkoztatott lemez látható, mely félig van tele, és véletlenszerűsítése nem volt megfelelő. A *2. ábrán* egy olyan lemez látható, melynek véletlenszerűsítését helyesen elvégezték, mielőtt megformázták volna a titkosított fájlrendszerrel. Vegyük észre, hogy az 1. ábra lemezéről lehet némi információt szerezni, például meg lehet állapítani, hogy az adatok csak félig töltik ki. A 2. ábrán látható lemeznél ilyesmitől nem kell tartani, a lemez akár teli, akár üres is lehet. Egy lemezrész véletlenszerűsítése tartalmának véletlenszerű adatokkal való felülírásával történik:

```
dd if=/dev/urandom of=/dev/hda4
```

A művelet végrehajtása eltarthat egy ideig, ugyanis a véletlenszerű adatok előállítása nem egyszerű feladat.

A ***/dev/hda4*** meghajtón az alábbi lépéseket követve hozhatunk létre titkosított ***ext3*** fájlrendszert:

1) Ellenőrizzük, hogy az `aes`, a `dm-mod` és a `dm-crypt` modul be van-e töltve a rendszermagba.
2) Válasszuk le a ***/home*** elérési út alól a titkosított gyökér fájlrendszer tárolására kiszemelt meghajtót vagyis a ***/dev/hda4***-et:

```
# umount /dev/hda4
```

3) Hozzunk létre egy véletlenszerű, 256 bites titkosító kulcsot, és mentsük el ***/etc/root-key*** néven:

```
# dd if=/dev/urandom of=/etc/root-key bs=1c
count=32
```

A kulcsot később át fogjuk másolni a flash meghajtóra.

4) Hozzunk létre egy dm-crypt eszközt, mely az imént létrehozott kulccsal kerül titkosításra:

```
# cryptsetup -d /etc/root-key create root /dev/hda4
```

A ***/dev/mapper/root*** most egy titkosított réteget szolgáltat a ***/dev/hda4*** felett. Alapesetben a `cryptsetup` egy ***AES*** algoritmussal titkosított `dm-crypt` eszközt hoz létre, és 256 bites kulcstér használatát feltételezi.

5) A ***/dev/mapper/root*** alatt hozzunk létre egy ***ext3*** fájlrendszert:

```
# mkfs.ext3 /dev/mapper/root
```

6) Fűzzük be az új fájlrendszert:

```
# mkdir /mnt/encroot
# mount /dev/mapper/root /mnt/encroot
```

7) Kész a titkosított fájlrendszer, fel kell töltenünk a ***/dev/hda5***, azaz az eredeti gyökér fájlrendszer tartalmával:

```
# cp -ax / /mnt/encroot
```

8) Végül létrehozunk egy bejegyzést a /mnt/encroot/etc/crypttab fájlban, a különféle segédprogramok innen tudhatják meg, hogy melyek a fájlrendszer beállításai:

```
root    /dev/hda4       /etc/root-key    cipher=aes
```

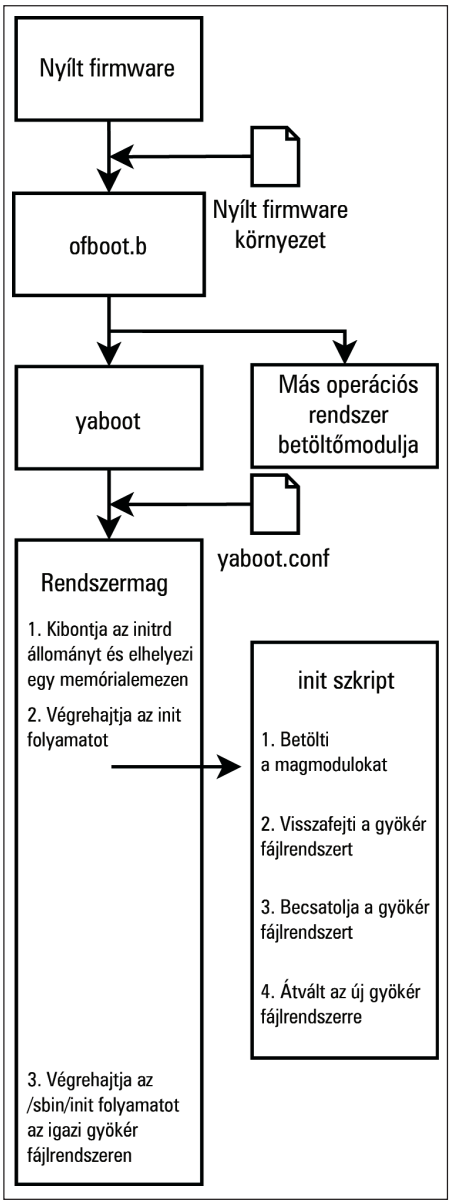
A titkosított fájlrendszer is készen áll, a továbblépéshez azonban ismernünk kell gépünk rendszertöltési folyamatát. A számítógépek általában rendelkeznek valamilyen belső programmal, ez adja át a vezérlést a rendszertöltést végigvivő programnak. A belső program védelmének tárgyalása túlmutatna a cikk témáján, ezért feltételezzük, hogy a rendszer belső programja megbízható. A legtöbb olvasó számára valószínűleg ismerős a ***BIOS***, a ***PC*** alapú gépek által használt rendszertöltő belső program. Itt az ***Open Firmware*** rendszertöltővel foglalkozok, ezt több gyártó is alkalmazza, például az ***Apple***, a ***Sun*** és az ***IBM***.

A *NetBSD/macppc* telepítési útmutatója kiváló ismertetőt tartalmaz az *Open Firmware*-hez. Célunk az, hogy az *Open Firmware* parancssoros felületének segítségével eltávolítható flash meghajtóról végzett rendszerindításra állítsuk be a számítógépet. Az *Open Firmware* lehetővé teszi a számítógéphez csatlakoztatott készülékek megtekintését, továbbá a belső program változóinak megtekintését és módosítását.
Az *Open Firmware* parancssorát a *New World* (*G3* vagy újabb) *Apple* gépeken az OPTION-COMMAND-O-F billentyűkombináció rendszertöltés közben való lenyomásával lehet elérni.
Azt, hogy a rendszer melyik eszközt használja a rendszertöltésre, a `boot-device` változó határozza meg.
A változó értékét a `printenv` paranccsal vizsgálhatjuk meg:

```
> printenv
[...]
boot-device      hd:,\\:txbi
➥hd:,\\:txbi
```

A fenti sor értelmezése a következő: *„a rendszerindítást az első IDE lemezen található, HFS típusú txbi fájl futtatásával kell elvégezni”*. A második kettőspont, mely a `txbi` előtt található, a zseton *HFS* fájltípusként való értelmezését váltja ki. Egyébként a program a `txbi`-t egy fájl elérési útjaként értelmezné. Esetemben a `hd` zseton valójában egy álnév a jóval bonyolultabb `/pci@f4000000/ata-6@d/disk@0` eszközhöz. A karakterlánc az első *IDE* merevlemez különféle alrendszereken keresztül vezető elérési útját adja meg. Azt, hogy egy álnév melyik eszközhöz tartozik, az *Open Firmware* `devalias` parancsával tekinthetjük meg.
A `boot-device`, a rendszerindító eszköz beállításához meg kell tudnunk, hogy az *Open Firmware* milyen névvel azonosítja flash meghajtónkat. Az ls paranccsal kapott eszközfát megvizsgálva azonnal fény derül a flash meghajtó elérési útvonalára:

```
> dev / ls
[...]
        /pci@f2000000
                [...]
                /usb@1b,1
                        [...]
                        /disk@1
[...]
```

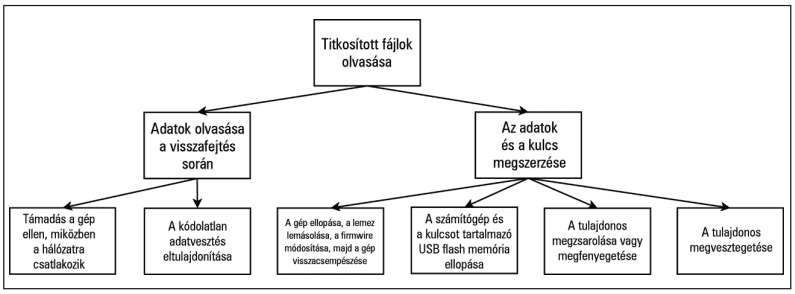


3. ábra Open Firmware-rel ellátott, PowerPC alapú rendszer indításának folyamata

Szereztünk némi betekintést a belső programnak a számítógépről alkotott képébe, ezért térjünk át a belső program által kezdésként futtatott program, vagyis a rendszertöltő vizsgálatára. Az *Apple PowerPC* gépein futó Linux rendszerek általában a yaboot nevű program közreműködésével indítják a gépet. A `yaboot` hasonló a *LILO*-hoz vagy a *GRUB*-hoz, és két kulcsprogramot tartalmaz, az *ofboot.b*-t és a *yabootot*. Az *ofboot.b* a rendszertöltés első fázisáról gondoskodik. Lényegében az *ofboot.b* feladata az indítandó operációs rendszer kiválasztása. Ha például a rendszerre a *Mac OS X* és a *Linux* is telepítve van, akkor az *ofboot.b* indítja el a *Mac OS X* vagy a *Linux* rendszertöltőjét. Ha a felhasználó a *Linux* betöltése mellett dönt, az *ofboot.b* a *yabootot* indítja el, mely a rendszertöltési folyamat második fázisát képviseli. A *yaboot* ezután betölti a *Linux* rendszermagot, és jelen esetben az *initrd*-t.
A *3. ábra* a *Linux* titkosított gyökér fájlrendszerről való elindítását szemlélteti (*PowerPC* géptípuson).
Eltávolítható rendszerindító eszközünkre az *ofboot.b* és a *yaboot* programot, egy *Linux* rendszermagot, valamint a titkosítási kulcsot tartalmazó *initrd*-t kell rámásolnunk. Az *Apple* jelenlegi *PowerPC* alapú gépei a rendszerindításra használt adathordozón *HFS* fájlrendszert várnak.

1) A `parted` programmal hozzuk létre a megfelelő, indításra is használható lemezrészt a flash meghajtón. (Saját meghajtóm 64 MB-os, és a */dev/sda* eszközcsomóponton keresztül érhető el.):

```
# parted /dev/sda
(parted) mklabel mac
(parted) print
Disk geometry for /dev/sda: 0.000-62.500 megabytes
Disk label type: mac
Minor   Start      End    Filesystem  Name    Flags
1       0.000      0.031       Apple
(parted) mkpart primary hfs 0.031 62.500
(parted) print
Disk geometry for /dev/sda: 0.000-62.500 megabytes
Disk label type: mac
Minor   Start      End    Filesystem  Name    Flags
1       0.000      0.03        Apple
2       0.031     62.500    untitled
```



2. ábra Hogyan tudja egy támadó olvasni a titkosított fájlrendszert?

```
(parted) set 2 boot on
(parted) name 2 Apple_Boot
(parted) quit
```

2) Hozzuk létre a ***HFS***-t a rendszertöltő lemezrészen:

```
# hformat /dev/sda2
```

3) A /mnt/encroot/etc/yaboot.conf módosításával állítsuk be a yabootot a megfelelő eszközről végzett rendszerindításra. Az alábbiakban egy minimális beállításkészlet szerepel:

```
boot=/dev/sda2
ofboot=/pci@f2000000/usb@1b,1/disk@1:2
partition=2
install=/usr/lib/yaboot/yaboot
magicboot=/usr/lib/yaboot/ofboot
default=linux
image=/vmlinux
        label=linux
        root=/dev/hda4
        initrd=/initrd.gz
        read-only
```

A `/pci@f2000000/usb@1b,1/disk@1:2` érték az ***Open Firmware*** eszközfájának korábbi vizsgálatából származik, a `/pci@f2000000/usb@1b,1/disk@1` pedig a `f2000000` címen elérhető ***PCI*** busz első ***USB*** buszának első lemeze. A bennünket érdeklő eszköz egy lemez (disk), a `:2` jelölés pedig a második lemezrészre utal.

4) Telepítsük a rendszertöltő programokat és a rendszermagot a */**dev**/**sda2***-re:

```
# ybin –config /mnt/encroot/etc/yaboot.conf -v
# mount /dev/sda2 /media/usbstick
# cp /boot/vmlinux /media/usbstick
```

A következő lépés a titkosításra képes initrd telepítése a flash meghajtóra. A Fedora tartalmaz egy mkinitrd nevű eszközt, mely alkalmas initrd létrehozására. Sajnos cikkem születésekor az mkinitrd még nem volt képes titkosított gyökérkönyvtárat befűzni. A ***https://bugzilla.redhat.com/bugzilla/show_bug.cgi?id=124789*** címen elérhető folttal viszont felülkerekedhetünk ezen a problémán is. A folt telepítése után az mkinitrd képes lesz a */**etc**/**crypttab*** olvasására, illetve a megfelelő ***initrd*** létrehozására:

1. `mkinitrd –authtype=paranoid -f /media/usbdisk/initrd.gz <rendszermagváltozat>`
2. `umount /media/usbstick`

Át kell írni a /mnt/encroot/etc/fstab fájlt is, követve a módosításokat:

```
/dev/mapper/root /      ext3    defaults      1 1
```

A cseretár titkosítása vagy teljes elhagyása a titkosított fájlrendszer használatának természetes velejárója. Ennek okai a „Titkosított saját könyvtárak megvalósítása” című cikkben és a ***BugTraq*** levelezési lista ***„Mac OS X stores login/Keychain/FileVault passwords on disk” (A Mac OS X lemezen tárolja a bejelentkezési/Keychain/FileVault jelszavakat)*** című beszélgetésében találhatók meg. Ha a ***https://bugzilla.redhat.com/bugzilla/show_bug.cgi?id=127378*** címről letölthető foltot feltesszük az ***initscripts*** csomagra, a ***Fedora*** lehetővé teszi a felhasználók számára, hogy véletlenszerűen létrehozott munkamenetkulccsokkal titkosítsák a cseretárként használt lemezrészt. Mivel általában a cseretárhelynek az egyes rendszerindítások alkalmával nem kell megőriznie tartalmát, a munkamenetkulcs a rendszer leállításakor nem kerül mentésre. A titkosított cseretár engedélyezéséhez a következő lépéseket kell végrehajtani:

1) A */**mnt**/**encroot**/**etc**/**fstab*** fájlhoz adjuk hozzá az alábbi sort, lecserélve vele a korábbi swap (cseretár) bejegyzést:

```
/dev/mapper/swap swap   swap    defaults      0 0
```

2) Adjuk hozzá az alábbi sort a */**mnt**/**encroot**/**etc**/**crypttab*** fájlhoz, ezzel közöljük a rendszerrel, hogyan kell elvégeznie a titkosítást:

```
swap    /dev/hda3       /dev/urandom    swap
```

Elértünk oda, hogy újra tudjuk indítani a rendszert, és használatba tudjuk venni a titkosított fájlrendszert. Ismét tartsuk lenyomva az OPTION-COMMAND-O-F billentyűkombinációt, és lépjünk be az ***Open Firmware*** parancssorába. Korábban már láttuk, hogy a flash meghajtó második lemezrészének elérési útja `/pci@f2000000/usb@1b,1/disk@1:2`. Ennek tudatában összeállíthatjuk a `/pci@f2000000/usb@1b,1/disk@1:2,\ofboot.b` elérési útvonalat. A vessző a lemezrész számát és a fájlrendszer elérési útját választja el egymástól. A `\ofboot.b` a fájlrendszerbéli elérési út, a \ pedig a ***UNIX*** / fájlrendszer gyökér jelöléséhez hasonlóan az eszköz gyökerét adja meg:

```
> dir /pci@f2000000/usb@1b,1/disk@1:2,\
Untitled            GMT                     File/Dir
   Size/      date       time   TYPE         Name
   bytes   9/ 3/ 4   21:44:41  ???? ????    initrd.gz
 2212815   8/28/ 4   12:24:21  tbxi UNIX    ofboot.b
    3060   9/ 3/ 4    2:21:20  ???? ????    vmlinux
```

```
  141868   9/28/ 4   12:24:22  boot UNIX  yaboot
     914   9/28/ 4   12:24:22  conf UNIX
yaboot.conf
```

Ezzel ellenőriztük, hogy az ***Open Firmware*** képes a rendszer elindításához szükséges fájlok olvasására. Ha a `boot-device` változót a `/pci@f2000000/usb@1b,1/disk@1:2,\ ofboot.b` értékre állítjuk, a rendszer a flash meghajtóról fog indulni:

```
 setenv boot-device
/pci@f2000000/usb@1b,1/disk@1:2,\ofboot.b.
```

Miután a gép sikeresen elindult a titkosított fájlrendszerről, semmisítsük meg a ***/dev/hda5*** által tárolt adatokat. Erre a célra a gyökér fájlrendszer lemezrészének véletlenszerűsítésénél már kipróbált módszert alkalmazzuk:

```
 dd if=/dev/urandom of=/dev/hda5
```

Ha gondoljuk, a felülírást többször is elvégezhetjük. A lemezek tartalmának kiirtására szabványt például az ***Amerikai Védelmi Minisztérium „National Industrial Security Program Operating Manual" (Nemzeti ipari biztonsági program üzemeltetési kézikönyve)*** dokumentumának 8. fejezetében találunk. Megfelelő törlés után a ***/dev/hda5*** például ***/home*** könyvtárként használható. A ***/home*** fájlrendszert szintén titkosítani kell. Szerencsére ennek folyamata jóval egyszerűbb, hiszen a gép nem a ***/home*** könyvtárból indul. Magát a fájlrendszert a gyökér fájlrendszerhez hasonló módon hozhatjuk létre.

1) Ellenőrizzük, hogy az ***aes***, a ***dm-mod*** és a ***dm-crypt*** modul be van-e töltve a rendszermagba.

2) Válasszuk le a ***/home*** elérési út alól a titkosított kezdőkönyvtár-fájlrendszer tárolására kiszemelt meghajtót, vagyis a ***/dev/hda5***-öt:

```
# umount /dev/hda5
```

3) Hozzunk létre egy véletlenszerű, 256 bites titkosító kulcsot, és mentsük el ***/etc/home-key*** néven. A szükséges parancs:

```
# dd if=/dev/urandom of=/etc/home-key bs=1c
count=32
```

4) Hozzunk létre egy ***dm-crypt*** eszközt, mely az imént összeállított kulccsal kerül titkosításra:

```
# cryptsetup -d /etc/home-key create home /dev/hda5
```

5) A ***/dev/mapper/home*** alatt hozzunk létre egy ***ext3*** fájlrendszert:

```
# mkfs.ext3 /dev/mapper/home
```

6) Fűzzük be az új fájlrendszert:

```
# mount /dev/mapper/home /home
```

7) Hozzunk létre egy bejegyzést a ***/etc/crypttab*** fájlban, a különféle segédprogramok innen olvashatják ki a fájlrendszer beállításait:

```
root    /dev/hda5       /etc/home-key    cipher=aes
```

8) Végül a ***/home*** könyvtárhoz tartozó bejegyzéssel bővítsük a ***/etc/fstab*** fájlt:

```
/dev/mapper/home /home   ext3    defaults      1 2
```

Szép munkát végeztünk, nekiláthatunk a további, nem root felhasználói fiókok hozzáadásához. A titkosított gyökér fájlrendszer létrehozása befejeződött.
Ha az összes adatunk titkosítva van, annak bizonyos veszélyei is lehetnek. Ha elveszítjük a titkosító kulcsot, vele együtt minden adatunk elvész. Éppen ezért ne feledjünk biztonsági másolatokat készíteni a kulcsot tartalmazó flash meghajtóról. Fontos továbbá, hogy a titkosított adatokról nyílt szövegű mentéseket is készítsünk. Ha rendszerindításra alkalmas vészlemezt is összeállítunk, jól gondoljuk át, hogy milyen összetevőket helyezünk el rajta. A gyökér és a kezdőkönyvtár-fájlrendszer kulcsára mindenképpen szükség lesz, de nem hiányozhat a `parted`, a `hfsutils`, a titkosítással kapcsolatos rendszermagmodulok és a `cryptsetup` sem.
Mennyire hatékonyan lehet ezzel a módszerrel megvédeni az adatokat? ***Secrets and Lies (Titkok és hazugságok)*** című könyvében ***Bruce Schneier*** ismertet egy olyan technikát, mellyel érdemi választ adhatunk erre a kérdésre. A fenyegetések modellezésére egy úgynevezett támadásfát lehet használni. A ***4. ábra*** titkosított fájlrendszerünk támadásfájának elejét tartalmazza. Mindenképpen ki kell emelni, hogy ez a támadásfa nem teljes, és talán soha nem is lesz az.
A cikkemben szereplő megoldást követve, némi kreativitással bárki erőteljes védelemmel láthatja el adatait bizonyos típusú lopások ellen. Nem szabad azonban elfeledkezni azokról a támadástípusokról, amelyek ellen ezek a védelmi megoldások hatástalanok. Nyilvánvaló, hogy a hálózati és egyéb támadásokkal szemben más módszerekkel kell védekezni, ám az itt szereplő fogások sokban hozzájárulnak a rendszer általános biztonságának megalapozásához.

Linux Journal 2005. január, 129. szám

Mike Petullo jelenleg tesztmérnök a WMS Gamingnél. 1997 óta ismeri a Linuxot. Bárki véleményét szívesen fogadja az lj@flyn.org címen.

KAPCSOLÓDÓ CÍMEK

- lwn.net/Articles/14776
- sourceforge.net/projects/loop-aes
- www.dss.mil/isec/nispom_0195.htm

Paranoid Pingvin: a Linux biztonságának kockázat alapú megközelítése

A kockázat elkerülhetetlen. Legyünk pesszimisták a programhibákkal kapcsolatban, készítsünk terveket a hibák kezelésére, így egész rendszerünket biztonságban tudhatjuk.

Amióta négy évvel ezelőtt elkezdtem írni ezt a rovatot, a ***Linux*** program sérülékenységei és fenyegetései nem sokat változtak. Veremtúlcsordulások, hibás beállítások (fájl jogosultsági kérdéseket is beleértve) nem megfelelő bemenet érvényesítés jelentik továbbra is a ***Linux*** sérülékenységek oroszlánrészét. Ha ugyanolyan típusú sérülékenységek bukkannak fel újra és újra, nem értelmetlen ez az egész foltverseny? Nincs egy szélesebb látókörű ***Linux*** biztonsági elgondolás?

Ebben a hónapban a ***Linux*** biztonságot kockázat alapú nézőpont szerint vizsgáljuk, és bemutatjuk, hogyan használjuk a kockázat alapú megközelítést az ismert ***Linux*** sérülékenységeken túl olyan problémák csillapítására, amelyeket még senki nem fedezett fel vagy adott közre.

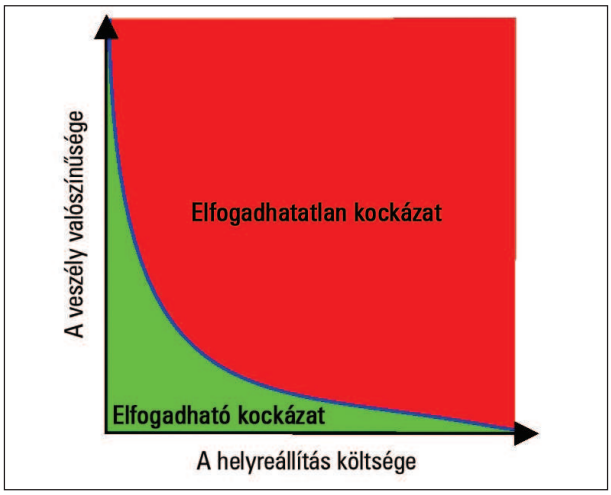
A biztonság kockázat alapú megközelítése

Biztosan vannak akik azt kérdezik magukban, mit értek egyáltalán kockázat alapú megközelítésen? Hát az információs biztonság nem elve a kockázatról szól? Tulajdonképpen igen, azonban ezt a kifejezést valójában a kockázat kezelés alapú megközelítés rövidítéseként használom.

Egy adott információs biztonsági kockázatot csak néhány módon lehet kezelni. Elkerülhetjük, azaz semmi olyat nem teszünk ami kitehetne bennünket a veszélynek. Megszüntethetjük, gyökerénél kezelve (amire sajnos a gyakorlatban ritkán van lehetőségünk). Enyhíthetjük, azaz, valami olyat csinálunk ami csökkenti a kockázat hatását valamilyen módon. Végül még egy lehetőségünk van: elfogadjuk.

Az egyik, szerencsére mostanra elavult elgondolás szerint a biztonság bináris egyenlet: a dolgok vagy biztonságosak vagy ostobák, és ha úgy találjuk, hogy egy tevékenység vagy eszköz nem biztonságos, akkor azt nem szabad csinálni vagy használni. Más szavakkal ez az iskola azt hirdeti, hogy a biztonság elsődleges irányelve az elkerülés.

Mint azt egyre többen felismerik manapság, tökéletes biztonság nem létezik. Nincsenek mágikus programkombinációk, program/rendszer összeállítások vagy hálózatszerkezet ami sebezhetetlenné tenne bennünket a betörésekkel szemben.



1. ábra Kockázati határok

Nincs ilyen összeállítás, legalábbis olyan, amivel dolgozni is lehet. A hálózati számítástechnikában valamilyen szintű kockázat elkerülhetetlen.

A kockázat kezelés alapú megközelítés felismerte, hogy a kockázat elkerülése, a kockázatcsökkentés és elfogadás közötti egyensúlyra kell törekedni, mégpedig a kockázatokat fontossági sorrendbe rendezve valószínűségük és romboló hatásuk alapján. Csökkentés vagy elkerülés szempontjából a legfontosabb kockázati tényezők azok lesznek, amelyek nagyobb valószínűséggel következnek be és helyreállításuk költséges. Az erősen valószínűtlen, vagy olcsón helyreállítható hatású kockázati tényezők viszont gyakran kerülnek az elfogadható kategóriába. Mondanom se kell, amikor a kockázati tényező bekövetkezésének költségéről vagy hatásáról beszélek, nem kizárólag a pénzügyi költségre gondolok. Az időbeli és termelékenységi veszteség valamint a jó hírnevünk éppúgy ide tartozik.

Az *1. ábra* mutatja a kockázat valószínűsége, költsége és elfogadhatósága közötti általános viszonyt. Az elfogadható és elfogadhatatlan kockázati területeket kijelölő görbe pontos alakja szervezetről szervezetre változhat.

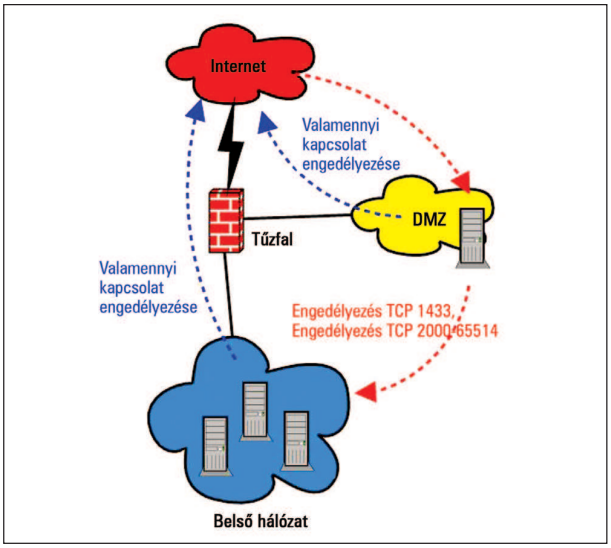
Egy pénzügyi cég például sokkal nagyobb vörös zónával rendelkezik mint egy egyetemi hálózat. A kockázat biztonság alapú megközelítése végül is annak felismerését jelenti, hogy nem minden kockázat azonos, ezért harcteret kell választatnunk. Azonban ahhoz, hogy ezt hatékonyan tehessük meg, becsületesen és találékonyan kell beazonosítanunk és felbecsülnünk az adott vállalkozás kockázati tényezőit. Egy létező hiba figyelmen kívül hagyása sokkal veszélyesebb mint tudomásul venni és elfogadni a hibát és elkészíteni a visszaállítási terveket arra az esetre, ha a legrosszabb bekövetkezne.
Ezzel el is érkeztünk a kockázat alapú megközelítés egy másik fontos tételéhez: a kockázat elfogadása még nem jelenti azt, hogy elégedettek lehetünk. Bármilyen kockázatot, amit nem tudunk elkerülni, vagy legalább enyhíteni, figyelembe kell vennünk az folyamatosság fenntartási és helyreállítási terveinkben. Csak nagyon kevés információs kockázat nem enyhíthető valamilyen mértékben; vannak veszélyforrások amelyek nem szüntethetőek meg, de a legtöbb felhígítható vagy lefékezhető.

Sérülékenységek és veszélyek

Rendben, tehát a ***Linux*** biztonsága legjobban a kockázat alapú szemléletmóddal kezelhető. Hogyan is nézne ez ki? Az első lépés ***Linux*** rendszerünk ismert és potenciális sérülékenységeinek számbavétele. A legtöbb ***Linux*** alkalmazás és rendszer-sérülékenység az alábbi kategóriák valamelyikébe esik:

- veremtúlcsordulás sérülékenység (nem megfelelő határ ellenőrzés).
- Elégtelen bemenet hitelesítés.
- Helytelen fájl jogosultságok.
- Nem megfelelő rendszer jogosultságok (felesleges root használat).
- Hanyag beállítások.
- Ideiglenes állományok nem biztonságos használata.
- Megjósolható vagy ismert alapértelmezett jelszavak alkalmazása.
- Adminisztrációs hátsó kapuk (tesztelési és hibakeresési felhasználónevek).

Valószínűleg a lista legelső sérülékenysége, a veremtúlcsordulás a legriasztóbb. A veremtúlcsordulások általában közvetlenül távoli root betöréshez vezetnek. A veremtúlcsordulás feltételeihez hasonlóan a fenti sérülékenységek egy része közvetlenül programozási hibák következménye. Ilyen például az ideiglenes állományok helytelen használata és adminisztrációs hátsó ajtók. Mások inkább felhasználótól függőek, ilyen a kiszámítható jelszó és a hanyag beállítások. Egyetlen sérülékenység sem jelent veszélyt, ha senki sem próbálja meg kihasználni azt. Más szavakkal, a fenyegetés két alkotóeleme a sérülékenység és a támadó.
A második lépés végiggondolni, milyen módszerekkel lehet kihasználni ezeket a sérülékenységeket. Bár a ***Linux*** biztonsági kockázatai nem sokat változtak az elmúlt évek során, az őket kihasználni szándékozó szereplők mások. Hatékonyabbá és butábbá váltak. Az a rémisztő igazság, hogy a törőkódok és szkriptek könnyű elérhetősége miatt a képzetlen támadók is egyre könnyebben tudnak kifinomult támadásokat levezényelni.



2. ábra Egyszerű tűzfal összeállítás

Korábban például egy veremtúlcsordulás támadás levezényléséhez komoly programozási tudás kellett, ki kellett találni a memóriában hová kerül a túlírt adat, a támadónak létre kellett hozni vagy be kellett szereznie a célrendszerre (például ***i386*** vagy ***SPARC***) jellemző assemblyben írt törőkódot vagy héjkódot. A héjkód az a kód ami majd túlcsordul és végrehajtódik, létrehozva egy héjat a célrendszeren, ideális esetben root jogosultságokkal.
A régi időkben az eltolások meghatározása és működő héjkód megírásnak nehézségei a veremtúlcsordulás alapon támadók táborát erősen leszűkítette. Manapság azonban, ha ki szeretnénk használni egy jól ismert veremtúlcsordulás sérülékenységet, nem kell mást tennünk mint kiadni egy jól formázott ***Google*** keresést, és máris megszereztük a törőkódot a hozzá tartozó, különféle rendszerekhez készült héjkóddokkal.
A törőkódokat író és az interneten közreadó emberek elég nagy problémát jelentenek. De nem ők az egyedüli szereplői a nagy egyenletnek; aki örömét leli benne, hogy script kiddie-ket fegyverez fel, már nem sok választja el attól, hogy féregbe vagy vírusba csomagolva automatizálja törőkódját. A vírusok természetesen magukat nem tudják terjeszteni; mindig valami másba ágyazva találhatóak, például e-mail csatolmányban vagy végrehajtható állományokban. A férgek viszont magukat terjesztik, így lényegesen félelmetesebbek, hiszen tulajdonképpen szárnyas vírusok. Ha féregtámadás idején a naplóbejegyzéseket nézzük, nagyon nehéz megkülönböztetni őket egy ember vezérelte támadástól. A féreg tulajdonképpen egy támadó robot.
Így a támadó lehet ember és lehet program. A jó hír, hogy mivel a ugyanolyan típusú sérülékenységet támadnak, a védekezés is hasonló. A rossz hír, hogy a támadó szkriptek férgek és vírusok exponenciális mértékben csökkentik a sérülékenység felfedezése és közzététele, valamint a rendszerünk valószínűsíthető megtámadása közötti időt.

Első védelmi megoldás: Tűzfal rendszabályok

Kezdjük el ezeket a fenyegetéseket a megfelelő védelemhez rendelni. Itt kezd majd a kockázat alapú megközelítés igazán fontossá válni.

Amennyiben mindent vagy semmit alapon tekintünk a biztonsági kérdésekre, a védelem egyszerű. A programokat nem képességeik, támogatottságuk és biztonságosságuk összesítése alapján, hanem tisztán a biztonságosságuk szerint válogatjuk. Minthogy a fő programfeltételünk a biztonság egyetlen dologra kell figyelnünk, nevezetesen a foltozásra, és minden rendben lesz.

Elképzelhető, hogy úgy állítjuk be a tűzfalunkat, hogy semmilyen kívülről jövő kapcsolatban se bízzon, de fogadjon el minden belülről érkezőt, hiszen, természetesen minden külsős gyanús és minden belső megbízható. Ami azt illeti, a programfoltok és tűzfalszabályok olyannyira fontosak ebben az elképzelésben, hogy tulajdonképpen semmi más nem számít.

A program foltok és tűzfal szabályok tényleg nagyon fontosak. Azonban a programfoltoktól való függésünk foka és tűzfalhasználatunk mikéntje egy kicsit eltérő lehet, ha vesszük magunknak a fáradtságot és meggondoljuk milyen valós fenyegetéstől óvnak bennünket. Képzeljük el a *2. ábrán* felvázolt helyzetet. A tűzfal a ***DMZ*** hálózatot védi a külső világtól, az pedig a belső hálózatot védi a külső világtól és a ***DMZ***-től.

A 2. ábrán pontozott vonallal jelölt tűzfal szabályok a következőképpen nézhetnek ki:

1. Az összes belső gép elérheti az Internetet bármelyik kapu/protokoll párossal.
2. Az ***DMZ*** gépnek engedélyezzük az Internet elérést bármely kapun/protokollon.
3. Az összes internet gép elérheti a ***DMZ***-t a 80-as TCP kapunk keresztül (HTTP).
4. A ***DMZ*** webkiszolgálója elérheti a belső gépeket a 1433 és 2000-65514 közötti TCP kapukon.

Elsőre egész jónak tűnik. A belső felhasználók mindenféle dolgokat csinálnak az Interneten, így ezt korlátozni zűrös lenne. A naplók miatt a ***DMZ DNS*** lekérdezéseket végez, szóval miért ne adnánk Internet elérést neki is? Van ezen felül egy háttéralkalmazás amit a ***DMZ***-be tett webkiszolgálónak el kell tudni érnie a belső hálózaton és ahol adatbázis lekérdezéseket adhatunk ki a 1433-as ***TCP*** kapun valamint egy véletlenszerűen választott magas kapun ami egy olyan véges tartományba esik, amit senkinek nem sikerült dokumentálnia. Ezért aztán a legegyszerűbb ha megnyitjuk az összes 1999-nél magasabb ***TCP*** kaput.

De nézzünk három kézenfekvő kockázati tényezőt:

1. Az internet alapú támadó feltöri a webkiszolgálót és más gépeket támad vele az interneten.
2. Egyik belső gépet féreg fertőzi meg valamilyen ***RPC*** sérülékenységen keresztül, és a fertőzött rendszer hatalmas mezőket kezd pásztázni az interneten más sérülékeny rendszereket kutatva.
3. Egy féreg megfertőzi a belső rendszert és hátsó kaput nyit a 6666-os ***TCP*** kapun. A támadó feltöri a webkiszolgálót, letapogatja a tűzfalat, felderíti a jól ismert féreg ajtót és belép a belső rendszerbe.

Az első kockázati kérdésben egyértelműen jogi problémáknak tesszük ki magunkat. Ha a webkiszolgálót feltörik és

a tűzfalunk nem akadályozza meg rajta keresztül a külső világ elérését, felelősnek érezhetjük magunkat, hiszen a mi rendszerünkről támadnak más rendszereket az interneten. A webkiszolgáló kimenő elérését a legszükségesebb szolgáltatásokra korlátozva csökkenthetjük a kockázatot. A gyakorlatban egy ***DMZ***-be helyezett webkiszolgálónak nagyon kevés adatfolyamra van szüksége a külső világ felé, ha szüksége van egyáltalán ilyesmire. Az a feladata, hogy az internetről érkező ***HTTP*** lekérdezésekre válaszoljon, nem pedig, hogy maga kezdeményezzen kapcsolatokat.

A második forgatókönyv szerint hasonló problémáknak tesszük ki magunkat, de itt a jogi következményeknél komolyabb gondot okoz a dolog hálózati teljesítmény oldala (a pásztázás forgalma eldugíthatja az Internetes kapcsolatunkat). Akárcsak az előbb, egy szigorúbb tűzfal rendszabály a kimenő forgalomra egyértelműen csökkenti a kockázatot.

A harmadik forgatókönyv egy kicsit szokatlanabb mint a többiek, végül is mennyi az esélye, hogy egy féreg megfertőzze a belső rendszert és egyúttal a ***DMZ***-ben található webkiszolgálónkat is feltörjék? Nos, tulajdonképpen a kettőnek nem kell egyszerre bekövetkeznie. Ha a féreg csendben marad miután elkészíti a hátsó ajtót a 6666-os ***TCP*** kapun, jó ideig nem fogjuk felfedezni. Tehát a webkiszolgáló feltörése nem kell, hogy aznap vagy akár abban a hónapban történjen, hiszen a féreg munkáját a fertőzött rendszerben nem hatástalanítottuk elég hamar. Akárcsak az előző két forgatókönyvben, egy szigorúbb rendszabály csökkentheti a kockázatot és minimalizálja annak az esélyét, hogy a féregfertőzést külsősök ki tudják használni.

Azon felül, hogy szigorúbb szabályokkal veszélyességük csökkenthető, e három veszélyforrásnak van még egy közös jellemzője. Csökkentésükhöz nem kell pontosan megjósolni őket. Elég ha csak arra gondolunk „mi van ha a tűzfal szabályaim ellenére valamilyen féreg vagy vírus bejön, és váratlan típusú kimenő forgalmat kísérelnek meg?" Nem tudom eléggé hangsúlyozni, nagyon fontos, hogy a tűzfal szabályok készítésekor ne csak a támadások megelőzésére koncentráljunk. Legalább ilyen fontos, hogy végiggondoljuk, mi történik ha a védelmünk csődöt mond. Az információs biztonság terén a pesszimizmus épít és nem rombol.

Remélem mostanra világossá vált, hogy véleményem szerint nem a tűzfal szabályok adnak választ az összes Linux biztonsági kérdésünkre. Összefoglalva, hatékony tűzfal szabályokat akkor tudunk készíteni, ha nem csak az ismert fenyegetéseket, hanem a lehetséges fenyegetéseket is figyelembe vesszük.

Második védelmi megoldás: az alkalmazások biztonsága

Nos, ha a tűzfal nem csodaszer, akkor mit tehetünk? A rovatban korábban a hanyag beállításokat az egyik legnagyobb sérülékenységi forrásnak neveztem; ezt megfordítva, a figyelmes beállítás az egyik legjobb védelem.

Tegyük fel, a ***DMZ***-ben van egy ***SMTP*** átjáró ami az Internet és a belső hálózat közötti levelezést bonyolítja. Tegyük fel továbbá, hogy a szervezetünk műszaki személyzetének nagy tapasztalata van a ***Sendmail*** terén, viszont se idejük, se hajlandóságuk megtanulni a ***Postfix*** használatát, amit egyébként mint megrögzött biztonságmániás, sokkal biztonságosabbnak tartok. A vezetés úgy dönt az átjáró ***Sendmailt*** fog futtatni. Minden elveszett? Nem szükségszerűen. Először is, mint azt a rovatban korábban is említettem, a ***Sendmail*** biztonsági mutatói tulajdonképpen egészen jók lettek az utóbbi években. Azonban, ha mindez meg is változik egyetlen éjszaka alatt, és a rossz fiúk három újabb veremtúlcsordulás hibát találnak a ***Sendmailben*** amit nem adnak közre azonnal, a ***Sendmail*** fejlesztők egészséges pesszimizmusának hála, ***Sendmail*** átjárónk még nem bukott el feltétlenül.

A ***Sendmail***-nek van néhány fontos biztonsági képessége, amiből kettő különösen hasznos lehet veremtúlcsordulások esetén. A ***Sendmail*** futhat ***chroot*** ketrecben, ami korlátozza a látható fájlrendszer területet, valamint előjogok nélküli felhasználó és csoportjogosultságokkal is indítható, így minimalizálhatjuk annak az esélyét, hogy a ***Sendmail*** sérülékenység közvetlenül root eléréshez vezessen. Minthogy a ***Sendmail*** a kivételezett 25-ös ***TCP*** kapura hallgat, legalább az idő egy részében rootként kell futnia, ezért a ***Sendmail*** gyakorlatilag időnként lefokozza saját magát különleges jogosultság nélküli felhasználó/csoport szintre. Ez tehát az enyhítés nem teljes, csak részleges.

Manapság a legtöbb jól tervezett hálózati alkalmazás chroot-olható és futtatható előjogok nélküli felhasználó/csoport jogosultságokkal. Ahogy a jó tűzfal szabály egyszerre célozza a megelőzést és a behatárolást, a helyes alkalmazás beállításkor is számításba kell venni, hogy az alkalmazással visszaélhetnek vagy eltéríthetik. Ezért aztán az alkalmazás biztosíthatóságát nem csak az méri hány ***CERT*** tanácsadóban szerepel. Az alkalmazásnak rendelkeznie kell beépített mérséklési lehetőségekkel is.

Összefoglalás

A biztonság kockázat alapú megközelítése két fontos előnyt jelent számunkra. Először is, ahelyett, hogy egyfolytában nemet mondunk mindenre, az ilyen megközelítést gyakorló biztonsági szakemberek inkább az „igen, ha" kifejezést használhatják (azaz „igen, használhatjuk az eszközt, ha lecsökkentjük X veszélyét, visszatartjuk Y veszélyt" és így tovább). Másodsorban, azáltal, hogy nem csak az ismert veszélyforrásokra koncentrálunk, hanem általánosabb kockázatokat is figyelembe veszünk, a kockázat alapú megközelítés mélyebb védelmet tesz lehetővé, ahol a réteges vezérlés csökkenti az esélyét, hogy egyetlen fenyegetésnek globális következményei legyenek (tűzfal szabályok a chroot-olt alkalmazásokkal valamint behatolásjelző rendszerek használata és egyéb módszerek).

Remélem sikerült hasznos leírást készíteni, ami talán kicsit jobb rálátást ad a rovatunkban felbukkanó más, „inkább drótelvű" biztonsági eszközökre és módszerekre is.

Csak biztonságosan!

Linux Journal 2005. január, 129. szám

Mick Bauer (mick@visi.com)
Biztonsági szakember, a Linux Journal biztonsági témákkal foglalkozó szerkesztője, biztonsági tanácsadó a Minnesota állambeli Minneapolisban található Upstream Solutions LLC Inc.-nél.

Maia Mailguard és amavisd-new a Spam levelek és a vírusok réme

Úgy érzi nem engedheti meg magának, hogy vállalkozása vezető spam és vírusvédelmet használjon? Két jó indok, amiért érdemes ezt újragondolni.

Manapság ahogy a spam és e-mail férgek egyre terjednek, eljött az anti-spam és anti-vírus megoldások szállítóinak aranykora. Az ***Európában*** és az ***USA***-ban bevezetett anti-spam törvények nem túl sok eredményt értek el, így az áldatlan állapotok sok embert késztettek technikai megoldások, spam és vírus szűrők vásárlására.
A tartalom szűrése minden egyes gépen ugyanakkor meglehetősen drága és nem is túl praktikus megoldás. Ideális esetben a spam és vírus problémákat gyökerüknél kell kezelni, így e pont mögött mindenkit megvédhetünk.
Ezt a stratégiát követő szervezeteket minden erőforrásukat egyetlen helyre összpontosítják, mégpedig általában a levelező kiszolgálóra.
A kiszolgáló alapú megoldások azonban ritkán olcsók. A legtöbb ilyen termék esetében levelesládánként kell fizetni, legyen szó levelezőkiszolgáló bővítményről vagy önálló tartalomszűrő alkalmazásról. Ezek a megoldások dollárezrekbe is kerülhetnek és gyakran a vírus- és spam-minták frissítéseiért is éves díjat szednek.
Ebben a cikkben az ***amavisd-new*** nyílt forráskódú tartalomszűrő rendszerrel ismerkedhetünk meg, valamint a projekt egyik hatékony kiegészítésével a ***Maia Mailguard***-dal.

amavisd-new

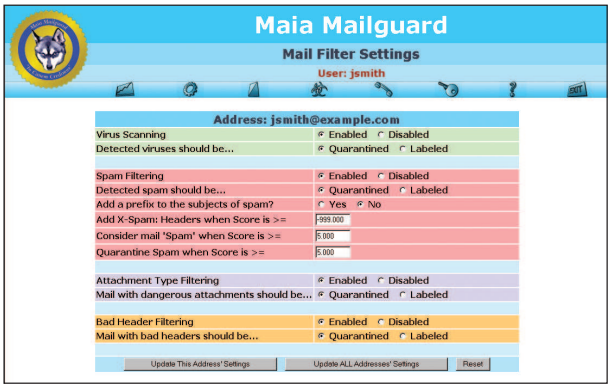
Az ***amavisd-new*** lényegében levelező-szűrő – leveleket fogad a levelezőkiszolgálótól, megállapítja hogy tartalmaznak-e spam-et vagy vírusokat, majd ennek megfelelően karanténba helyezi, elutasítja vagy kidobja a szabálysértő elemeket végül a maradékot a kézbesítést végző másik levelezőkiszolgálóra irányítja. Gyakorlatban az ***amavisd-new*** gyakran található két azonos gépen futó levelezőkiszolgáló között, hiszen, különösen a kisebb helyek esetében, a levelezőkiszolgálót és a tartalomszűrőt praktikus lehet azonos gépen elhelyezni. A nagyobb helyek külön tartalomszűrő gépre telepíthetik az ***amavisd-new***, ***SpamAssassin*** és vírusirtó programjaikat. Még nagyobb helyeken ilyen gépek terheléseloszlásos hálózatát érdemes felépíteni.
Az ***amavisd-new Perl*** nyelven, a biztonságot és megbízhatóságot szem előtt tartva készült és lényegében minden ***UNIX*** rendszeren jól üzemel. Magja az ***RFC***-nek megfelelő levélkezelő, amelyet úgy terveztek, hogy soha ne veszítsen el egyetlen levelet sem. Ennek érdekében az ***amavisd-new*** mindaddig nem veszi át véglegesen a levélelemet amíg az alatt elhelyezkedő levelező kiszolgáló át nem veszi tőle azt. Ez annyit jelent, hogy ha bármilyen hiba történik a levél szűrése közben, a levél nem veszik el, hiszen a felső levelezőkiszolgáló sorában megmarad. Az ***amavisd-new*** négyféle szűrési lehetőséget kínál: virus/malware keresés, spam szűrés, veszélyes csatolmányok és érvénytelen fejlécek tiltása.

Víruskeresés

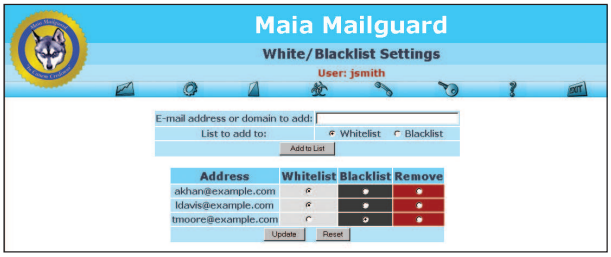
Az ***amavisd-new*** nem víruskereső; inkább egy keretrendszer, amely egy vagy több víruskereső meghívására képes. Több mint 30 népszerű víruskeresőt támogat,többek között olyan üzleti termékeket is mint a ***Sophos***, ***Symantec*** és ***Network Associates***, és persze a nyílt forráskódú ***Clam Antivirus***-t.
A parancssoros és démon-alapú víruskeresők egyaránt támogatottak. Persze a démon alapú keresők sokkal hatékonyabbak mint parancssoros rokonaik. Amennyiben a levelezőkiszolgálónk nagy mennyiségű levelet dolgoz fel, nem valószínű, hogy szívesen töltenénk be a memóriába egy parancssoros víruskeresőt minden egyes levélhez, amit aztán eltávolítunk onnan. A démon alapon futó víruskeresők csak egyszer töltődnek be, végig a memóriában maradnak, így az egész folyamat sokkal gyorsabb.
Amennyiben több víruskeresőnk is van, elsődleges és másodlagos csoportokba sorolhatjuk őket. A másodlagos csoporthoz akkor nyúl a rendszer, ha egyik elsődleges kereső sem érhető el.

Spam szűrés

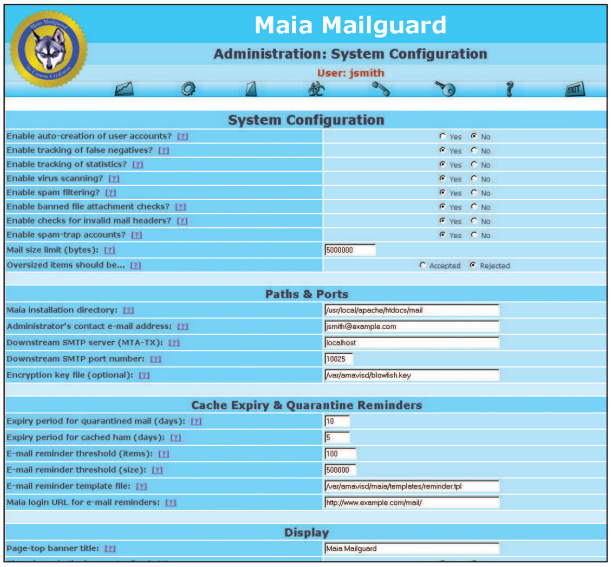
Az ***amavisd-new*** a spamszűrést a ***SpamAssassin*** beépítésével végzi. Az ***amavisd-new*** a címzettek számától függetlenül minden egyes levélhez egyszer hívja meg a ***SpamAssassint***, így a levelezőlisták vizsgálata sem foglal több erőforrást mint az egy címzettel rendelkező levelek.
A ***SpamAssassin*** a spamszűrő módszerek széles skáláját vonultatja fel: képes jellegfelismerésre, ***DNSBL*** és ***SPF*** keresésre, kezeli a ***együttműködésen alapuló jelentési hálózatokat (collaborative reporting networks)*** és a ***Bayes***-féle tanuló



1. ábra Minden e-mail cím saját tartalomszűrő beállításokkal rendelkezik

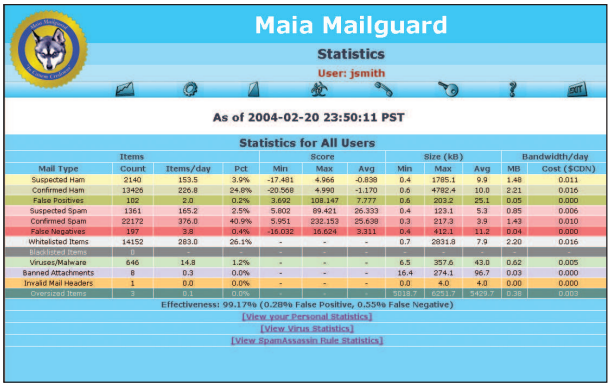


2. ábra A felhasználók saját fehér- és feketelistákat tarthatnak fenn

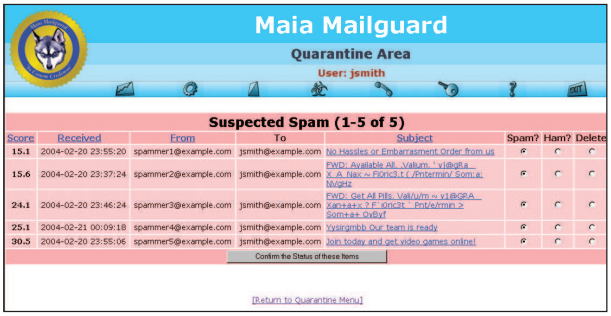


3. ábra A rendszergazda a legtöbb teljes körű beállítást elérheti a webes felületről

módszert. Minden teszt megállapít valamilyen számszerű értéket, amelyet aztán levelenként összegezünk, a felhasználók pedig tetszőleges határértéket állíthatnak be, ezáltal eldöntve, hogy mit tekintenek spam-nek és ham-nek. (A „ham" szó spamszűrő körökben a nem-spam levelet jelenti azaz a spam ellentéte, a *Monty Python* féle spam (löncshús) mintájára. a *fordító*) Ez elég hatékony megoldás, hiszen az egyik módszer gyengéit a többi módszer javítja. A jellegfelismerő a levél fejlécét és a levéltestét vizsgálja olyan nyomok után kutatva, melyeket az emberek spam vagy ham (nem-spam levél) jellegzetességeknek



4. ábra A szűrők által látottakat statisztikák foglalják össze



5. ábra A felhasználói karantén a spam-pontszám szerint rendezett

tartanak. Az a tény például, hogy a levél ***Date:*** fejléce 12 órával a jövőbe mutat vagy, hogy a levél szöveget nem, csak képet tartalmaz, könnyen utalhat spam jelenlétére, míg egy több mint ezer szót tartalmazó levél valószínűsíthetően ham.

A ***SpamAssassin*** képes ellenőrizni a csatlakozó levelező-kiszolgáló vagy ügyfél IP címét, majd ellenőrizni szerepel-e valamelyik ***DNS***-alapú tiltólistán (***DNSBL***), így képes meghatározni, hogy a küldő esetleg ismert spam forrás.

A ***DNSBL*** listák hagyományos alkalmazásával szemben azonban a ***SpamAssassin*** nem feltételezi, hogy a listán való szereplés önmagában végzetes bizonyíték; egyszerűen csak megnöveli a levél teljes pontszámát. Ez sokkal rugalmasabb megközelítés és lehetőséget ad arra, hogy külön beállítsuk valamennyi ***DNSBL*** pontszámát, attól függően mennyire bízunk meg listában és karbantartóinak rendszabályaiban.

A hamarosan megjelenő ***SpamAssassin 3.0*** már támogatja a ***Sender Policy Framework (SPF)*** kereséseket, amely megpróbálja azonosítani, hogy a kapcsolódó gépnek van e joga levelet küldeni az adott tartomány neve alatt.

Az együttműködésen alapuló jelentési hálózatok, mint a ***Vipul Razor***-ja, ***Pyzor*** és a ***Distributed Checksum Clearinghouse (DCC)*** egy másik tájékozódási forrást kínálnak a ***SpamAssassin*** részére. Az alapötlet az, hogy miután a spam leveleket fogadók millióinak küldik el, mire megkapjuk a nekünk szólót, rengeteg ember kap többé kevésbé hasonló leveleket. Amennyiben elég sok ember jelezte hogy az adott levél spam a mi spamszűrőnk is felhasználhatja ezt a tényt saját döntésfolyamatában.

Végül, de természetesen nem utolsósorban, a ***SpamAssassin Bayes***-féle tanuló mechanizmust is tartalmaz, amely lénye-

6. ábra A levélnézegetővel biztonságosan megvizsgálhatjuk a gyanús leveleket

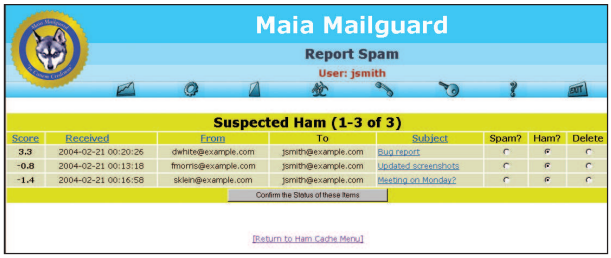
gében automatizált jellegfelismerő. Míg a korábban ismertetett jellegfelismerő rendszer az emberekre bízza a spam és ham közötti felismerhető különbségek azonosítását, a ***Bayesian*** megközelítés magától próbálja meg felderíteni ezeket a jellegzetességeket a korábban kapott spam és ham levelek vizsgálata alapján.

Veszélyes csatolmánytípusok tiltása

Üzembiztossági megfontolásokból gyakran jó ötlet megakadályozni a végrehajtható csatolmányokkal érkező leveleket, még ha a víruskeresőink szerint tiszták is. Végül is a víruskeresők sem tökéletesek, ráadásul a legfrissebb gonoszságok könnyen elérhetik a rendszerünket még mielőtt az anti vírus forgalmazó elkészítené a felderítéséhez szükséges új vírusmintát. Az ***amavisd-new*** lehetővé teszi, hogy megadjunk fájlkiterjesztés listákat, tartalomosztályokat és ***MIME***-típusokat, amelyeket karanténba szeretnénk zárni, elutasítunk vagy törölni kívánunk.

Érvénytelen levélfejlécek kezelése

Az ***RFC 2822*** szabvány szerint, a levélfejlécekben nem szerepelhet semmilyen 127-nél nagyobb értékű karakter, ***NULL*** vagy önmagában álló soremelés. Ezen a tartományon kívüli karakterek különleges kódolásúak lehetnek, így a világ különféle levelezőprogramjai probléma nélkül tudják értelmezni őket. Ha érvénytelen fejléccel érkező levelet kapunk, az lehet egy gyengén megírt levelezőügyfél hibája is, de igen gyakran olyan egyedi tervezésű program terméke, amilyeneket a spammerek használnak a tömeges levelezéshez. Az ilyesfajta, úgynevezett ***ratware*** (rat=patkány) készítői általában angol anyanyelvűek, és többnyire nem is gondolnak rá, hogy programjukat más nyelveken beszélők is használni fogják. Amikor a spammerek megpróbálják ezekkel a programokkal elküldeni a leveleiket, a ratware nem kódolja a különleges karaktereket, így hibás levélfejléceket készít. Az amavisd-new programban eldönthetjük, mit kívánunk tenni az érvénytelen fejlécű levelekkel: karanténba zárjuk, elutasítjuk, elvetjük, vagy keresztülengedjük.



7. ábra A ham gyorstár segítségével a felhasználó egyszerűen jelezheti az átcsúszott elemeket

Tartalomszűrő házirend beállítása

Az ***amavisd-new*** segítségével a rendszergazdák rendszer szintű tartalomszűrő rendszabályokat készíthetnek, ám ezeket a szabályokat tartomány illetve felhasználói szinten felülbírálhatjuk. Vannak felhasználók akik szeretnék átvizsgáltatni leveleiket mind a négy gyanús tartalomtípus (vírusok, spam, tiltott fájlok és érvénytelen fejlécek) szerint, mások viszont inkább kikapcsolnák valamelyik vagy akár valamennyi ellenőrzést. Az egyik felhasználó szeretné, ha az 5.0 vagy ennél nagyobb értéket elérő levelei a karanténba kerülnének, míg mások inkább azt részesítenék előnyben, ha a ***Subject:*** fejléc bővülne egy speciális előtaggal (például ***SPAM***), amennyiben a pontszám eléri a 4.0-t és csak akkor blokkolnák, ha 8.0-at is meghaladja. Ez a fajta finom felbontású vezérlés a teljes szűrőfolyamat alatt lehetővé teszi, hogy a rendszergazdák eltérő igényekkel rendelkező felhasználók széles körét szolgálják ki.
Ezen felül az ***amavisd-new*** mindhárom szinten fehér és feketelistákat tárol. Ezáltal a rendszergazdák rendszerszintű listákat hozhatnak létre; míg a rendszer másik végén a felhasználók saját egyedi listákat hozhatnak létre.

Karantén és figyelmeztetési lehetőségek

Előre meghatározhatjuk milyen lépéseket végezzen el az ***amavisd-new*** amikor megállít egy e-levelet. A levelet tárolhatjuk a karantén könyvtárban vagy levelesládában, vagy akár felhasználónkénti külön levelesládákban (pl. jozsi+spam). Választhatjuk az is, hogy az ***amavisd-new*** utasítsa el a levelet, azaz vagy ne fogadja el a felette elhelyezkedő kiszolgálótól vagy csendben dobja ki azt. Amennyiben a szervezeti szabályzatunk előírja, hogy a felhasználókat figyelmeztetni kell a blokkolt levelekkel kapcsolatban, az ***amavisd-new***-ban ez is beállítható. Ez azonban vitatott téma. Manapság sok ember a vírusfigyelmeztetéseket és spam reklamációkat inkább idegesítőnek mint hasznosnak tartja , különösen amióta a levelek feladója általában hamisított. Amennyiben mégis kellene küldenünk vírusfigyelmeztetéseket, az ***amavisd-new*** tartalmaz egy listát azokról a vírusokról, amelyek bizonyítottan meghamisítják a levélfejléceket így ilyenkor nem küld figyelmeztetéseket sem. Ezt a listát kézzel kell karbantartanunk és meg kell egyeznie azokkal a nevekkel, melyet az általunk használt víruskereső visszaad. Amennyiben egyszerűbbnek találjuk felsorolni azokat a vírusokat, amelyek nem hamisítanak címet, használhatunk inverz listát is.

Maia Mailguard

A ***Maia Mailguard*** egyszerű ***amavisd-new*** webes felületként született pályafutását, amely lehetővé teszi a felhasználóknak, hogy egy kényelmes felületen keresztül változtassák meg tartalomszűrő beállításaikat és kezeljék karanténjüket. A projekt egyre népszerűbb lett az ***ISP***-k, ***Web-mail*** szolgáltatók és a lapon kívüli szűrést kínáló cégek körében, így ezek a nagyobb igényű felhasználók hatására a ***Maia Mailguard*** hamarosan sokkal kifinomultabb rendszerré alakult.

A ***Maia Mailguard*** teljes értékű spam és vírus kezelő rendszer, amely ***PHP***, ***SQL*** és ***Perl*** parancsfájlokat, ***MySQL*** vagy ***PostgreSQL*** adatbázist valamint természetesen az ***amavisd-new***, ***SpamAssassin*** rendszert és egyéb támogatott víruskeresőket tartalmazhatja. Több Tartalomszűrő kezelhető egyetlen ***Maia*** csatolófelületről, amelyek egyazon ***SQL*** adatbázison osztoznak. Tekintve, hogy a tartalomkezelés, a karantén karbantartás és spamjelentések leegyszerűsítésére tervezték, a ***Maia Mailguard*** sok szempontból új eszköz a levelező felhasználók kezében.

Webes felület

A ***Maia*** web-alapú felülete több forrás szerinti azonosítást is lehetővé tesz, használhatunk ***POP3*** vagy ***IMAP*** kiszolgálót, ***LDAP*** kiszolgálót, külső ***SQL*** adatbázist vagy a ***Maia*** saját belső adatbázisát. A felhasználókat a rendszergazda felveheti kézzel vagy is felkerülhetnek amikor olyan levél érkezik helyi címre melynek címzettjével a ***Maia*** még nem találkozott azelőtt.

A felhasználóknak több e-mail címe is tartozhat egyetlen azonosítóhoz, de minden e-mail cím saját tartalomszűrő beállítással rendelkezik (*1. ábra*). A felhasználók a webfelületen keresztül címeket adhatnak hozzá és távolíthatnak el a fehér- illetve feketelistáikról (*2. ábra*), míg a rendszergazdák egy másik weblapkészleten tartomány illetve rendszer szintű beállításokat módosíthatják (*3. ábra*). Az amavisd-new mind a négy levéltípusáról statisztikák készülnek, nem különben a fehér- és feketelistára kerülő elemekről, túlméretes elemekről, ***hamis találatokról (false positives)*** és ***átcsúszásokról (false negatives)*** (*4. ábra*). Más táblázatok típusonként nyomon követik az egyes vírusokat valamint azt is, hogy mely ***SpamAssassin*** szabály hány alkalommal aktiválódott. Az adatokból valós időben grafikus ábrák készíthetők vagy adott időközönként statikus lapként elmenthetők.

Annak köszönhetően, hogy a ***Maia*** a karanténkezelést és a tartalomszűrő vezérlést egyenesen a felhasználók kezébe helyezi, a rendszergazdáknak nem sok naponta elvégezendő munkájuk marad. A ***Maia*** adott időben lefutó ***Perl*** parancsfájljaival kiegészítve, melyek jelentik a felhasználók által jóváhagyott spam leveleket és kezelik a lejárt karantén elemeket, a rendszer szinte önmagát működteti.

Karanténkezelés

Felhasználói szemszögből igen fontos, hogy amikor egy levél a karanténba kerül, könnyedén el tudjuk érni azt. A ***Maia*** a felhasználói karanténjában megtekinthetjük az elemek listáját, mégpedig spam pontszám szerint rendezve. Tehát azok az elemek amelyek a legnagyobb valószínűséggel tévedésből kerültek ide – azaz a hamis találatok – a lista tetején foglalnak helyet, így könnyebb észrevenni őket (*5. ábra*).

Amennyiben a fejléc alapján nem tudjuk eldönteni, hogy egy levél kívánatos-e vagy sem, a címre kattintva megtekinthetjük azt a ***Maia*** levélnézegetőjében (*6. ábra*). A levélnézegetőt bármilyen típusú levélen biztonságosan használhatjuk, ugyanis a legtöbb csatolmányt nem dekódolja, viszont letiltja a távoli képeket és kiszedi azokat a ***HTML*** tagokat amelyek más lapra irányíthatnának át bennünket. A levelet dekódolt vagy nyers formában is megtekinthetjük, valamennyi eredeti levélfejléccel együtt.

Amennyiben úgy döntünk, hogy a levél végül mégiscsak kívánatos, egy kattintással kimenthetjük a karanténból és elküldhetjük magunknak. Ezzel egy időben a ***Maia*** értesíti a ***SpamAssassin***-t a hibáról, így a ***Bayesian*** tanulórendszer kisebb valószínűséggel követi el még egyszer ugyanezt a hibát. A ***Maia***-t úgy is beállíthatjuk, hogy az ilyesfajta kimenekítések esetén a küldő címét automatikusan felvegye saját fehér listánkra.

A karanténon kívül a ***Maia*** rendelkezik egy úgynevezett ***ham gyorstárral*** is, amelyben tulajdonképpen a mostanában kapott elfogadott leveleink találhatók (*7. ábra*). A ham gyorstár célja, hogy könnyedén jelenthessük a szűrőn átcsúszott spam leveleket (false negatives). Ezeket az elemeket helyesen spamnek kijelölve taníthatjuk a ***SpamAssassin Bayes***-féle szűrőjét. A karantén és a ham gyorstár egyaránt a kapott levelek állapotának hitelesítésre szolgál. Ezzel nem csak a ***Bayes***-féle tanuló rendszert oktatjuk, hanem egyúttal lehetővé tesszük a spam levelek helyes beazonosítását is, hiszen az intézkedést ember hagyja jóvá.

Spam jelentés

A legtöbb spamszűrő csak a spam támadások kivédésre koncentrál és nem sokat, vagy egyáltalán nem törődik azok megelőzésével. Minthogy a ***Maia*** lehetővé teszi felhasználóinak, hogy spam-ként azonosítsák a leveleiket miközben az eredeti levélfejléceket nem módosítja, a spam több különféle módon is jelenthető. A ***Maia*** következő verziói képesek lesznek részletes fejlécanalízist végezni és félautomata jelentéseket küldeni az ***ISP***-k részére. Ezek a jelentések segítenek másoknak hatékonyabban kiszűrni a spam leveleket végül egyfajta büntetést jelenthetnek a spammelő számára.

A színfalak mögött a ***Maia*** automatikus parancsfájljai szabályos időközönként feldolgozzák a karantént, jelentik a helybenhagyott spam leveleket a ***SpamAssassin*** által használt együttműködésen alapuló hálózatoknak (***Vipul's Razor***, ***Pyzor*** és ***DCC***). Azzal, hogy megosztjuk az információt ezekkel a hálózatokkal, valamit adunk is és nem csak nyerünk mások jelentéseiből.

Hatékony, teljes körű megoldás

Végül, szóljunk pár szót arról ami a leginkább számít, vajon mennyire hatékony az ***amavisd-new*** és a ***Maia Mailguard*** a spam levelek elfogása terén, és milyen eséllyel kerülik el ham leveleink a karanténba kerülést? Saját helyem statisztikáiból ollózva, ez az érték szimpatikus 99.22%, 0.26% hamis találati (false positive) és 0.52% átcsúszó levél (false negative) értékkel. Ami a legjobb, ezeket a hamis találatokat könnyedén kigyűjthetjük a karanténból míg az átcsúszott leveleket jelenthetjük a ham gyorstárban.

A vírusok és egyéb rossz szándékú küldemények terén a hatékonyság még figyelemreméltóbb: 100%. Az alatt a hat hónap alatt amióta ezt a tartalomszűrő megoldást telepítettem, az asztali gépekre feltett víruskeresők semmit sem fogtak ami átcsúszott volna a tartalomszűrőn. Persze ebben nagy szerepe van annak is, hogy az ***amavisd-new*** több eltérő gyártótól származó víruskereső együttes használatát is lehetővé teszi – amit az egyik kereső elhibáz a másik általában elkapja.

Ha a teljesítményt nézzük, minden tartalomszűrő megoldás lelassítja valamilyen mértékben a levélfeldolgozást. Gyakran előfordul, hogy a nagyobb sebesség érdekében engedményeket teszünk a szűrő hatékonyság rovására, és inkább kikapcsolunk néhány szűrőt és tesztet ezzel növelve az átviteli teljesítményt. A nálam mért 99.22%-os hatékonyság eléréséhez például valamennyi szűrő bekapcsolt állapotban volt, ennek megfelelően 1–3 másodpercre volt szükség minden egyes levélelem átvizsgálásához egy közepesen terhelt dual-PIII 733MHz-es gépen 1GB memória mellett. Egy elfoglaltabb helyen elképzelhető, hogy ekkora késlekedés már nem elfogadható. Ők vagy kikapcsolják az időigényesebb teszteket, vagy gyorsabb processzort és több RAM-ot vásárolnak a tartalomszűrőhöz esetleg terheléselosztáson alapuló hálózatot építenek több tartalomszűrőből. Ettől függetlenül több mint 50.000 felhasználónak helyet adó rendszerek is használnak ***Maia Mailguard*** és ***amavisd-new*** párost, ahol több mint 350.000 e-levelet kezelnek naponta, tehát a megoldás skálázható, amennyiben elegendően jó alkatrészek állnak rendelkezésünkre.

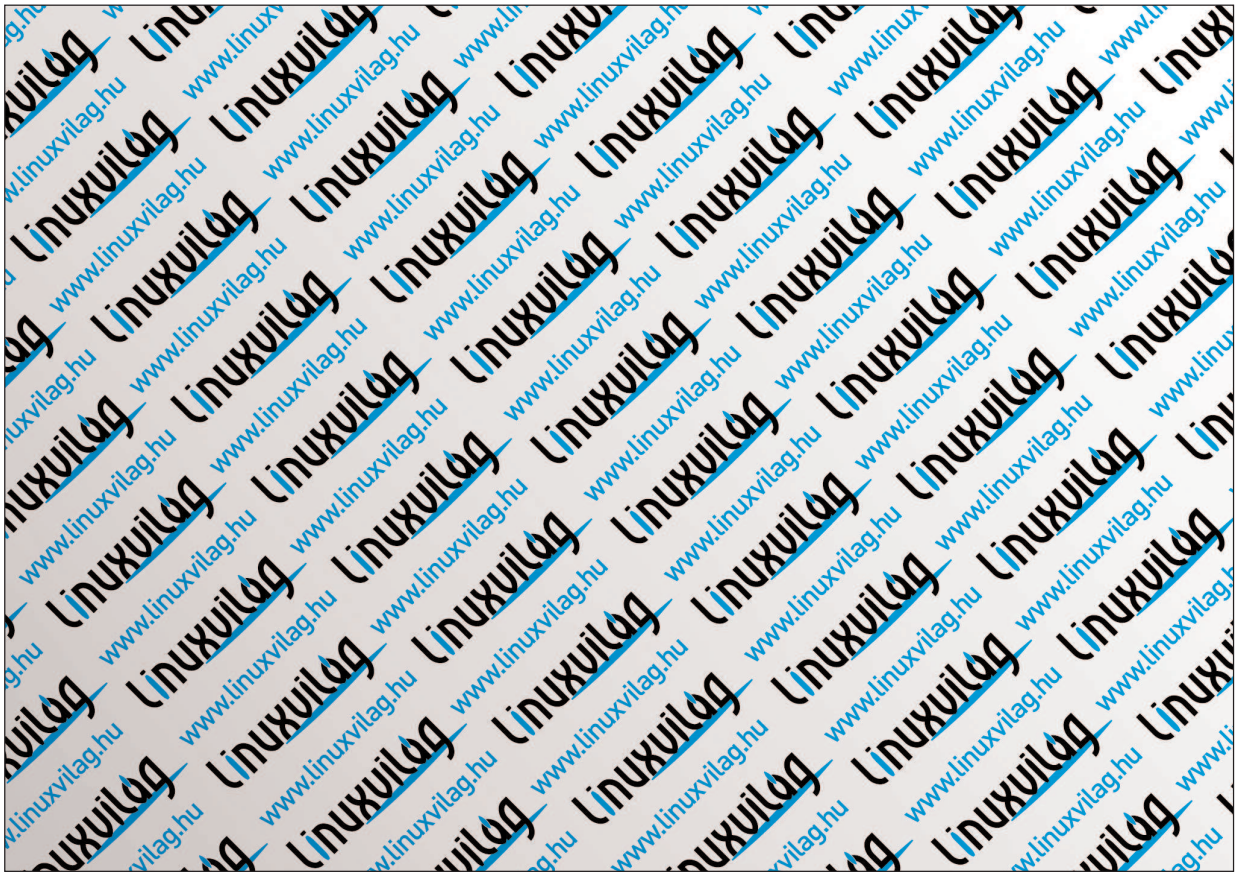
Mint azt valószínűleg sokan megfigyelték, a spam és vírusok elleni harcban éppen a nyílt forrású eszközök a legjobb fegyverek. Az ***amavisd-new***, ***Maia Mailguard***, ***SpamAssassin*** és a ***Clam Antivirus*** segítségével hálózatunknak elsőrangú védelmet adhatunk anélkül, hogy hatalmas költségekbe vernénk magunkat.

Linux Journal 2004. december, 128. szám

Robert LeBlanc a Renaissoft elnöke (www.renaissoft.com), a Maia Mailguard szerzője, és az AnswerSquad spam-harcos guruja (www.answersquad.com). Amikor éppen nem a spanyolviaszt találja fel vagy még jobb egércsapdákat fabrikál, általában négy Alaszkaija Klee Kai, Zorro, Sikari, Piyomi és persze a Maia társaságában található.

KAPCSOLÓDÓ CÍMEK

- www.ijs.si/software/amavisd
- www.renaissoft.com/maia
- www.spamassassin.org
- www.clamav.net
- razor.sourceforge.net
- pyzor.sourceforge.net



Cikkgyűjtemények tartalmának begyűjtése

Adjunk lehetőséget alkalmazásunknak, hogy kedvenc weblapjainkról összegyűjtse a friss írásokat!

Az elmúlt néhány hónapban az *RSS* és *Atom XML*-alapú fájlformátumokkal foglalkoztunk, azzal a párossal, mellyel könnyedén elkészíthetjük és elterjeszthetjük egy weboldal összefoglalóját. Igaz ugyan, hogy – mint az ismeretes – az ilyesfajta cikkgyűjtemények készítése hagyományosan inkább a Weblogokra és híroldalakra jellemző, egyre nagyobb az érdeklődés más területeken is. Bármilyen web alapú információforrás esetében érdekes és hasznos jelölt lehet akár az *RSS* akár az *Atom*.

Ez idáig azt néztük meg, hogyan készíthetünk *RSS* és *Atom* tartalmat Weblapunkhoz. Természetesen az cikkgyűjtemény tartalom elkészítése csak az egyenlet fele. Legalább ilyen fontos és talán még hasznosabb megérteni, hogyan szerezhetjük be és használhatjuk ezeket az cikkgyűjtemény tartalmakat saját webhelyünkről és más érdekes oldalakról.

Amint azt láthattuk, három különböző cikkgyűjtemény tartalom létezik: az ***RSS 0.9x*** és fejlettebb verziója az ***RSS 2.0,*** a nem együttműködő ***RSS 1.0***, valamint az ***Atom***. Valamennyi nagyjából azonos feladatot végez el és elég nagy mértékű átfedés van a szabványok között. Ugyanakkor, a hálózati protokollok általában nem működnek valami fényesen, ha naivan feltételezzük, hogy minden rendben lesz és elegendően azonos működésű. Ez alól az cikkgyűjtemények sem kivételek. Amennyiben valamennyi összesítővel rendelkező lapot el szeretnénk tudni olvasni, az összes protokollt és azok verzióit meg kell értenünk. Az *RSS* például jelenleg kilenc különféle verzióval rendelkezik amelyhez ha az *Atom*-ot is hozzávesszük, összesen tíz különféle cikkgyűjtemény formát használhat valamely weblap. A legtöbb eltérés valószínűleg elhanyagolható, de nem lenne bölcs dolog teljesen figyelmen hagyni őket, vagy feltételezni, hogy mindenki a legfrissebb verziót használja. Ideális esetben van valamilyen modulunk vagy eszközünk amely több különféle protokoll nyelvén is ért eltüntetve a különbségeket, miközben valamennyi protokoll egyedi előnyeit kihasználja.

E hónapban ***Mark Pilgrim univerzális tartalom értelmezőjét (Universal Feed Parser)*** vesszük szemügyre, amely e problémára kínál nyílt forráskódú megoldást. ***Pilgrim*** jól ismert weblog szerző és ***Python*** programozó, valamint az ***Atom*** cikkgyűjtemény formátum megalkotásának egyik kulcsfigurája. Ez persze nem is meglepő, ha azt vesszük mennyi problémába ütközött az ***Universal Feed Parser*** megíráskor. A program a ***Microsoft*** üzleti ***CDF*** formátumát is kezeli, amely az aktív asztal és programfrissítések összesítőit terjeszti. A ***Linux*** asztali felhasználók számára ez a rész talán nem túl izgalmas viszont érdekes egységesítési lehetőséget kínál a már telepített ***Microsoft*** rendszerekkel. Az ***Universal Feed Parser*** jelenleg a 3.3 verziónál tart, saját kategóriájában a legjobbnak számít nyelvtől és licensztől függetlenül.

A feedparser telepítése

A ***feedparser*** telepítése rendkívül egyszerű. Töltsük le a legfrissebb verziót, tegyük a telepítési könyvtárába és gépeljük be a

```
python setup.py installc
```

parancsot. Ezzel elindítjuk a ***Python*** szabványos telepítési eszközét, amely a ***feedparser***-t ***a Python site-packages*** könyvtárába helyezi. A ***feedparser*** telepítése után próbáljuk ki a ***Python*** segítségével valamelyik héjablakból:

```
>>> import feedparser
```

A >>> jelek az interaktív módban meghívott ***Python*** szabványos parancssorának felelnek meg. A fenti utasítás beolvassa a ***feedparser*** modult a ***Pythonba***. Ha nem telepítettük a ***feedparser***-t, vagy valamilyen gond történt telepítés során, a parancs végrehajtása ***Python ImportError*** hibát okoz.

Miután importáltuk a modult a memóriába, nézzünk belea legfrissebb hírekbe a ***Linux Journal w***eblapján. Gépeljük be:

```
>>> ljfeed = feedparser.parse
("http://www.linuxjournal.com/news.rss")
```

Nem szükséges megadnunk milyen protokoll vagy verzió szerinti tartalmat szeretnénk az értelmezővel feldolgoztatni – a csomag elég okos ahhoz, hogy magától felismerje a verziókat, még olyankor is amikor az *RSS* tartalom maga nem tudja beazonosítani a verzióját. Írásunk születésekor a LJ lapja ***PHPNuke*** alapú az tartalmat használ, melyet egyértelműen ***RSS 0.91*** típusként sikerült azonosítani.

Miután letöltöttük az új tartalmat, megnézhetjük pontosan hány bejegyzést kaptunk, amit többé kevésbé a kiszolgáló beállításai határoznak meg:

```
>>> len(ljfeed.entries)
```

természetesen, az elemek száma kevésbé érdekes mint az elemek maguk, ezeket egy egyszerű `for` ciklussal kaphatjuk meg:

```
>>> for entry in ljfeed.entries:
...     print entry[´title´]
...
```

Ne feledjük, a ciklus sorait beljebb kell kezdenünk, hogy a ***Python*** tudja mely sorok tartoznak a ciklusba. Aki még csak most ismerkedik a ***Python*** nyelvvel, esetleg furcsának találja a `...` jelzéssel kezdődő sorokat amelyek azt mutatják, hogy a ***Python*** készen áll és a `for` ciklus utáni sorokra várakozik. A ciklus végrehajtásához egyszerűen csak nyomjunk ENTER-t és máris láthatjuk az összes friss címet.
Csinos átnézeti képet kaphatunk az ***URL***-ekről és a címekről a ***Python*** karaktersorozat behelyettesítésének segítségével:

```
>>> for entry in ljfeed.entries:
...     print ´<a href="%s">%s</a>´ % \
...     (entry[´link´], entry[´title´])
```

Mint korábban jeleztem, a `feedparser` megpróbálja feloldani a különböző verziók közötti különbségeket, és úgy dolgozhatunk a különböző összesítési formákkal mintha azok nagyjából azonos módon működnének. Ezért a fenti parancsokat saját weblogom cikkgyűjtemény tartalmára is megismételhetem. Nemrég tértem át a ***WordPress***-re, amely ***Atom*** tartalmat használ:

```
>>> altneufeed = feedparser.parse(
... "http://altneuland.lerner.co.il/wp-atom.php")
>>> for entry in altneufeed.entries:
...     print ´<a href="%s">%s</a>´ % \
...     (entry.link, entry.title)
```

Figyeljük, meg hogy ez az utóbbi példa az `entry.link` és `entry.title` attribútumokat használja, míg a korábbi példánk az `entry['link']` és `entry['title']` könyvtárhivatkozásokat alkalmazta. A ***feedparser*** rugalmasan próbálja megközelíteni a problémát, és egyazon információhoz többféle elérési lehetőséget kínál, kiszolgálva a különböző igényeket és stílusokat.

Milyen friss a hír?

A hírösszesítők és más ***RSS*** vagy ***Atom*** használó alkalmazások fő célja a nemrégiben frissített hírek összegyűjtése és bemutatása. Az hírösszesítő csak azokat a híreket szolgáltathatja amelyeket a kiszolgáló felajánl. Amennyiben az ***RSS*** tartalom csak a két legutóbb frissített elemet tartalmazza, az hírösszesítő felelőssége begyűjteni, gyorstárazni, és megjeleníteni azokat az elemeket, amelyek már nem kerülnek az összesítőkbe.

Ez két eltérő, bár szoros kapcsolatban álló kérdést vet fel: Hogyan biztosíthatjuk, hogy az összesítő csak olyan elemeket mutasson, amelyeket még nem láttunk? Van-e lehetőség rá, hogy az hírösszesítőnk csökkentse a terhelést a weblog kiszolgálón, és csak azokat a bejegyzéseket töltse le amelyek a legutóbbi frissítésünk óta kerültek fel? Az első kérdésre a válasz, hogy minden egyese elemnél ellenőriznünk kell a módosítási dátumot, amennyiben az létezik.
A második kérdés jelenleg egyre népszerűbb vitatéma a webes közösség berkeiben. Ahogy a weblog népszerűsége növekszik, úgy nő az egyes cikkgyűjteményekre feliratkozó emberek száma is. Ha a weblog cikkgyűjtemény tartalma 500 feliratkozót szolgál ki, és az összes feliratkozott óránként gyűjti be a frissítéseket, az óránként további 500 lekérdezést jelent amit a Webkiszolgálónak teljesíteni kell. Ha a cikkgyűjtemény tartalom a teljes webhelyet lefedi ez igen komoly elpazarolt sávszélességet is jelenthet – ami csökkenti az oldal válaszidejét más látogatók számára, illetve előfordulhat, hogy az oldal tulajdonosának fizetnie kell a engedélyezett havi sávszélesség túllépése miatt.
A ***feedparser*** használatával önmagunkhoz és az összefoglaló kiszolgálóhoz is kíméletesek lehetünk, ugyanis lehetővé teszi, hogy csak a akkor töltsük le az cikkgyűjtemény tartalmakat, ha van figyelemre méltó újdonság. Ez úgy lehetséges, hogy a ***HTTP*** modern verziói lehetővé teszik, hogy a kérelmező ügyfél beállítsa az ***If-Modified-Since (adott időpont óta módosult)*** fejlécet, amit a dátum követ. Amennyiben a kért ***URL*** módosult a megadott dátum óta, a kiszolgáló az ***URL*** tartalmával válaszol. Ha viszont a kért ***URL*** változatlan, a kiszolgáló a 304-es válaszkódot küldi, jelezve, hogy az aktuális tartalom továbbra is a korábban letöltött verzió.
Mindezt úgy érhetjük el hogy egy lehagyható módosító paramétert adunk át a `feedparser.parse()` függvénynek. Ez a paraméter a ***time*** modul szerint megadott szabványos ***Python*** szerkezet, melyben az első hat elem az év, hónap szám, nap szám, óra, perc és a másodperc. Az utolsó három elem minket jelenleg nem érdekel így nyugodtan hagyhatjuk őket 0 értéken. Tehát, amennyiben a 2004 Szeptember 1. óta beérkezett tartalmakat akarjuk látni, a következőket gépeljük be:

```
last_retrieval = (2004, 9, 1, 0, 0, 0, 0, 0, 0)
ljfeed = feedparser.parse(
          "http://www.linuxjournal.com/news.rss")
```

Ha a ***Linux Journal*** kiszolgálója helyesen van beállítva a fenti kód a teljes cikkgyűjtemény tartalmat letölti az ***ljfeed***-be és vagy a `HTTP OK` állapot üzenetet kapja vissza (melynek számszerű kódja 200) vagy egy jelzést, miszerint a az adat nem változott az utolsó letöltés óta (amelynek számszerű kódja 304). Igaz ugyan, hogy ha tárolni szeretnénk mikor kértük le az utolsó cikkgyűjtemény tar talmat, több bejegyzést kell nyilvántartanunk, mégis fontos betartanunk ezt a szabályt, különösen ha rendszeresen kérünk tartalom frissítéseket, ugyanis ellenkező esetben alkalmazásunk nem igazán lesz szívesen látott vendég bizonyos honlapokon.

1. lista aggregator.py

```
#!/usr/bin/python

import feedparser
import sys

# ------------------------------------------------
# a személyes tartalom kimeneti állomány
# megnyitása

aggregation_filename = “myfeeds.html”
max_title_chars = 60

try:
    aggregation_file =
open(aggregation_filename,”w”)
    aggregation_file.write(“””<html>
<head><title>My news</title></head>
<body>”””)
except IOError:
    print “Error: cannot write ´%s´ “ % \
    aggregation_filename
    exit

# ------------------------------------------------
# A feeds.txt minden nem üres sora
# tartalom forrás.

feeds_filename = “feeds.txt”
feeds_list = []

try:
    feeds_file = open(feeds_filename, ´r´)
    for line in feeds_file:
        stripped_line = line.strip().rstrip()

        if len(stripped_line) > 0:
            feeds_list.append(stripped_line)
            sys.stderr.write(“Adding feed ´” + \
            stripped_line + “´\ n”)

    feeds_file.close()

except IOError:
    print “Error: cannot read ´%s´ “ %
    ➥feeds_filename
    exit

# ------------------------------------------------
# végiglépdelünk a feeds_list listán,
# és leszedjük a megfelelő tartalmat

for feed_url in feeds_list:
    sys.stderr.write(“Checking ´%s´...” %
    ➥feed_url)
    feed = feedparser.parse(feed_url)
    sys.stderr.write(“done.\ n”)

    aggregation_file.write(´<h2>%s</h2>\ n´ % \
                           feed.entries[0].title)

    # Végiglépdelünk az tartalom összes
    # bejegyzésén, megjelenítjük és beírjuk
    # az összefoglalóba
    for entry in feed.entries:
        sys.stderr.write(“\ tWrote: ´%s´” % \

entry.title[0:max_title_chars])

        if len(entry.title) > max_title_chars:
            sys.stderr.write(“...”)

        sys.stderr.write(“\ n”)

        aggregation_file.write(
           ´<li><a href=”%s”>%s</a>\ n´ %
           (entry.link, entry.title))

    aggregation_file.write(´</u2>\ n´)

# ------------------------------------------------
# HTML befejezése

aggregation_file.write(“””</body>
</html>
“””)
aggregation_file.close()
```

Munka a tartalmakkal

Most már van némi elképzelésünk róla, hogyan kell dolgozni a ***feedparser***-el, készítsünk hát egy egyszerű összesítő eszközt. Eszközünk bemenetét a ***feeds.txt*** állományból kapja kimenetét pedig a ***feeds.html HTML*** állományban adhatjuk meg. A programot a `cron` eszközzel futtatva és naponta belenézve a ***HTML*** állományba egyszerű de működőképes hírtartalmat kaphatunk a minket leginkább érdeklő honlapokról.

A ***feeds.txt*** állomány a tényleges tartalmak ***URL***-jeit tartalmazz és nem a honlapok címét ahonnét az abrakot le szeretnénk tölteni. Más szóval a felhasználóra bízzuk, hogy megtalálja és bevigye minden egyes tartalom ***URL*** címét.

A kifinomultabb összesítő eszközök általában képesek beazonosítani a tartalom ***URL*** címét a a webhely honlapjának fejlécében található `link` tag alapján.

Ráadásul, a korábbi figyelmeztetésem ellenére, miszerint minden hírösszesítőnek követnie kellene a legfrissebb híreket hogy ne terhelje túl a kiszolgálókat, ez a program nem tartalmaz ilyen képességet, hiszen megpróbáltam a tömörségre és olvashatóságra törekedni.

Az *1. listában* olvasható ***aggregator.py*** nevű program négy részre bontható:

1. Először is megnyitjuk a kimeneti fájlt, azaz a ***myfeeds.html*** nevű ***HTML*** formátumú szöveges állományt. Ezt a z állományt Webböngészőben való nézegetésre szánjuk.

Ezt a helyi, `file:///` ***URL***-el megadott állományt akár fel is vehetjük a személyes könyvjelzőink közé, vagy esetleg beállíthatjuk kezdőlapnak is. Miután meggyőződtünk róla, hogy valóban tudunk írni az állományba, megkezdjük a ***HTML*** állományt.

2. Beolvassuk a ***feeds.txt*** tartalmát, amelyben soronként egy tartalom ***URL*** található. Hogy elkerüljük a szóközök és üres karakterek használatából adódó problémákat, levágjuk az üres karaktereket és figyelmen kívül hagyunk minden sort amiben nincs legalább egy nyomtatható karakter.
3. Ezek után végiglépkedünk a `feeds_list` tartalom listán, és meghívjuk `feedparser.parse()` függvényt az adott ***URL***-re. Ha kapunk választ, kiírjuk a ***myfeeds.html*** kimeneti állományba, az ***URL***-el és a cikk címével együtt.
4. Végül lezárjuk a ***HTML***-t és az állományt.

A kódba belepillantva láthatjuk, hogy személyes használatra nagyon könnyen készíthetünk hírösszesítőt. Persze ez csak nagyon vázlatos alkalmazás. Ha használhatóbbá szeretnénk tenni, valószínűleg érdemes lenne a ***feeds.txt*** és ***myfeeds.html*** tartalmát relációs adatbázis-kezelőbe vinni, a tartalom ***URL*** címét pedig automatikusan vagy félautomatikusan meghatározni a lap ***URL*** címe alapján, továbbá érdemes lenne tartalom kategóriákat is alkalmazni, így a többszörös tartalmak egységes egészként lennének olvashatóak. Amennyiben a leírás ismerősen hangzik, a kedves olvasó biztosan ***Bloglines*** felhasználó, hiszen ez a web-alapú blog gyűjtögető éppen a fenti módon működik. Nyilvánvaló módon a ***Bloglines*** sokkal több tartalmat és sokkal több felhasználót kezel mint amennyi az egyszerű játékpéldánkban található. Ha el szeretnénk készíteni a ***Bloglines*** belső szervezeti verzióját, némi személyre szabott kóddal kiegészített ***Universal Feed Parser*** és egy adatbázis-kezelő (például a ***PostgreSQL***) alkalmazásával a kialakítás egyaránt könnyű és hasznos is.

Összefoglalás

A spanyolviasz feltalálásának esetét gyakran említik a számítógépipar problémájaként. ***Mark Pilgrim Universal Feed Parser*** programja lehet, hogy csak egy apró igényt elégít ki a programok világában, de ez az igény szinte bizonyosan növekedni fog, ahogy az cikkgyűjteményeket használó személyek és szervezetek száma egyre szaporodik. Amennyiben kedvünk támad cikkgyűjtemény tartalmakat olvasni vagy értelmezni, érdemes használnunk a ***feedparser***-t. Kipróbált és jól dokumentált alkalmazásról van szó, amely gyakran frissül és fejlődik, munkáját pedig gyorsan és jól végzi.

Linux Journal 2004. december, 128. szám

Reuven M. Lerner, veterán web/adatbázis tanácsadó és fejlesztő, aki jelenleg Northwestern Egyetem Learning Sciences programjának végzős hallgatója. Weblogja az altneuland.lerner.co.il címen olvasható, és a reuven@lerner.co.il címen érhető el.

A felhasználói viselkedés vizsgálata (4. rész)

A sorozat előző részében megvizsgáltuk, hogyan juthatunk a modellezéshez szükséges adatokhoz. Most egy egyszerű program segítségével bemutatom az alkalmazott algoritmusokat és azok különböző paraméterezési lehetőségeit.

Az eddigiekben körüljártuk az elméleti lehetőségeket: modellezési kérdések és modelltípusok, adatgyűjtési módszerek, felhasználók azonosítása és elkülönítése. Most nézzük meg, hogy hogyan lehet használható modelleket, adatokat készíteni. Ehhez egy ***PHP*** nyelven írt szkriptet használunk, illetve ezen keresztül vizsgáljuk meg a lehetőségeket és az eredményeket.

Követelmények

Néhány elérhető modellező szoftver vizsgálata után fogalmaztam meg a legfontosabb követelményt: minél többféle modell készítését támogassa az eszköz, az előállított adatsorokat lehessen később más szoftverrel is feldolgozni (grafikus megjelenítés, elemzés), azaz legyen felkészítve sokféle (elsősorban szöveges) formátumú kimenet előállítására.

Előszűrés

A megvalósítandó eszköz egyik legfontosabb része az előfeldolgozó. Az általános felhasználhatósághoz ennek az alrendszernek széles paraméterezhetőséggel kell rendelkeznie, hogy bármilyen portál rendszer eseménynaplóját fel tudja dolgozni.

Három elkülöníthető funkcióra van szükség:

- ***Kihagyás***. A modellezés során a portáltól függően rengeteg eseménynapló bejegyzés nem hordoz releváns (a modellezés szempontjából felhasználható) információt. Ezeket a bejegyzéseket ki kell venni, mert csak zavarnák és lassítanák a feldolgozás és modellalkotás menetét.
- ***Összevonás***. A webszerverek és a HTTP protokoll lehetőséget nyújt ún. átirányításokra. Ez azt jelenti, hogy ha a felhasználó adatokat küld a szervernek, az azt feldolgozó program a lefutása után gyakorlatilag észrevétlenül egy másik weboldalra küldi tovább a kliens böngészőjét. Másik lehetőség, hogy egy weboldal több keretből (frame) áll, ekkor az egyes keretekbe betöltődő weblapok mind külön-külön bejegyzéseket jelentenek az eseménynaplóban. Ilyen esetekben tehát szükség lehet több eseménynapló bejegyzés összevonására, és egy weboldalként való szerepeltetésére a modellezés során.
- ***Átalakítás***. Egy másik jellemző megfontolás, hogy a portál összes weboldala egy fájlon keresztül érhető el. Ekkor az eseménynaplóban is csak ez az egy fájl jelenik meg, de a HTTP kérésből kiolvashatóak a paraméterek. Ebben az esetben arra van tehát szükség, hogy az ilyen paraméterezett lekérésekből egyszerű, könnyen olvasható és a lényeget kiemelő „weboldalakat” (tartalmat jellemző, címet és nevet tükröző modell bejegyzéseket) csináljunk.

Az előszűrés feladata továbbá, hogy a feldolgozhatatlan eseménynapló bejegyzéseket is eltávolítsa. Ilyen bejegyzések például a hibák, a nem létező erőforrásokra mutató lekérések.

Felhasználói azonosítás

A megvizsgált szoftverek között van, amelyik képes az egyes eseménynapló bejegyzéseket elkülöníteni, és egy-egy látogatóhoz rendelni. Ezen szoftverek legnagyobb hátránya az volt, hogy sehol nem volt elérhető a használt módszer(ek) leírása. (Az általam használt módszerekről cikksorozatom előző részében volt szó). Ezen okból kifolyólag a legfontosabb követelmény az, hogy az eszköz képes legyen a lehető legtöbb módon ezt a hozzárendelést elvégezni, és az egyes módszerek paraméterei széles skálán legyenek állíthatók (ne legyenek „bedrótozva” a szoftverbe).
A felhasználók elkülönítésére használatos módszereket alapvetően meghatározza a rendelkezésre álló eseménynapló, ezért szükséges, hogy a szoftver ilyen képességét (felhasznált metódusokat) mindig az adott adatforráshoz lehessen igazítani.

A lehetséges módszerek:

- munkamenet azonosító alapján, amelyet a webszerver helyez el az eseménynaplóban;
- idő (két lekérés között eltelt idő figyelése) és kliens IP cím alapú hozzárendelés legegyszerűbb (legkevesebb adatot tartalmazó) eseménynaplók esetén;
- részletesebb eseménynaplók esetén a ***referer*** és ***user agent*** értékek is bevonásra kerülnek az összerendelés folyamatának pontosításába.

Az eseménynapló két bejegyzése akkor tartozik egy felhasználóhoz, ha a weboldal lekérések között eltelt idő értéke (idő paraméter) kisebb, és akkor különítendő el más-más látogatóhoz, ha nagyobb, mint a beállított idő paraméter értéke. Ezen idő paraméter értéke nagymértékben befolyásolja a létrejövő modelleket (látogatók száma, különböző útvonalak megoszlása, stb.), ezért kiemelten kell kezelni ennek beállítási lehetőségét.

Exportálás

A szoftverrel szemben nem követelmény, hogy szép formában (grafikusan, szemléletesen) állítsa elő az eredményt, de fontos cél, hogy az adatokat többféle szöveg alapú forrásba képes legyen menteni a további feldolgozhatóság miatt. A lehetséges formátumok például: ***text***, ***CSV (comma separated value)***, ***XML***, ***HTML***. Ez azért szükséges, hogy további modellező és megjelenítő eszközök felé biztosított legyen az átjárás, megkönnyítve ezzel az érthetőséget és az ellenőrzést.

Az analizáló szoftver

Az analizáló szoftver egymás utáni feldolgozási lépések folyamatából áll, ugyanakkor a további felhasználás és fejlesztés szempontjából érdemes egy osztályba összefogni az egyes lépéseket megvalósító kódokat. Ennek további előnye, hogy az OOP-szemlélet következtében az eseménynapló vizsgálatának eredménye egyszerűen beilleszthető egy másik szoftverbe is (például egy általános eseménynapló statisztikai programba).

A fentieknek megfelelően a szoftver két fájlból áll:

- A ***WebUserModeller.class.php*** tartalmazza azt az osztályt, amely egybefogja a modellezés lépéseit megvalósító függvényeket, a függvények által közösen használt változókat, elrejti a belső funkciókat (például: eseménynapló sor beolvasása) és megfelelő metódusokat (interfészeket) biztosít az igénybevételhez. A modellezést végző kódok és algoritmusok ebben az osztályban találhatóak meg.
- A ***wum.php*** használja az előzőekben ismertetett osztályt a vizsgálathoz. Egyes paramétereket a ***wum.php*** programkódjában lehet változtatni, ezeket a beállításokat ebben a fájlban lehet megadni (például: vágási küszöbértékek). Ez a fájl biztosítja jelenleg a modellező osztály használatának lehetőségét.

Az elemzőprogram a ***wum.php*** parancssori meghívásával indítható el a megfelelő paraméterek megadásával. Feladata a megadott paraméterek szerint (használt konfigurációs fájl, elkészítendő modell típusok, kimeneti könyvtár) a modellező osztály megfelelő metódusaink meghívása, és a létrejött eredmények feldolgozása, ami jelen esetben a fájlba mentést jelenti.

Konfigurációs lehetőségek

Minden eseménynaplóhoz készíteni kell egy konfigurációs fájlt, amely az adott naplófájlra jellemzően és a felhasználás módjától függően tartalmazza a modellezési paramétereket. Ezek a paraméterek alapvetően befolyásolják az elkészülő modelleket.

A következő paramétereket kell beállítani a konfigurációs fájlban:

- ***log_file***: az analizálandó eseménynapló elérési útvonala, kötelezően megadandó.
- ***base_href***: a portál alap linkje (URI).
- ***use_cookie***: a felhasználók azonosításához lehetőség van munkamenet azonosító használatára. Ellenkező esetben idő alapú elkülönítéssel választja el a szoftver a látogatókat. Alapértelmezésben nem munkamenet alapú azonosítás történik.
- ***root_page***: a weben általában egy portál nyitóoldalának eléréséhez elegendő egy tartomány címet (például: ***http://www.linuxvilag.hu***) ismerni, nem kell a pontos kezdő fájl teljes nevét is manuális megadni (például: ***http://www.linuxvilag.hu/index.html***), ugyanakkor a háttérben mindig van egy fájl. A két elérési lehetőség egybeolvasztásához lehet itt megadni a kezdő fájl nevét, és a szoftver automatikusan lecseréli a nyitóoldali lekéréseket az itt megadottra, így biztosítva a modellek egységességét. Elhagyása esetén nem kerül figyelembevételre.
- ***allow_without_extension***: hasonló a ***root_page***-hez, beállításával a modellezés során a könyvtár lekéréseket lehet kihagyni illetve bevonni az analizálásba. Alapértelmezésben megengedő állapotú.
- ***remove_get_params_from_request***: a kérésekben szereplő ***GET*** paramétereket lehet eltávolítani, így például a felhasználó adatküldéseit lehet kiszűrni, vagy a ***GET*** paraméterbe kódolt különböző oldalakat lehet egynek venni, ezáltal akár radikálisan csökkentve a modellek felbontását. Alaphelyzetben nem engedélyezett.
- ***time_window***: a legfontosabb paraméter, ezt az időt veszi alapul a szoftver a felhasználók elkülönítésénél. Ha a ***use_cookie*** be van kapcsolva, akkor ezen érték nem kerül felhasználásra. Megadása kötelező.
- ***request_file_types***: az analízis folyamat az itt megadott típusú fájlokat (lekéréseket) veszi csak figyelembe a modellalkotás során. Ha nem szükséges a képek lekéréseinek modellezése, akkor itt nem kell megadni az adott portálra jellemző képtípusokat. Azokat a fájltípusra jellemző karakterláncokat kell megadni felsorolva, amelyek szerepelhetnek az engedélyezett kérésekben. A szoftver egyszerű mintakeresést végez. Mivel ez a modellezés egyik alapbeállítása, ezért mindenképpen meg kell adni a feldolgozandó kérések jellegét.
- ***request_simplify***: a modellezést nem befolyásolja közvetlenül, azonban megkönnyíti a modell elemzését azáltal, hogy a megadott paramétereknek megfelelően az egyes lekéréseket egyszerűbb szövegre cseréli le (például: ***index.php?modul=bolt&kategoria=muszaki_cikk*** helyett csak ennyi lesz a kimeneti eredményekben: ***bolt-muszaki_cikk***). A cserélendő kérésre jellemző (vagy azzal megegyező) mintához kell megadni az új (cserélt) karaktersorozatot. Opcionális paraméter.
- ***groupize_path***: az előszűrésnél említett összevonásokat lehet itt megadni Az új, összevont kéréshez fel kell sorolni az összevonandó kérésekre jellemző mintákat. Opcionális.
- ***page_referers***: az analízis során a felhasználók elkülönítéséhez felhasználható a ***combined*** szerkezetű eseménynaplókban szereplő ***referer*** érték is. Itt lehet megadni,

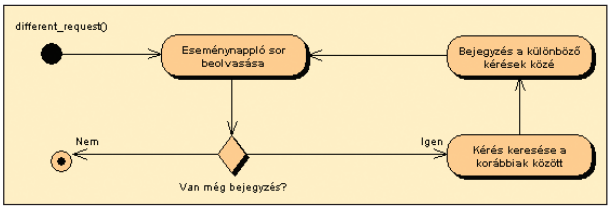
hogy az egyes weboldalakhoz mely másik weboldalak tartozhatnak ***refererként***. A portál weboldalaihoz felsorolásszerűen kell megadni a lehetséges ***referer*** értékek mintáit. Ha egy adott weboldalhoz nincs megadva lehetséges ***referer*** érték, az azt jelenti, hogy bármilyen ***referer*** értéket elfogadunk. Opcionális, csak a megadott weboldalaknál használja, vagyis minden egyes weboldalra külön-külön lehet megadni.

Használat, indítási paraméterek

Az analizáló szoftver képességei a ***wum.php*** indításával használhatók ki. Kötelezően megadandó paraméter a `--ini`, amely értéke határozza meg a felhasználandó konfigurációs fájl útvonalát.

A következő paraméterek opcionálisak:

- `--help`: minimális leírást ad a megadandó paraméterekről.
- `--output`: a kimeneti fájlok mentési könyvtára, ha nincs megadva, akkor a wum.php futtatási könyvtárába kerülnek az eredményeket tartalmazó fájlok.
- `--version`: a különböző beállításokkal készült modellezések elkülönítéséhez lehetőség van egy verziószám megadására, ez a verzió azonosító az eredmény fájlok nevét befolyásolja (hozzáfűzésre kerül), elhagyása esetén egy időbélyeggel helyettesítődik. Létező fájlnév esetén automatikus felülírás történik.
- `--diff`: az a küszöbérték, amely alatti előfordulásokat az egyes eredményekből el kell távolítani. A viszonyítás alapja a felhasználói útvonalak száma (vagyis az összes látogató száma). Megadása opcionális, alapértéke a 0, ekkor nem kerül figyelembe vételre.
- `--model`: a különböző lehetséges modellek elkészítését lehet bekapcsolni segítségével. Elhagyása esetén minden lehetséges modellezési folyamat lefuttatásra kerül.
- A `--model` paraméter lehetséges értékei és a hozzá tartozó modellek:
- ***up (user_path)***: A modellezés alapja, a felhasználók elkülönítését és az útvonalak összegyűjtését végzi. A további modelltípusok elkészítéséhez ennek a modellnek már rendelkezésre kell állnia.
- ***ro (routes)***: a felhasználók útvonalait állítja össze, megszámolja, hogy ugyanazokat a weboldalakat hány különböző látogató járta végig. Itt a weboldalak meglátogatási sorrendje is számít.
- ***sp (same_pages)***: abban különbözik a routes adatoktól, hogy itt már nem számít az oldalak sorrendje, tehát csak azt vizsgáljuk, hogy a felhasználók közül egy látogatás során hányan jártak azonos oldalakon (a teljes látogatást tekintve).
- ***ct (click_times)***: az egyes weboldalak közötti átkattintási időminimumokat, időmaximumokat és időátlagokat mutatja. További felhasználása: az itt kapott értékek alapján munkamenet azonosító nélküli, idő alapú eseménynapló elemzés.
- ***wmi (web_model_in)***: azt mutatja meg, hogy az egyes oldalakra honnan és hányszor érkeztek a felhasználók. A bejövő (tulajdonképpen az előző) oldalakra nincs szűrés, így nincsenek elkülönítve a kívülről illetve a portálon belülről érkezettek.



1. ábra Munkamenet azonosító alapú analízis

- ***wmo (web_model_out)***: hasonló a ***wmi*** modellhez, a különbség az irányban van. Ebben az esetben azt vizsgáljuk, hogy egy adott weboldalról melyik másik weboldalra léptek tovább a felhasználók.
- ***dr (different_requests)***: az eseménynaplóban megtalálható összes különböző kérést gyűjti ki. Nem igazi modell, segédeszköz a modellezés paramétereinek pontosításához. Önmagában is használható.

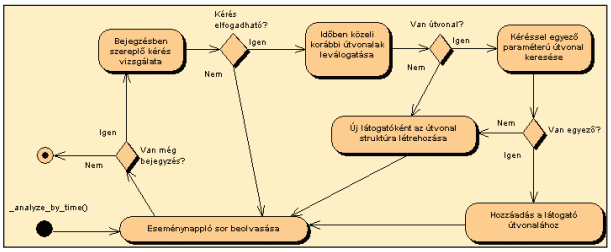
Működés, algoritmusok

Az analizáló szoftver lineáris lefutású: eseménynapló sorának beolvasása, feldolgozás, az adatok szerint egy felhasználó keresése vagy új létrehozása, majd a különböző modellek összeállítása a felhasználók adatsorai alapján.
Az analizáló szoftver legfontosabb része a felhasználók elkülönítéséért felelős rész, mert minden további modellezést végző részfeladat ennek az eredményein operál.

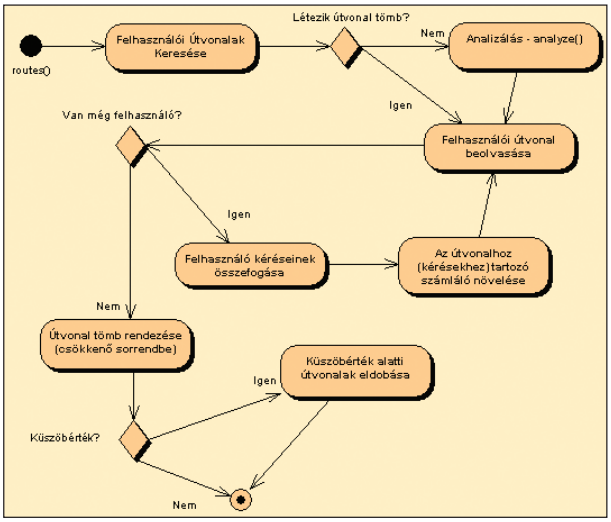
Felhasználók válogatása

- ***Felhasználók válogatása munkamenet azonosítók alapján***: A munkamenet azonosító alapú eseménynapló analízis folyamata egyszerű, mert minden egyes bejegyzésben már szerepel a látogatókat egyértelműen elkülönítő (azonosító) adat. Ez a módszer egyszerűen az azonosító alapján tömbbe gyűjti a látogatókat (*1. ábra*), és a hozzájuk tartozó weboldal lekérések paramétereit. A szoftver ebben az esetben nem veszi figyelembe az IP címet, időbélyeget, ***referer***t, stb., hiszen ezeket az adatokat a webszerver a weboldal lekérésekor „belekódolta" a munkamenet azonosítóba.
- ***Felhasználók válogatása idő alapú elkülönítéssel***: A látogatók elkülönítése és útvonalaik összegyűjtése elsődlegesen az IP cím, távoli eseménynapló, távoli felhasználó adatok alapján történik. Ez önmagában nem elegendő az egyértelmű megkülönböztetéshez. További segítséget nyújt a ***referer*** és a böngésző adatok jelenléte. Azonban a legegyszerűbb szerkezetű eseménynaplók nem tartalmaznak az IP címen és a kérés időpontján kívül több olyan adatot, amely segítené a látogatók elkülönítését, vagy egy nagyvállalati hálózatból érkező látogatók esetén jellemzően a böngésző adatok is azonosak.

Az idő alapú elkülönítés során (*2. ábra*) az éppen vizsgált kérést a megadott időintervallumon belül eső, kéréssel rendelkező látogatóhoz próbáljuk meg csatolni, oly módon, hogy a fent említett adatok közül a lehető legtöbb egyezzen. Teljes egyezés esetén a vizsgált kérést az adott látogatóhoz kapcsoljuk, ha nem találtunk teljes egyezést, akkor mint új látogató kezeljük, és egy új útvonal tömb bejegyzést nyitunk neki.



2. ábra Idő alapú analízis



3. ábra Különböző útvonalak modellezése

Felhasználói útvonalak

Az analízis eredményeképpen előálló felhasználónkénti lekéréseket (útvonalakat) mutatja meg. Az eredmény egy többdimenziós tömb, amely felhasználónként tartalmazza az egyes (sorrendben) lekért weboldalakat és néhány, az egyes kérésekhez tartozó kiegészítő adatot. Ez egy interfész, a tömb valójában az analizáló alrendszer eredménye.

Például:

```
[15] => Array
  (
    [0] => Array
      (
        [ip_address] => 213.157.96.206
        [remote_log] => -
        [user_id] => -
        [time] => 1050899398
        [request] => /index.php
        [referer] => http://www.jegyzet.com/
        [user-agent] => Mozilla/4.0 (compatible;
        ➥MSIE 6.0; Win98)
      )

[1] => Array
      (
        [ip_address] => 213.157.96.206
        [remote_log] => -
        [user_id] => -
        [time] => 1050899431
        [request] => /kereses.php
        [referer] => http://www.jegyzet.com/
        ➥index.php
        [user-agent] => Mozilla/4.0 (compatible;
        ➥MSIE 6.0; Win98)
      )
  )
```

A példa a „15-ös” látogató útvonalát mutatja: először lekérte az ***index.php*** weboldalt, majd a ***kereses.php*** weboldalt. Több kérése nem volt ennek a látogatónak. A példából kiolvasható, hogy melyik honlapot milyen időpillanatban kérte le, mi volt a kéréshez tartozó ***referer***, böngésző, kliens IP cím, stb. A következő folyamatok az ilyen tömbökből álló felhasználói útvonalak tömb alapján készítik el a specializált modellt.

Különböző útvonalak

A felhasználói útvonalakban (*3. ábra*) keresi meg az azonosakat (a bejárt weboldalak és sorrendjük azonos), ezen útvonalakat összegzi. Lehetőség van arra, hogy egy paraméterként megadott százalék alatti előfordulásokat eldobja, a viszonyítási alap a látogatók száma. Paraméterként megadható, hogy a megadott előfordulási százalék alatti útvonalak ne szerepeljenek az eredményben.
A módszer a következő: a felhasználói útvonalak tömbjét végigolvasva minden felhasználó útvonala leképzésre kerül egy egyedi karakterláncba. A megegyező útvonalakat (sorrend és többes látogatások is számítanak) bejárt felhasználóknál ez az egyedi karaktersorozat azonos. A továbbiakban a létrejött egyedi útvonalleírók kerülnek összegzésre (megszámoljuk, hogy egy egyedi útvonalat leíró karaktersorozatból hány van).

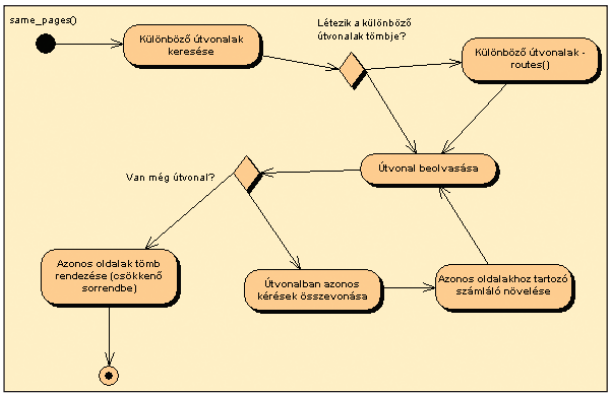
Például:

```
[/index.php] => 3128
[/index.php /eredmeny.php] => 1016
[/index.php /eredmeny.php /eredmeny.php] => 386
[/index.php /eredmeny.php /kereses.php] => 271
```

A fenti példában látható, hogy csak az ***index.php*** fájlt összesen 3128 látogató nyitotta meg, ugyanakkor az ***eredmeny.php*** weboldalt már csak 1016-an nézték meg. A weboldal lekérések sorozata különálló, vagyis a példa első sora nem tartalmazza a második sorból az ***index.php*** weboldalra irányuló lekéréseket.

Azonos oldalak egy látogatáson belül

Hasonló a különböző útvonalak módszerhez, azzal a különbséggel, hogy az egyes weboldalak meglátogatási sorrendje nem számít. Itt is lehetőség van a nagyon ritka előfordulások elhagyására.
A módszer (*4. ábra*) majdnem azonos a különböző útvonalak elkészítésének menetével. A különbség annyi, hogy itt a felhasználói útvonalakból az azonos weboldalra irányuló többes lekérések összevonásra kerülnek (egyszer szerepelnek), illetve nem számít az egyes lekérések sorrendje sem (ABC sorrendbe van rendezve a lekéréseket leíró karakterfűzér).



4. ábra Azonos oldalak egy látogatáson belül

Például:

```
[/index.php] => 3350
[/eredmeny.php /index.php] => 1947
[/eredmeny.php /index.php /kereses.php] => 1392
[/index.php /kereses.php] => 195
```

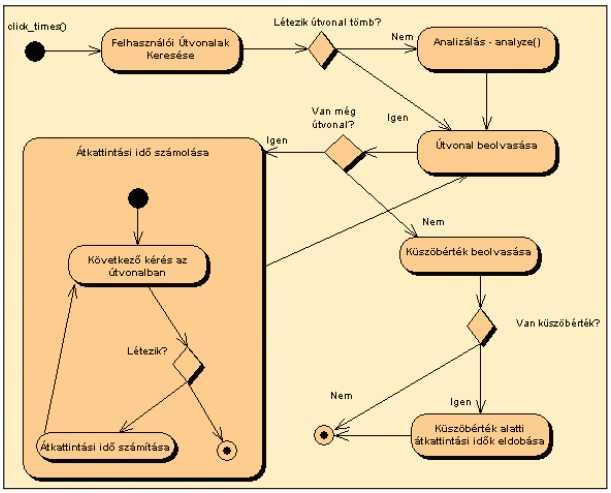
Átkattintási idők

Az egyes weboldalak közötti átkattintások számát, minimális, maximális, összes és átlagos idejét adja meg egy többdimenziós tömbben. Az átkattintásoknak irányuk van (forrás és cél), az eredményben külön van választva a két irány. Elsősorban az idő alapú látogató-elkülönítésnél fontos, az átlagos és maximális idők alapján lehet a megfelelő idő paramétert módosítani. Paraméterként megadható, hogy azok az átkattintások ne szerepeljenek az eredményben, amelyek száma nem éri el a felhasználók számának megadott százalékát.
A különböző átkattintási idők számítása (*5. ábra*) a felhasználói útvonalak vizsgálatával történik, az útvonal két lekérése közötti idők kerülnek vizsgálatra és összegzésre, átlagszámításra.

Például:

```
Array
(
  [/index.php] => Array
    (
      [/eredmeny.php] => Array
        (
           [clicks] => 3463
           [duration] => 121315
           [avg_dur] => 35.03 sec
         )
    )
)
```

A példa azt mutatja, hogy az ***index.php*** weboldalról összesen 3463 átkattintás történt az ***eredmeny.php*** weboldalra, a két weboldal lekérése között összesen 121315 másodperc telt el, amely adatokból már egyszerűen kiszámolható, hogy az ***index.php*** lekérése után átlagosan 35.03 másodperccel kérték le a felhasználók az ***eredmeny.php*** weboldalt.



5. ábra Átkattintási idők számítása

A modellezés inkább az általános érvényű megállapításokat keresi, ezért itt is lehetőség van a ritka átkattintások elhagyására, ezáltal egyszerűsítve a modell érthetőségét és áttekinthetőségét.

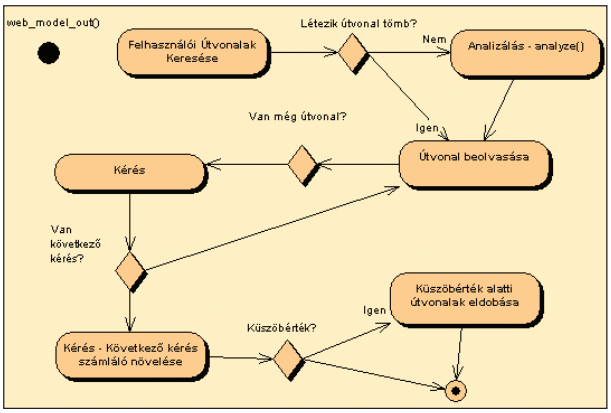
Be- és kilépő oldalak

A belépő weboldalak egy portálon belüli weboldalhoz adják meg, hogy honnan és hányszor nyitották meg (ezek között szerepel portálon belüli és kívüli forrás is). Ez az adatsor az analízis során létrejön, mert egyszerűen előállítható az eseménynapló bejegyzésből a ***referer*** adatok használatával. A belépő oldalak statisztikája csak akkor állítható elő, ha a ***referer*** adatok léteznek az eseménynaplóban, ellentétben a kimenő oldalak statisztikájával, ami a felhasználók lekérései alapján kerül összeállításra (útvonalban következő lekérés alapján), vagyis a legegyszerűbb szerkezetű eseménynapló alapján is előállítható.

Például:

```
[/index.php] => Array
  (
    [-] => 4979
    [http://www.oktaton.hu/index.php] => 1355
    [http://www.puska.hu/portal1.htm] => 373
    [http://www.lapkereso.hu/keres.php] => 350
  )
```

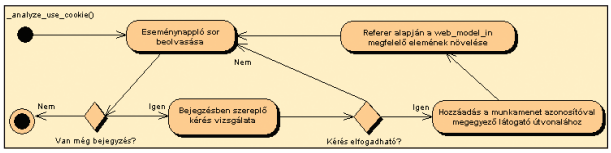
A fenti példában látható, hogy az ***index.php*** weboldalra 4979 felhasználó érkezett ***referer*** adat nélkül (ezt jelenti a „-”). Ez jellemzően abban az esetben fordul elő, amikor a felhasználó egy új böngésző ablakot nyit és annak címsorába maga gépeli be az adott portál elérhetőségét.
A kilépő weboldalak ellenkezőleg működnek, mert itt azt vizsgáljuk, hogy egy adott weboldalról merre mentek tovább a látogatók, ahol minden weboldal szigorúan az adott portálhoz tartozik. Ez a modell a felhasználói útvonalak elemzésével gyűjti össze az adatokat, megvizsgálja, hogy egy lekérés után melyik másik weboldal lett lekérve és összeszámolja az egyes párok előfordulásait (*6. ábra*).



6. ábra Kilépő oldalak statisztikák készítése

Például:

```
Array
(
  [/index.php] => Array
    (
        [/eredmeny.php] => 3774
        [-] => 3751
        [/kereses.php] => 640
        [/index.php] => 610
        [/reszlet.php] => 97
        [/figyeles.php] => 62
```



7. ábra Különböző lekérések kigyűjtése

```
        [/feladas.php] => 54
        [/tajekoztato/?t=ismerteto] => 15
        [/tajekoztato/] => 47
        [/tajekoztato/?t=szabalyzat] => 25
    )
)
```

Fenti példában a „-” jelentése: nem volt a felhasználónak további weboldal lekérése az ***index.php*** után, azaz távozott az adott portálról és az utolsó meglátogatott weboldala az ***index.php*** volt.
Mindkét esetben megadható az a százalékos paraméter, ami alatti előfordulásokat ki kell hagyni az eredmény tömbből.

Különböző kérések

Eltérően az eddig ismertetett analizáló és modellező alrendszerektől, ez a metódus inkább az előzetes eseménynapló vizsgálatot segíti azáltal, hogy a megtalálható összes különböző kérést kigyűjti (*7. ábra*). Így megvizsgálható, hogy a tényleges modellezéshez mely lekérések (weboldalak)

használandóak. Más szemszögből megtudható, hogy egyáltalán mely weboldalakat kérték le a látogatók a szerverről. A függvény használja az eseménynapló bejegyzéseket szűrő és egyszerűsítő folyamatokat, amelyek a konfigurációs fájlban akár ki is kapcsolhatóak.
Paraméterként megadható, hogy a kérések egyszerűsítés után kerüljenek összehasonlításra, vagy minden kérést úgy vegyen a szoftver, ahogy a naplófájlban szerepel.

Továbbfejlesztési lehetőségek

A megvalósított analizáló szoftver elsősorban az algoritmusok és lehetséges modellek bemutatására készült, az alapfunkciókat biztosan képes nyújtani. A fejlesztés során felmerült továbbfejlesztési lehetőségek ismertetése következik.

Gyorsítás, nagyméretű eseménynaplók feldolgozása

A első fontos fejlesztési lépés a szoftver működésének gyorsítása, amely összefügg a nagyméretű (200 MB feletti) eseménynaplók feldolgozásával. A szoftver jelenleg a teljes adatstruktúrát (felhasználói útvonalak) memóriában tárolja, ami legalább annyi memóriát igényel, mint a feldolgozandó eseménynapló mérete.
Tekintve, hogy már egy közepes forgalmú portálon is akár napi néhány száz megabyte eseménynapló keletkezik, megoldandó, hogy ne kelljen az egész struktúrát a memóriában tárolni. Elég volna csak azokat a felhasználói útvonalakat megtartani a memóriában, amelyek az éppen elemzett eseménynapló bejegyzés feldolgozásakor még számításba jöhetnek, mint olyan felhasználói útvonal, amelyhez az adott kérés hozzácsatolható (beilleszthető a látogató útvonalába). A figyelembevétel eldöntését megkönnyíti, hogy mind a munkamenet azonosító alkalmazása, mind az idő alapú látogató elkülönítés esetén behatárolt, hogy milyen időintervallumban beérkezett eseményeket (naplóbejegyzéseket) kell még memóriában tárolni. Mivel a webszerver a munkamenet azonosítók érvényességét ugyanúgy idő alapján figyeli, mint ahogy a látogatókat a lekérések időkülönbsége alapján elkülönítjük, ezért adható meg a szerver beállításának ismeretében a fenti intervallum.
Mivel a további feldolgozások a felhasználói útvonalakat használják, ezért célszerű például egy adatbázisba tölteni az analizálás során keletkező útvonalakat. Így a további modellkészítési lépések is gyorsíthatóak az adatbázis kezelő rendszerek optimalizálásával.

XML alapú konfiguráció

A fejlesztés során egyértelművé vált, hogy az eddig elterjedt konfigurációs fájl struktúra nem alkalmas akármilyen beállítás tárolására. Főleg a szöveges és speciális karakterek, tömbök kezelésre nem alkalmas.
Mivel az egyes eseménynapló fájlok szerkezetileg egyediek is lehetnek, ezért lehetőséget kell biztosítani a reguláris kifejezések eseménynaplóhoz kötéséhez (vagyis a konfigurációs fájlban kell tudni megadni).
A *XML* alapú konfiguráció további előnye a csoportosíthatóság több mint két szint mélységben is. Így kényelmesebbé tehető a tömb jellegű adatstruktúrák megadása, pontosabban leírható már a címkékkel is az egyes elemek jelentése és célja.

Az *XML* fájlok feldolgozása még nem teljesen kiforrott a *PHP* esetében, de már vannak használható és gyorsan alkalmazható megoldások, kiegészítések is, amelyek tudása olykor még korlátozott, vagy egyedi fejlesztést igényel.

Kereső robot a referer alapú vizsgálathoz

A *referer* alapú vizsgálathoz szükséges, hogy az adott portálra vonatkozóan az egyes weboldalak között az összes átjárási lehetőséget megadjuk. Nagyobb, rendszeresen bővülő portálokon (például áruház, hírmagazin) ennek még az időszakosan történő megadása is igen komoly munkát igényel.
A *referer* alapú vizsgálathoz kifejleszthető egy (általánosan is használható) úgynevezett ***crawler (kereső robot)***, amely automatikusan végigjárja a portál összes oldalát, kigyűjtve az összes linket, rendszerezve az átjárási lehetőségeket. Jelenleg elterjedt portál navigációs és hirdetési jellegzetesség, hogy minden weboldal (funkció) elérhető minden weboldalról, vagy a linkek dinamikusan (adott tartalomhoz, hírhez kapcsolódóan, napszaktól függően, vagy véletlenszerűen) látogatásonként változóan kerülnek ki egy-egy weboldalra. Fentiek miatt a kereső robot igazán csak olyan esetekben tud jól segítségünkre lenni az átjárási lehetőségek begyűjtésében, ahol a portál szerkezete viszonylag fix, jól elválasztott navigációs struktúrák vannak.

XML és grafikus kimenetek

Az elkészült analizáló szoftver jelen formájában egy prototípusnak megfelelő kimenetet produkál, amelynek megértése erősen feltételezi a szoftver és a modellek ismeretét.
A modellek további feldolgozására (például egyedi megjelenítés, beillesztés más programba) ma az a legkézenfekvőbb és legjobb kompatibilitási megoldás, ha az eredményeket előre definiált szerkezetű *XML* fájlba is ki lehet menteni.
A megértést és szemléltetést könnyítheti meg, ha az eredmények grafikus formában jelennek meg (például oszlopdiagramm, gráf, folyam). Ez segítheti a nem szakértőket a szoftver mélyebb ismerete nélkül is a megfelelő döntés (szerkezet, navigáció, tartalom, stb. módosítása) meghozatalában, az analizálás eredményeinek és jelentőségének felismerésében.

Összefoglalás

Jelen cikkben egy konkrét modellező szoftver működését vizsgáltuk meg, külön kitérve az egyes algoritmikus kérdésekre.
Következő cikkemben a most bemutatottak alapján néhány konkrét példán keresztül szemléltetem, hogy mik a felhasználói viselkedés vizsgálatának gyakorlati eredményei.

Beszédes Balázs (beszedes@ei.hu)
24 éves, az e-Média Informatikánál mérnök-informatikus. Hobbija a kerékpározás és a kirándulás.

Samba – Windowsban is otthon (1. rész)

Néhány hónapja az eWeek egyik szerzője arra a következtetésre jutott, hogy a Sambát futtató linuxos kiszolgálókat immár nem egyszerűen a Windows rendszerekkel való együttműködésre célszerű használni, hanem magának a Windows 2003 Servernek a kiváltására...

Napjainkban szerte a médiában azt lehet olvasni, azt hallani, hogy a nyílt forráskódú rendszerek, ezen belül a ***Linux*** alapú rendszerek mindenki – vagy legalábbis majdnem mindenki – nagy örömére egyre nagyobb népszerűségnek örvendenek. A böngészők terén a ***Firefoxnak*** sikerült elérni azt, amit nagyon régóta senkinek, 90% alá szorította az ***Internet Explorer*** használtságát. Ez ugyan kicsit fanyar humornak is beillik, de a lényeg, hogy talán megmozdult valami, talán végre verseny lesz, amiből pedig egy fél kerülhet ki nyertesen, a felhasználó.

A kiszolgálók területén talán a fentinél jobb a helyzet, a verseny sokkal kiegyenlítettebb, a ***Linux*** népszerű, sőt egyre népszerűbb. Viszont hiába használunk a kiszolgálónkon ***Linuxot***, ha minden kliensünk ***Windowst*** használ és az általunk nyújtott szolgáltatásokat adott esetben vagy nem tudja igénybe venni, vagy egyszerűen csak hasznát nem tudja venni. De ne keseredjünk el, ne essünk pánikba, mert mint mindenre, erre is van megoldás. Úgy hívják, hogy ***Samba***.

A ***Samba*** egy windowsos kliensek kiszolgálására alkalmas rendszer, amely amellett, hogy állomány- és nyomtatómegosztást kínál, ennél sokkal többet tud. Alkalmas ***Windows NT*** rendszerű tartományvezérlésre, ***Windows*** profilok létrehozására és tárolására, házirendek készítésére és alkalmazására és szinten mindenre, amit egy windowsos munkacsoportos kiszolgálótól elvárunk.

Cikksorozatomban ezeket a funkciókat vesszük szemügyre, nézzük meg, hogy mely szolgáltatás milyen módon érhető el.

A telepítés

Kezdjük az elején, tegyük fel a csomagokat. Illetve, mielőtt feltennénk a csomagokat nézzük meg, hogy milyen csomagok vannak és ezekből miket szeretnénk telepíteni.

A ***Samba*** kiszolgálóból jelenleg két fő verzió érhető el, a 2.2.x és a 3.0.x verziók. Ebben a cikkben a 3.0-s verziót és annak szolgáltatásait tárgyaljuk, úgyhogy javaslom ezen verzió telepítését. Ugyan a főbb pontokban a 2.2 és 3.0 telepítése és beállítása hasonló, ám a 3.0 több olyan új szolgáltatást is tartalmaz, ami a 2.2-ben még nincs jelen.

Samba kiszolgáló a legtöbb ***Linux*** és ***Unix*** terjesztésre elérhető bináris formában, tehát a dolgunk nem több, mint a megfelelő csomagot letölteni és telepíteni. Általában a ***Samba*** majd' minden terjesztésben elérhető az adott disztribúció csomagkezelőjén keresztül, de ha nem találnánk, akkor a ***www.samba.org*** weboldalról letölthető több tucat különböző kiadás.

Én egy ***Debian Sarge*** kiszolgálóra telepítettem a csomagokat, teljesen egyszerűen az ***apt*** forrásból elérhető verziókat. Javaslom továbbá telepíteni a ***swat*** csomagot, amely egy webes alapú konfigurációs eszköz a ***Samba*** beállításához.

Az első lépések

Ha megtörtént a telepítés, akkor nézzük, hogy mit és hol találunk. A legfontosabb talán az ***/etc/samba*** könyvtár, ugyanis a ***Samba*** az összes beállítását ebben a könyvtárban tárolja – bár ez terjesztésenként esetleg változó lehet, de egy kis logikával könnyen meg lehet találni a megfelelő könyvtárat. A másik fontos könyvtár, illetve állomány az ***/etc/init.d/samba***. Ezzel a szkripttel lehet a kiszolgáló démont indítani és leállítani.

Miután telepítettük a kiszolgálót, a ***samba*** könyvtárban találunk egy ***smb.conf*** állományt. Ez az állomány tárolja a kiszolgáló beállításait, amennyiben változtatni kívánunk a rendszeren, azt itt kell megtenni. A másik fontos állomány a ***samba*** könyvtárban az ***smbpasswd***, ugyanis ebben a fájlban kerülnek tárolásra a ***Samba*** által beregisztrált felhasználók és a hozzájuk tartozó jelszavak. Ezt az állományt az ***smbpasswd*** parancs használatával tudjuk kezelni. Így lehet új felhasználót hozzáadni, jelszót cserélni és hasonló adminisztrációs műveleteket elvégezni.

Az smb.conf

Nézzük akkor, hogy miből áll az ***smb.conf*** állomány, milyen részekre oszlik, azoknál milyen beállítási lehetőségeink vannak. Első nekifutásra egy nagyon egyszerű beállítást nézünk meg, majd ezt bővítjük a későbbiekben.

Az első bekezdés az ***smb.conf*** állományban a `[global]`, tehát a globális beállítási opciók listája. Itt olyan beállításokat tehetünk, amelyek a teljes kiszolgáló entitásra vonatkoznak, bár ezek egyes paramétereit a későbbiekben az egyes kiosz-

tások beállításánál felül írhatjuk. Nézzünk egy egyszerű beállítást és nézzük meg lépésről-lépésre, hogy mi mit is jelent. Ha ezen túl vagyunk és működnek a beállításaink, akkor elkezdjük a rendszert bővíteni és felruházzuk minden jóval, amit a *Samba* kínál.

A `workgroup` változó értékének azt a csoport, vagy tartomány nevet kell megadni, amiben szeretnénk a kiszolgálót üzemeltetni. A `server string` változó értéke egy szabadon választott karaktersorozat, amelyet a kiszolgálónk nevének adhatunk meg. Ez most jelenleg a `$hosztnév server (Samba $verzió_szám)` lesz.

Amennyiben szeretnénk a kiszolgálót *WINS* szerverként is üzemeltetni, amihez *Win9x*-es gépek kiszolgálásához szükségünk lehet, akkor adjuk a `wins support` változónak a `yes` értéket.

A `dns proxy` változóval állíthatjuk, hogy az *nmbd* démon a gépek *NetBIOS* nevének visszakeresését a *DNS*-ben is megpróbálja vagy sem. Ennek az alapértelmezett beállítása `no`, úgy gondolom, hogy kezdetnek ez tökéletesen megfelel, hacsak nincs olyan *DNS* kiszolgálónk, amely ezt a műveletet támogatja.

A `name resolve order` azt mutatja, hogy a gépek nevének visszafejtéséhez melyik adatbázisokat milyen sorrendben használjuk.

A `log file` változónak megadhatjuk, hogy a *Samba* hová helyezze el a saját naplóállományát. Amennyiben nem tetszik az alapértelmezett beállítás, nyugodtan cseréljük ki.

A `max log size` változó értékének megadhatjuk kilobájtban kifejezve a naplóállomány maximális méretét. A `syslog = 0` azt szimbolizálja, hogy a *Samba* ne a */var/log/syslog* állományban, tehát nem a rendszernaplóban helyezi el saját bejegyzéseit. Amennyiben mégis erre volna szükségünk, akkor változtassuk 1-re az értéket.

A `security` változó már egy érdekesebb opció, ennek a beállításaival már érdemes esetleg játszadozni. Nálunk a beállítás a `user` szint, ami annyit tesz, hogy a felhasználók azonosítását az *smbpasswd*, vagy az aktuálisan beállított felhasználói adatbázis alapján a kiszolgáló végzi. Amennyiben a `share` értéket adjuk meg, úgy egy egyszerű, jelszó hitelesítés nélküli hozzáférést teszünk lehetővé a rendszerben. Amolyan *Windows9x* stílusban – megosztunk mindent, aztán akinek van megfelelő nevű felhasználója látja a teljes tartalmat. Amennyiben a `domain` beállítást választjuk, úgy a kiszolgáló a felhasználót az aktuális tartományvezérlő felhasználói adatbázisából fogja hitelesíteni. Ez akkor jó megoldás, ha a szervert egy *Windows* tartományhoz csatolt kiszolgálóként kívánjuk üzemeltetni. A `security = server` beállítással elérhetjük, hogy a kiszolgáló entitás egy másik megfelelően beállított *SMB* kiszolgáló(egy *Samba* vagy NT kiszolgáló) adatbázisa alapján azonosítja és hitelesíti a felhasználót. Amennyiben nem található használható kiszolgáló az azonosításra, úgy a *Samba* megpróbál a `security = user` metódus alapján beléptetést végezni. Végezetül érdemes megemlíteni az `ADS` beállítási lehetőséget, amellyel Windows *Active Directory Server* támogatásával tudunk felhasználói hitelesítést végezni. Fontos azonban megjegyezni, hogy ezt az opciót használva a kiszolgálót nem fogjuk tudni *Active Directory* kiszolgálóként üzemeltetni sőt, a Samba egyáltalán nem képes, még *LDAP* támogatással sem *AD* kiszolgálóként működni!

Példa a beállításra

```
[global]
workgroup = HOME
server string = %h server (Samba %v)
wins support = yes
dns proxy = no
name resolve order = lmhosts host wins bcast
log file = /var/log/samba/log.%m
max log size = 1000
syslog = 0
panic action = /usr/share/samba/panic-action %d
security = user
encrypt passwords = true
passdb backend = smbpasswd
obey pam restrictions = yes
guest account = nobody
invalid users = root
unix password sync = yes
passwd program = /usr/bin/passwd %u
passwd chat = *Enter\snew\sUNIX\spassword:*
➥%n\n *Retype\snew\sUNIX\spassword:* %n\n .
load printers = yes
printing = cups
printcap name = cups
preserve case = yes
short preserve case = yes
socket options = TCP_NODELAY
domain master = auto
```

Az `encrypt passwords` változót állítsuk `true` értékre, ugyanis ez a változó azt a beállítást hivatott elvégezni, miszerint a jelszavak a klienssel való egyeztetés során kódolt formában kerüljenek közlésre. Ez egyfelől mindenképpen ajánlott az esetleges biztonsági kockázatok elkerülése végett, másfelől a *Windows* ügyfelek *NT 4.0 SP3* és a *Windows 98* óta ezt a beállítást használják alapértelmezettként. (Ezt a *Windows registry*-ben ugyan ki lehet kapcsolni, de ez egy nem támogatott konfiguráció.)

A `passdb backend` szintén egy érdekes beállítási lehetőség. Itt tudjuk megadni, hogy milyen adatbázis alapján történjen a felhasználói hitelesítés. Ez alapértelmezésben az *smbpasswd* állomány, de akár használhatunk *LDAP*, vagy *MySQL* alapú adatbázist is. Ezekről még a későbbiekben ejtünk szót, most hagyjuk az alapértelmezett `smbpasswd` értéken.

A következő érdekesebb beállítás a `unix password sync`. Ezt a jelzőt állítva tudjuk megadni, hogy történjen-e egyeztetés a *UNIX passwd* és az *smbpasswd* állományok tartalma között, amennyiben a kódolt *SMB* jelszó megváltozik az *smbpasswd* állományban. Amennyiben `true` értéket adunk meg, úgy a beállított *passwd* program változó szerinti szkript a root felhasználó jogaival történő futtatásával felülírásra kerül a beállított *UNIX* jelszó a rendszerben.

A `printing` beállításra akkor lehet szükségünk, amennyiben a *Sambán* keresztül szeretnénk nyomtatókat is megosztani a hálózatban. Ez a paraméter írja le, hogy a nyomtató állapotinformációit milyen interpretáció alapján kell feldol-

gozni. Alapbeállítás az ***LPR***, vagy ***BSD***, de amennyiben például ***CUPS*** alapú nyomtatást használunk, úgy a `cups` érték megadása az ajánlott. A nyomtatás beállításaival a későbbiekben még bővebben foglalkozom.
Az általános beállításaim között az utolsó fontosabb a `domain master` paraméter beállítása. Ez az érték írja le. hogy a ***Samba*** kiszolgáló a kiszolgált tartományban, munkacsoportban a ***master browser*** szerepét felvegye-e, vagy sem. Az alapértelmezett érték az `auto`. Amennyiben mindenképpen ezt a gépet szeretnénk dedikálni a hálózatban ***master browser***-nek, úgy állítsuk `yes` értékre. Megjegyezném, hogy ***Windows NT Primary Domain Controller***-ek előszeretettel ragadják magukhoz ezt a feladatot, így amennyiben egy ***NT PDC***-vel rendelkező hálózatban ezt az értéket `yes`-re állítjuk és az ***NT PDC*** később csatlakozik a hálózathoz, mint a ***Samba*** kiszolgáló, úgy a két rendszer összeakadhat.
Ezzel a globális beállítások végére is értünk. Hangsúlyoznám, hogy ezek tényleg csak alapvető beállítások, ezzel egy működő, de meglehetősen egyszerű rendszert hoztunk létre. A ***Samba*** ennél lényegesen többet tud és ezeket végig is fogjuk nézni.
Most következzen a megosztások beállítása, elsőnek a rendszerben jelen lévő ***UNIX*** felhasználók ***home*** könyvtárának a kiosztása.

```
[homes]
comment = Home Directories
browseable = no
writable = yes
create mask = 0700
directory mask = 0700
```

Ezzel a beállítással elértük azt, hogy amennyiben egy olyan felhasználó csatlakozik a ***Samba*** kiszolgálóhoz, akinek van ***UNIX*** felhasználója is, úgy a saját és mindig csak a saját ***home*** könyvtára kiosztásra kerül. Ez egy nagyon egyszerű és jó megoldás az alapvető felhasználónként elkülönülő fájlmegosztási szolgáltatás biztosítására.
A két `mask` beállítási opció természetesen a fájl és könyvtár létrehozásra vonatkozóan azt mutatja, hogy a kiszolgálón milyen ***UNIX*** jogokkal kerüljenek az adott állományok létrehozásra.

```
[printers]
comment = All Printers
browseable = no
path = /tmp
printable = yes
public = no
writable = no
create mode = 0700
```

A `printers` bekezdésben a rendszerben megosztott nyomtatók beállítása történik, ebben a mappában kerülnek majd közzétételre a megosztott nyomtatók. Ezt az opciót a nyomtatómegosztás kapcsán részletesen fogom majd tárgyalni.

```
[pub]
comment = Samba server's PUB directory
writable = yes
locking = no
path = /pub
public = yes
```

Végül, de nem utolsó sorban nézzünk egy olyan megosztást, amelyet minden felhasználó látni fog a rendszerben, amelyben közös állományok kerülhetnek elhelyezésre.
A fenti beállítással egy írható, minden felhasználó által látható kiosztást hoztunk létre, amely a ***/pub*** mappára mutat a ***UNIX*** fájlrendszeren.
Ezzel az alapvető beállításokat el is végeztük, már csak annyi dolgunk van hátra, hogy a ***Samba*** felhasználóinkat létrehozzuk, illetve a felhasználókhoz jelszavakat állítsunk be. Amennyiben új felhasználót kívánunk a rendszerhez adni, úgy az `smbpasswd -a user` parancsot használjuk, amennyiben pedig a felhasználó jelszavát szeretnénk megváltoztatni, úgy az `smbpasswd user` parancsot.
A kiszolgáló elindítása után ejtsünk néhány szót olyan linuxos segédprogramokról, amelyekkel a ***Samba***, vagy a ***Windows*** hálózati megosztásait tudjuk kezelni.
Az egyik legfontosabb ilyen segédprogram gyűjteményt Debian alatt az `smbclient` csomag tartalmazza. Ezek olyan parancssoros eszközök, amelyek segítségével ***SMB*** protokollt támogató gépeken végezhetünk le lekérdezéseket (`smbclient`), küldetünk parancssorból nyomtatási parancsot (`smbspool`), valamint megnézhetjük például a hálózati struktúrát (`smbtree`).
Amennyiben ***Samba***, vagy ***Windows*** kiszolgálók megosztásait szeretnénk használni, úgy az `smbfs` csomag telepítésére lesz szükségünk. Ez a csomag tartalmazza a ***Samba*** állományrendszer becsatolásához szükséges eszközöket.
Amennyiben telepítjük a csomagot, akkor a `mount` parancsnak a `-t` kapcsolóval `smbfs` értéket adva be fogunk tudni Windowsos megosztásokat csatolni a ***UNIX*** fájlrendszerbe.
Amennyiben szeretnénk ilyen megosztásokat rendszeresen használni, úgy érdemes ezt a bejegyzést elhelyezni az ***/etc/fstab*** állományban, ahol a fájlrendszer típusának szintén az `smbfs`-t kell megadni.
Nézzünk egy egyszerű példát:

```
mount -t smbfs -o
username=viktor,password=MyPassword
➥ //myServer/viktor /mnt/tmp/
```

Ezzel a paranccsal a `myServer` kiszolgáló `viktor` kiosztását a `viktor` felhasználó jogaival az `/mnt/tmp` könyvtárba csatoljuk be.
Ezzel a sorozat nyitó cikkének végére is értünk. A folytatásban megpróbáljuk a ***Samba*** minden mélységét meghódítani. Érdemes lesz figyelni, mert egy nagyon hasznos eszközt kapunk egy jól beállított ***Samba*** személyében.

Illés Viktor (viktor@ei.hu)
23 éves, a BME műszaki informatikus szakának hallgatója, mellette weblapokkal, linuxos és windowsos rendszerekkel foglalkozik. Szabadidejét legszívesebben a szabadban tölti, teniszezik és kerékpározik.

FreeBSD – a szomszéd vár (4. rész)

Egy munkaállomás felépítése.

Legtöbbünknek van saját számítógépe, amelyet többnyire rendszeresen használunk a mindennapi életben: elektronikus leveleket fogalmazunk és olvasunk, turkálunk a világháló távoli bugyraiban, hivatalos leveleket írunk meg s nyomtatunk ki, esetleg táblázatokat-adatbázisokat hozunk létre, illetve néha játszunk vagy filmet nézünk. A saját géptermünkben rendszergazdák és felhasználók vagyunk egy személyben. A hiedelmekkel ellentétben – avatatlan szem számára – a „vár díszterme” pont úgy néz ki, mint a vendégeké... nézzünk tehát a díszletek mögé.

A grafikus felület

Bár az ***igazi UNIX*** felhasználó egy szöveges konzolon minden felmerülő problémát meg tud oldani, valljuk be: jól jön a grafikus felület, amely természetesen ***FreeBSD*** esetén is rendelkezésre áll. Talán nem okoz meglepetést, ha a jó öreg ***X11R6*** felülettel találkozunk, azonban a konkrét implementáció a megszokott ***XFree86*** helyett már itt is az új kiadás: az ***X.org*** lesz (meg kell jegyeznek, hogy a ***FreeBSD*** verziószámozása egy kis késéssel szokta követni a Linux alatt megjelenő verziókat, hiszen át kell alakítani egy kicsit, hogy a ***FreeBSD*** rendszermaggal együtt tudjon működni). Ennek megfelelően a beállítás és a használat teljesen azonos módon történik, mint egy átlagos Linux esetén. Különbség abban lehet, hogy némely Linux terjesztő saját programot készít a grafikus felület automatikus beállítására, amely jelentősen segít a kezdő felhasználónak – ezt ***FreeBSD*** esetén mellőznünk kell. Alapvetően két úton tudjuk beállítani az ***X.org*** programot, az egyik nehézkes, a másik járhatatlan. Tradicionális módszer az `xorgconfig` használata, amely szöveges módban fut, s olyan kérdéseket képes feltenni, amelyek megválaszolásába még egy jól felkészült informatikus mérnök is belesülhet. Ezért aztán az `xorgcfg` programot szoktuk használni, mivel ez képes felderíteni a gépünkben lévő videókártya típusát-jellemzőit, majd ez alapján már grafikus felületen tudjuk a beállításokat eszközölni; a használata viszonylag egyszerű, bár kényelemben messze van az ideális állapottól. Létezik egy harmadik út is, amely egy kis ***csalással*** sok nehézségtől tud megkímélni minket: ha a gépünkön van már ***Linux*** rendszer, akkor egyszerűen csak át kell másolni az ***XF86Config*** nevű állományt a ***/etc/X11/*** könyvtárba, ahol majd az ***X.org*** keresni fogja, és nagy valószínűséggel működni is fog ezen állomány alapján. Azért csak ***valószínűleg***, mert előfordulhat, hogy a videókártyát nem az ***X.org*** támogatja, hanem a gyártó ad ***Linux*** meghajtóprogramot a kártyához, s ekkor – többnyire – nem tudjuk teljes mértékben kihasználni a kártya tudását. Általában ez a hardveres 3D gyorsítást érinti, amelyet mellőznünk kell, ha ***nem NVidia*** kártyánk van. Ezt a kivételt azért említem meg, mert az ***NVidia*** ezidáig az egyetlen gyártó, amely nemhogy a Linux rendszert, hanem a ***FreeBSD***-t ***is*** támogatja saját készítésű rendszermag modullal.

Sokan nem is szembesülnek a gyártó által adott támogatás hiányával, hiszen nem tudják, miből maradnak ki, ezért írnék néhány példát, amire képesek leszünk ***FreeBSD*** alatt az ***NVidia*** modult használva (természetesen ugyan ezek használhatók Linux esetén is). Az ***X.org*** önmagában nem igazán kezeli a kártyák TV kimenetét, illetve nem mindig kapunk megfelelő minőségű képet. A TV kimenet használatához az ***XF86Config*** állományt kell szerkeszteni, mégpedig a `Section „Device”` résznél kell a vége felé beilleszteni a következő sorokat:

```
Option  “TwinView”              “true”
Option  “TwinViewOrientation”   “Clone”
Option  “MetaModes”             “1024x768,1024x768”

Option  “SecondMonitorHorizSync”        “60”
Option  “SecondMonitorVertRefresh”      “50-70”
Option  “ConnectedMonitor”              “TV, CRT”
```

Az ***X*** kiszolgálót újraindítva meg fog jelenni a TV képernyőn is a monitoron látható kép. Sajnos munkára nem

lehet használni a TV kis felbontása miatt, viszont filmek – élvezhető – megtekintésére annál inkább. Lehetőségünk van a ***Dual View*** kihasználására is, amely lehetővé teszi két monitor egyidejű használatát. Ehhez olyan videókártya kell, amelynek két monitorcsatlakozása van, vagy egy átlagos laptop is megteszi, amelyekben az ***NVidia*** vezérlő többnyire tudja ezt a működési módot. Otthoni körülmények esetén kevéssé használható, viszont az oktatásban látom nagy előnyét, hiszen így egy kivetítővel sokkal kényelmesebb a munka. A munkaasztal ezáltal méretben megduplázódik (beállítás függvénye, hogy magasabb vagy szélesebb lesz), így kétszer akkora helyünk lesz, mint eddig, és ebből a hallgatók csak az egyik részt látják. A kivetítőn futhat a bemutatott program, vagy a prezentáció, a munkaasztal másik részén pedig a előadás következő fázisának előkészítése, vagy egy új program indítása történhet, s ezt nem látják az előadás résztvevői. Ezáltal sokkal rugalmasabb és szebb lehet az előadás. Ehhez egyszerűen csak a fent leírt részt kell egy kicsit átszerkeszteni:

```
Option  "TwinView"              "true"
Option  "TwinViewOrientation"   "RightOf"
Option  "MetaModes"             "1024x768,1024x768"

Option  "SecondMonitorHorizSync"        "30-68"
Option  "SecondMonitorVertRefresh"      "50-70"
Option  "ConnectedMonitor"              "DFP, CRT"
```

Miután szükségeinknek megfelelően beállítottuk, és sikeresen el is indítottuk az ***X*** felületet, már csak a megszokott ablakkezelő felület feltelepítése van hátra, amelyek közül a „nagyok" minden bizonnyal, de a „kicsik" nagy része is többnyire telepíthető a ***ports*** adatbázisból; sajnos a verziók szintén egy kis késéssel jelennek meg. A többségünk ***KDE***-t vagy ***GNOME***-ot használ, jelenleg a 3.3.2 verzió telepíthető a ***KDE*** ablakkezelő rendszerből, illetve a ***GNOME 2.8.1*** is elérhető már (bár mire e sorokat újságban olvassuk, már a ***GNOME 2.8.2*** is elérhető lesz). Mind a két ablakkezelő rendszer szinte összes funkciója azonos módon használható, mint Linux esetén, esetleg egy-két hardverközelibb (például az ***APM*** vagy az ***APCI*** kezelése laptopok esetén) segédprogram működése azért akadozhat, hogy azért ne legyen olyan szép a kép...

```
#!/bin/sh
case $1 in
start)
  /usr/local/bin/kdm
  ;;
esac
```

Kezdők számára problémát okozhat, hogy (például) a feltelepített ***KDE*** nem fog önmagától elindulni, mint ahogy a grafikus felület sem, ezt minden esetben nekünk – mint rendszergazdának – kell megtennünk, parancssorban kiadva a `kdm` parancsot. Ezt természetesen akár automatizálhatjuk is, ez lényegében egy egyszerű programocska készítését jelenti, amelyet a ***/usr/local/etc/rc.d/*** könyvtárba kell elhelyeznünk, a neve lehet `kdestart.sh`, a tartalma pedig a fent látható néhány sor (ne felejtsünk el futtatási jogot is adni erre az állományra, különben nem fog induláskor lefutni). Ennek hatására a ***KDE*** bejelentkezés-kezelője – a ***KDM*** – fog elindulni, s ezzel már bármilyen telepített ablakkezelőt el tudunk indítani, illetve a felhasználóink is könnyebben képesek a gépet használni. Más bejelentkezés-kezelőre is könnyedén át lehet írni a kis programot, egyszerűen meg kell keresni a megfelelő állományt, amely indítja azt.

A hang

A hangkártya már minden számítógép szerves tartozéka, szinte már nem találunk olyan munkaállomásnak szánt alaplapot, amelyben nem található integrált hangvezérlő. A ***FreeBSD*** szinte az összes kommersz célra szánt hangkártyát (jobban mondva a vezérlő lapkát) ismeri, képes azt használni. Ez azonban csak egy szűk körét jelenti a forgalomban lévő eszközöknek – gondolok itt elsősorban a modern 5.1 kimenettel bíró házimozi rendszerekkel együttműködni képes kártyákra – amelyeket még ***Linux*** alatt is csak nehézkesen vagy egyáltalán nem tudunk használni. Ebben az ügyben tehát a csodát ne a ***FreeBSD*** rendszertől várjuk, az is nagy öröm, ha sikerül egy olyan hangkártyából hangot csiholni, amely már Linux alatt bizonyított.
Alapvetően az összes szükséges modul rendelkezésre áll, csak azt kell betöltenünk, amelyik a kártyát kezelni fogja. Ezt a tevékenységet ésszel vagy erővel tudjuk végezni. Az első módszer szerint megpróbáljuk kideríteni, hogy a kártyán milyen lapka van, és a modul nevéből kikövetkeztetjük, hogy kezeli-e esetleg. A használható modulokat a ***/boot/kernel/*** könyvtárban találjuk meg, akár ki is listázhatjuk `snd_*.ko` mintára illeszkedő állományokat. A modul betöltése például a `kldload snd_ich` parancs kiadásával történik. Ha inkább a nyers erő mellett szavaztunk volna, akkor az előbb említett könyvtárban kiadott `kldload snd_*.ko` parancs is sikerhez vezethet, ekkor ugyanis az összes modult betöltjük: valamelyik csak kezeli a gépünkben található kártyát.
Ha sikeresen eltaláltuk a kártyára szerelt lapka típusát, s a rendszermag is képes volt megfelelően kideríteni a modulhoz rendelhető hardverértékeket (I/O, IRQ, DMA), akkor a mag üzenetei között meg kell látnunk a kártya bejelentkezését (ezt a `dmesg` parancs kiadásával láthatjuk, illetve az első konzolra is kiíródnak ezek a szövegek):

```
pcm0: <Intel ICH3 (82801CA)> port 0xcdc0-0xcdff,
      ➥0xce00-0xceff irq 11 at device 3
1.5 on pci0
pcm0: [GIANT-LOCKED]
pcm0: <Yamaha YMF753 AC97 Codec>
```

Minden más – nem ehhez hasonló – üzenet azt jelzi, hogy az áhított zenekar nem vállal egyszerű fellépést a vár dísztermében... :(

Az USB támogatás

Két éve senki sem gondolta volna, hogy az ***USB*** ekkora karriert fog befutni, s manapság már szinte mozdulni sem lehet ***USB*** támogatás nélküli géppel. Gondoljunk csak a flopilemezeket felváltó az ***USB „kulcstartóra"***, amely fizikai méretben sokkal kisebb, tárkapacitásban viszont sokkal tágabb „elődjénél". A ***FreeBSD*** alapvetően támogatja az alaplapi

USB lapkákat, legyen szó az *USB 1.1* vagy az *USB 2.0* kiadásokról; ellenben az *USB* eszközök egy részét nem tudjuk még életre kelteni, bár határozott fejlesztőmunka van ezen a téren is. Az *USB* tárolók nagy hányada támogatott, csatlakoztatva a megfelelő helyre a rendszermag üzenetei között meg kell látnunk a következő üzenethez hasonlót:

```
umass0: USB 2.0 Flash Disk USB Mass Storage Device,
        ➥rev 2.00/0.01, addr 3
da0 at umass-sim0 bus 0 target 0 lun 0
da0: <USB 2.0 Flash Disk PROL> Removable Direct
     ➥Access SCSI-0 device
da0: 1.000MB/s transfers
da0: 125MB (256000 512 byte sectors: 64H 32S/T 125C)
```

A létrejövő /dev/da0 eszközt – a rajta lévő fájlrendszernek megfelelően – tudjuk felcsatolni, s írni-olvasni a tárterületet. A fájlrendszer kiderítése empirikus módszerekkel is lehetséges, de javaslom a `disktype` használatát, amely megmondja egy tetszőleges eszközről, hogy milyen fájlrendszert találunk rajta:

```
$ disktype /dev/ad0s2

-- /dev/ad0s2
Character device, unknown size
Ext2 file system
  UUID 222F5D0C-F0BD-4D33-8491-347734D7DFBB
  ➥(DCE, v4)
  Volume size 10.00 GiB (10742214656 bytes, 2622611
  ➥blocks of 4 KiB)
```

A digitális fényképezőgépek támogatottsága kicsit felemás. Amelyek képesek *USB Storage* módban működni, azok probléma nélkül csatlakoztathatók *FreeBSD* alatt is, amelyek csak a saját programjukkal értik meg egymást, azok nagy része a `gphoto2` csomaggal lesz használható. A fennmaradó néhány típust sajnos nem tudjuk majd használni.
Gyakori, hogy *USB-IDE* átalakítón át csatlakoztatunk *IDE* eszközöket a gépünkre. Ennek főleg hordozható gépek esetén van értelme, például egy *DVD-*írót tudunk ilyen módon egy laptophoz csatlakoztatni (amely például nem tartalmaz *DVD-*írót). A legtöbb ilyen átalakító támogatott *FreeBSD* alatt is, csatlakoztatás után a rendszermag üzenetei között megtaláljuk a létrejött eszköz nevét, amelyet céljának megfelelően tudunk használni.

```
umass0: Acer Labs USB 2.0 Storage Device, rev
        ➥2.00/1.09, addr 2
cd0 at umass-sim0 bus 0 target 0 lun 0
cd0: <HL-DT-ST DVDRAM GSA-4082B A201> Removable
     ➥CD-ROM SCSI-0 device
cd0: 1.000MB/s transfers
cd0: cd present [2236704 x 2048 byte records]
```

Esetleg nyomtatót is rákötethetünk az *USB* csatlakozóra, legfeljebb nem tudunk rajta nyomtatni. Ha `ulpt0` lesz az eszköz neve, akkor bizakodhatunk, mert egy kis lépés kell már csak a nyomtató felélesztéséhez (amelyet a következő részben írok le, mert kis lépés ugyan, de nem rövid a leírása :).

```
ulpt0: hp deskjet 5100, rev 2.00/1.00, addr 2,
       ➥iclass 7/1
ulpt0: using bi-directional mode
```

Akár egy *USB Bluetooth* adapter is szóba jöhet, a nagy részük támogatott már *FreeBSD* alatt is, nem lehet már akadály a mobiltelefonnal vagy kéziszámítógéppel való kommunikáció sem.

```
ubt0: Cambridge Silicon Radio Ltd. Bluetooth USB
      ➥dongle, rev 1.10/5.25, addr 2
ubt0: Cambridge Silicon Radio Ltd. Bluetooth USB
      ➥dongle, rev 1.10/5.25, addr 2
ubt0: Interface 0 endpoints: interrupt=0x81,
      ➥bulk-in=0x82, bulk-out=0x2
ubt0: Interface 1 (alt.config 5) endpoints:
      ➥isoc-in=0x83, isoc-out=0x3;
      ➥wMaxPacketSize=49; nframes=6,
      ➥buffer size=294
```

Mint látjuk a *FreeBSD* elég jól lekezeli az *USB* eszközök széles választékát, bár kevesebb eszközt tudunk majd ténylegesen használni, mint Linux esetén – vásárláskor jobban meg kell fontolni az új eszköz megvételét.

Hálózati eszközök

Néhány év alatt a legfontosabb kommunikációs felület a helyi hálózat, illetve ezen át az internet elérése lett. Szinte már nem is találni olyan számítástechnikát alkalmazó vállalatot, ahol nincs kiépítve helyi hálózat. A *FreeBSD* egyik fő erőssége a hálózati szolgáltatások nyújtása a helyi hálózat, illetve az Internet felé, ezért fontos kérdés, hogy milyen eszközöket támogat.
Az *Ethernet* felülettel rendelkező kártyák szinte teljes típusválasztéka támogatott, nagyon szerencsétlennek kell ahhoz lennünk, hogy a 10/100 kártyák közül pont olyannal rendelkezzünk, amelyet még nem támogat. A legújabb gigabites kártyák még tartogathatnak meglepetéseket, de itt is gőzerővel folyik a fejlesztés.
Korunk slágerével, a WLAN kártyákkal már nem ilyen szép a helyzet, ugyanis ezen eszközök támogatási térképén sok még a fehér folt, bár vannak nagyon érdekes kezdeményezések ezen a téren is. Érdemes megnézni a ➲ http://www.freebsd.org címen a kézikönyv Wireless Network fejezetét, illetve magyar nyelven ennek tömörebb változatát, amely (pillanatnyilag) a ➲ http://www.bsd.hu/dokumentacio/haladoknak/FreeBSD-WLAN/view címen érhető el, és nagyon jól összefoglalja azt, amit a témáról tudni érdemes.
Természetesen otthoni körülmények között a modemet se temessük el, bár csak azokat tudjuk használni, amelyek úgynevezett hardveres modemek, tehát nem a *WinModem* néven árult olcsó kártyák. Sajnos manapság alig-alig kapni olyan modemet, amely soros portként jelenik meg az operációs rendszer felé, így sajnos *FreeBSD* esetén a jelenleg kapható modemek nagy része nem fog megszólalni...
Pár éve a *GPRS* kapcsolat is – elérhető áron – komoly alternatíva lett, ehhez egyszerűen egy mobiltelefon kell, amely képes *GPRS* szolgáltatásra, és valamilyen módon képesek vagyunk a géphez csatlakoztatni. Ez többféle módon is

lehetséges: kábellel, infrán vagy *Bluetooth* felületen át. Mind a három módszer gyengén támogatott *FreeBSD* alatt, mivel a kábelek eléggé speciális felülettel rendelkeznek, ezeket néha *Linux* alatt is nehéz életre kelteni; az infra kapcsolat szinte teljes egészében kimaradt a *FreeBSD* kernelből; a *Bluetooth* pedig még új technológia, bár a legtöbb Bluetooth lapkát már ismeri a *FreeBSD* is. Részemről ez utóbbi javasolnám a telefonnal való kapcsolat kiépítésére. A megnevezett eszközökkel a hálózati kapcsolat kiépítését (modemmel, *ADSL* vagy *GPRS* útján) a következő részben írom le.

Fájlrendszerek felcsatolása

A *FreeBSD* kicsit más módon oldja meg a különféle eszközök fájlrendszerbe csatolását, mint azt *Linux* alatt megszokhattuk. A felcsatolásnál a fájlrendszer típusát kétféle módon adhatjuk meg, vagy a `mount` parancs `-t` paraméterével, vagy közvetlenül a parancs nevében adjuk meg: `mount_ext2fs`. Az első eset tulajdonképpen meghívja azt a parancsot, amit a paraméterben kapott név meghatároz, de mégis „szebben” néz ki. A felcsatolás lényege – mint említettem – azonos, mégis van néhány eleinte sőt akár hosszú távon bosszantó tulajdonsága. Példának okáért az egyik ilyen a karakterkészlet megadásának hiánya, amely akkor jönne jól, ha egy nemzeti karakterekkel teli *CD/DVD* lemezt kellene használni, mivel ez mindig a beállított kódlapnak megfelelően jelenik meg, a kódlapot átállítva pedig a régi bejegyzések lesznek olvashatatlanok. Elég rég óta ígérik ennek a problémának a megoldását, de ezidáig nem került bele az alaprendszerbe.
A másik fontos hiányosság, hogy a felhasználók nem képesek a fájlrendszerbe eszközt csatolni, ugyanis erre csak a *root* felhasználó képes. Ez különösen akkor zavaró, ha egy munkaállomást többen is használnának, s közülük nem mindenki „megbízható” felhasználó (gondoljunk csak egy iskolai gépteremre). Ekkor szinte megoldhatatlan a feladat, hogy egy egyszerű flopit, *CD*-lemezt vagy *USB* tárolóeszközt bármelyik felhasználó használni tudjon. A felcsatolás még megoldható, hogy például a `mount_ext2fs` programra ***root setuid*** bitet állítunk be, ekkor ugyanis az adott típusú fájlrendszert bárki képes lesz felcsatolni. A felhasználók körét *ACL* jogok beállításával tudjuk szűkíteni, viszont a lecsatolás mindenképpen gondot fog okozni: az összes fájlrendszer típushoz csak egy `umount` parancs tartozik. Ha ebből csinálunk több másolatot, akkor is csak annyit érünk el, hogy bizonyos felhasználók képesek a nekik megengedett fájlrendszer típusokat felcsatolni és ***lecsatolni***, ***függetlenül attól***, hogy ők csatolták-e ***fel***. Ne is kísérletezgessünk a probléma megoldásán, többnyire észrevétlen biztonsági réseket tudunk csak ütni – az amúgy biztonságos – várfalon.

A játékok

Azt kell mondjam, hogy a *FreeBSD* nem egy játékplatform, de annak sem kell aggódnia, aki játszani szeretne. Elsőnek rögtön itt van a `freebsd-games` csomag, amely a tradicionális *BSD* játékgyűjteményt tartalmazza. Ilyen például a `hangman`, amely egy a klasszikus akasztófajáték, természetesen angol nyelven, és szöveges felületen. Ezek kedves kis játékok a régi szép időkből.
Aki többre vágyik, feltelepítheti a *ports* adatbázis `games` szekciójából bármelyik játékot, ám érdemes előtte elolvasni a leírását, mivel a többségük egyszerű grafikával, és minimális kidolgozottsággal rendelkezik, noha a játék maga remekül el van készítve: a programozóknak nem a grafikák és a hangok készítése a kedvenc tevékenység, hanem a program megírása.
Néhány kivételt érdemes megemlíteni, ilyen például a *FreeCiv*, amely *FreeBSD* alá is feltelepíthető, szépen kidolgozott grafika és hang jellemzi, és egy olyan légkör, amelyből nehéz kimászni, ha belemerültünk. Aztán itt a *Quake* trilógia (a *quakeforge* és a *quake2forge* a *ports*-ból telepíthető, a *QuakeIII* esetén a *Linux* verziót kell telepíteni), amelyet szintén képesek vagyunk játszani, a két utolsó részéhez azonban nem árt a hardveres 3D gyorsítás megléte, különben könnyen képregényhez hasonló játékmenet lesz a végeredmény.

Az irodai programok

Alapvetően a *FreeBSD* nem rendelkezik irodai munkára alkalmas programcsomaggal, azonban van néhány, amelyet telepíteni tudunk. Ezek közül szövegszerkesztésre az *AbiWord* programot használhatjuk, amely az egyszerűbb munkákra tökéletes. Akinek ez nem elég, telepítheti a *KOffice* programcsomagot is, amely több olyan kisebb program együttese, amelyekkel szinte minden otthon felmerülő munkát meg lehet oldani. Természetesen telepíthető az *OpenOffice.org 1.1.3* is, remélem mindenkinek ismerősen cseng a neve. Ez utóbbiból érdemes egy előre lefordított bináris csomagot megszerezni, mivel a fordításához akár egy nap és 4-5 gigabájt adatterület szükséges.

A multimédia használata

Zene hallgatása és film megtekintése a multimédia alapja, s ezeket nem is kell mellőznünk az életünkből *FreeBSD* rendszert használva. A két használandó program mindenki számára ismerős szokott lenni: az *XMMS* és az *MPlayer*. Egyszerűen csak annyit kell tennünk, hogy feltelepítjük ezeket a programokat, a használatuk is teljesen azonos, mint azt *Linux* esetén megszoktuk.

Auth Gábor (auth.gabor@enaplo.hu)
Egy pécsi középiskolában informatikát és programozást oktat. Tíz éve botlott először a UNIX rendszerekbe, 7 év Linux használat után kapta el a FreeBSD lázat, amiből máig nem tudott kigyógyulni.

KAPCSOLÓDÓ CÍMEK

A FreeBSD projekt honlapja
➲ http://www.freebsd.org

A magyar FreeBSD honlap
➲ http://www.freebsd.hu

A magyar BSD honlap
➲ http://www.bsd.hu

A kézikönyv magyar fordítása
➲ http://www.enaplo.hu/FreeBSD/handbook/

Az OpenOffice.org programozása

Egy elveszett fejlesztőkörnyezet titkai kedvenc irodai csomagunkban.

Az *OpenOffice.org* révén a *Linux* mára egyértelműen eljutott arra a szintre, hogy teljes értékű helyettesítője lehet egy windowsos irodai munkaállomásnak. Mit is jelent ez? Tartalmaz egy szövegszerkesztőt, egy táblázatkezelőt, van benne bemutató-készítő, egy rajzoló-program, sőt mi több, egy remekbe szabott képletszerkesztőt is találunk a csomagban. Mindezt egy ízléses felhasználói felületen, és nem utolsó sorban széles fájlformátum támogatással.

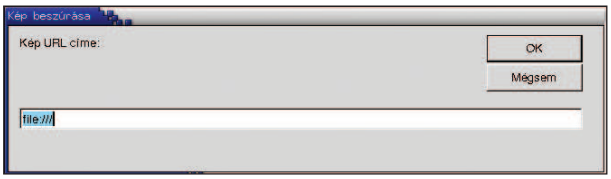
Mindenünk megvan? Mint tudjuk, az *MS Office* tartalmaz egy önálló adatbázis kezelőt, *Access* néven, itt mintha ez hiányozna. Ám a súgó rövid tanulmányozása után azonnal kitűnik, hogy nem erről van szó. Az *Eszközök* menü *Adatforrások...* menüpontja alatt beállíthatjuk, hogy mely *Adabas*, *MySQL*, *dBase*, tetszőleges *ODBC*, vagy *JDBC* adatforrást szeretnénk használni. Természetesen a sima szöveges állomány, illetve a munkafüzet is használható.

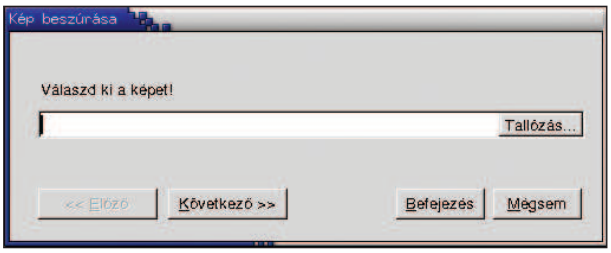
Mintha még mindig nem lenne teljes a kép. Gondolkozzunk csak egy picit. Hát persze, a makrók, és az ezekkel járó vírusok. Ez az, aminek a hiánya eddig álmatlan éjszakákat okozott mindannyiunknak, hát végre *Linux* alatt is elérhetőek! Na jó. Hosszú és parttalan vitát indítana el, ha azt kezdeném el boncolgatni, hogy mi értelme egy közel teljes értékű fejlesztőkörnyezetet építeni egy szövegszerkesztőbe. Inkább nézzük meg, mi mit nyerhetünk abból, ha tudjuk ezt használni.

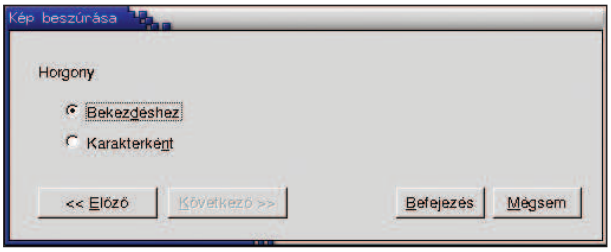
Essünk neki! Nyissunk meg egy új szöveges dokumentumot bármely *OpenOffice.org* alkalmazásból a *Fájl*/*Új*/*Szöveges dokumentum* menüponttal. Azonnal mentsük is el egy beszédes nevű állományba a Fájl/Mentés másként... menüpont segítségével. Legyen mondjuk ez az *ElsoMakrom.sxw*. Ezután válasszuk ki az *Eszközök*/*Makrók*/*Makró...* menüpontot és időzzünk el egy pillanatra.

A most látható ablakban találjuk az elérhető makrók listáját, gyűjtőkbe szervezve. A bal oldali fa struktúra tartalmazza az előbb említett gyűjtőket, azaz azokat a helyeket, ahol *Basic* nyelven írt makrókat találhatunk. Nem említettem volna? Az *OpenOffice.org*-ban is egyfajta objektum-orientált *Basic* nyelven dolgozhatunk, akárcsak a másik irodai csomagban. Visszatérve a gyűjtőkre, egy *soffice* nevűt mindenképp látni fogunk, az ebben található modulok makróit bármely dokumentum elérheti. Ezen kívül minden megnyitott dokumentum egy további gyűjtőt képez.

Dolgozzunk tehát a saját állományunkba. Válasszuk ki a *Makró* forrásaként az *ElsoMakrom.sxw* gyűjtő alatti *Standard könyvtárat*, és kattintsunk az *Új* gombra. Ekkor az *OpenOffice.org* megkér, hogy határozzuk meg a modul nevét. Fogadjuk el nyugodt szívvel a *Module1* nevet, és kattintsunk az *Ok* gombra. Ekkor elindul a *Basic IDE (Integrated Development Environment - Integrált Fejlesztői Környezet)*, melyben két üres eljárás fogad minket.







Mielőtt továbblépnénk, vessünk még egy pillantást a makrók szervezése mögötti elképzelésre. A gyűjtő egy logikai szervező egység, amely egy-egy dokumentumot jelképez, vagy adott esetben maga az *soffice*, amely bárki számára elérhető. A gyűjtők alatt található könyvtárak olyan egységek, melyek moduloknak adhatnak helyet. A modulok jelentik a legkisebb egységet. Egy modul egy forrásállományt jelent, amely természetszerűleg eljárásokból épül fel. Utóbbi eljárások a makrók.

Ezek szerint az előbb látott két üres eljárás, a *Main* és a *Macro1* egyből két új makrót jelent? A válasz igen. Zárjuk be a *Basic IDE*-t és keressük meg újra a *Makró...* menüpontot. Kiválasztva a dokumentumunk *Standard* könyvtárának *Module1* modulját, látható, hogy az előbb

még üres jobb oldal már két makrót tartalmaz. Válasszuk ki bármelyiket és kattintsunk a Szerkesztés gombon. Így ismételten a fejlesztőkörnyezetben találjuk magunkat. A fejlesztőkörnyezet színkiemeléssel segíti a munkát, és hely érzékeny súgó is van, amit az ***F1*** billentyűvel csalogathatunk elő. Sajnos ez utóbbi angol nyelvű, viszont annál bőségesebb. Lássunk végre egy példát egy valódi makróra! Először is töröljük a ***Macro1*** eljárást, majd a Main rutint az alábbiaknak megfelelően írjuk meg:

```
Sub Main

   kepURL = InputBox ("Kép URL címe:", "Kép
   ➥beszúrása", "file:///")
   if kepURL = "" then ' a felhasznalo Megse-t
   ➥nyomott
      exit sub
   endif

   dokumentum = ThisComponent
   szoveg = dokumentum.getText ()

   grafika = dokumentum.createInstance
   ➥("com.sun.star.text.GraphicObject")
   grafika.GraphicURL = kepURL

   lathatoKurzor = dokumentum.getCurrentController
   ➥().getViewCursor ()
   szovegKurzor = szoveg.createTextCursorByRange
   ➥(lathatoKurzor.getStart ())

   szoveg.insertTextContent (szovegKurzor.getStart
   ➥(), grafika, false)

End Sub
```

Ennek az egyszerű makrónak az a feladata, hogy egy képet szúrjon be arra a helyre, ahol a kurzor áll. Megjelenít egy beviteli ablakot, ahol a kép URL-je határozható meg. Az ***Ok*** gombra kattintva a kép beszúrásra kerül. Ennél azonban felhasználóbarátabb megoldást is alkalmazhatunk. Készítsünk egy igazi tündért, és helyettesítsük ezzel a ***Beszúrás*** menü ***Kép*** pontját.
Ehhez már egy önálló párbeszédablakra lesz szükségünk. A ***Basic IDE***-ben alul láthatjuk azt a fület, amely jelzi, hogy épp a ***Module1*** modult szerkesztjük. Kattintsunk ezen a fülön jobb egérgombbal és válasszuk az előugró helyi menüből a ***Beszúrás/BASIC***-párbeszédablak pontot. A párbeszédablak szerkesztőjét látjuk, egyelőre egy üres ablakkal. Ettől kezdve az alul található fülekkel ugrálhatunk a forráskód és az ablak szerkesztője között.
Vezérlők elhelyezéséhez meg kell nyitnunk a ***Vezérlőelemek*** ablakot. Ezt a felül található eszköztárak azon ikonján történő kattintással csalogathatjuk elő, melyen egy jelölőnégyzet, egy választógomb, valamint egy gördítősáv egyszerre látható. Innen kedvünkre válogathatunk a vezérlők között az olyan egyszerűektől, mint a nyomógomb, egészen az olyan összetettekig, mint a fájlkiválasztás.
Rakjunk is fel egy fájlkiválasztó elemet az ablakra. Az elem felhelyezése, átméretezése és mozgatása magától értetődő, játszunk is el vele egy picit. Ha már elégedettek vagyunk a munkánkkal, helyezzünk el négy új gombot is. Bármely elemen a jobb egérgombbal előhívhatjuk a helyi menüt. Itt a ***Tulajdonságok...*** menüpont kiválasztásával, vagy az elemen kettős kattintással megnyithatjuk az elem Tulajdonságok lapját. A gomboknak adhatunk saját nevet, melyre a makrókból hivatkozhatunk, illetve egy szövegcímkét. Első gombunkat nevezzük el `Elozo`-nek, címkéje legyen ***„<< Előző"***. A következő neve legyen `Kovetkezo`, címkéje pedig ***„Következő >>"***. A továbbiaknak legyen `Befejezes` neve, ***Befejezés*** címkéje, illetve `Megsem` neve, ***Mégsem*** címkéje. Az `Elozo` gomb ***Engedélyezett mezőjét*** pedig állítsuk ***Nem*** értékre.
A felhasználóbarátság jegyében adjunk hozzá az ablakhoz még egy egyszerű címkét, és írjuk bele a ***„Válaszd ki a képet!"*** szöveget. A teljes ablaknak is előhívható a Tulajdonságok lapja, így lehet neki címet adni. Végezetül a ***Vezérlők*** eszköztár utolsó ikonjára kattintva bekapcsolhatjuk a ***Tesztüzemmódot***, ezáltal kipróbálhatjuk, hogyan is fog kinézni az ablak, amikor élesben használjuk.
A ***Tesztüzemmódot*** az ESCAPE billentyűvel hagyhatjuk el. Ahhoz, hogy gombjaink értelmes életet éljenek, eljárásokat kell hozzájuk írnunk, majd ezeket az eljárásokat össze kell kötni a gombok megfelelő eseményeivel. Váltsunk tehát vissza a kódszerkesztőre és programozzunk egy kicsit.
A párbeszédablakunk referenciáját modulszinten határozzuk meg, ez már majdnem olyan, mint egy globális változó. A megvalósítás azonban így jelentősen egyszerűsödik.

```
Private parbeszedAblak as Variant

Sub Main

   parbeszedAblak = createUnoDialog
   ➥(DialogLibraries.Standard.Dialog1)
   parbeszedAblak.execute ()

End Sub

Sub ParbeszedAblakMegsem

   parbeszedAblak.endExecute ()

End Sub

Sub ParbeszedAblakBefejezes

   Dim kepURL as String

   parbeszedAblak.endExecute ()

   ' Eleresi utat fogunk kapni, ezt at kell
   ➥alakitani
   kepURL = ConvertToURL
   ➥(parbeszedAblak.Model.FileControl1.Text)

   dokumentum = ThisComponent
   szoveg = dokumentum.getText ()

   grafika = dokumentum.createInstance
   ➥("com.sun.star.text.GraphicObject")
```

```
   grafika.GraphicURL = kepURL

   lathatoKurzor = dokumentum.getCurrentController
   ➥().getViewCursor ()
   szovegKurzor = szoveg.createTextCursorByRange
   ➥(lathatoKurzor.getStart ())

   szoveg.insertTextContent (szovegKurzor.getStart
➥(), grafika, false)

End Sub
```

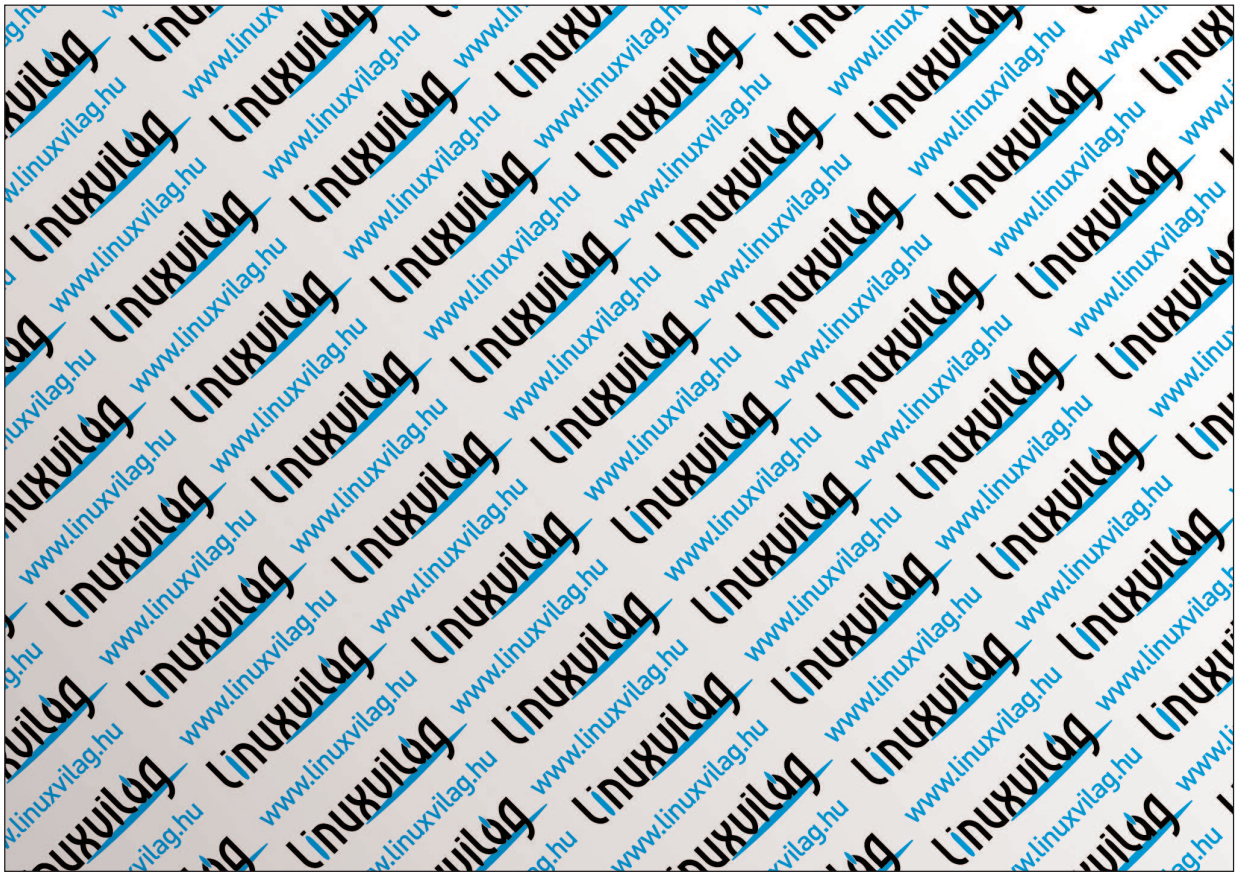
A régi `Main` eljárás `ParbeszedAblakBefejezes` nevet kapott, és ennek megfelelően módosult. Az új `Main` létrehoz egy példányt az ablakból és elindítja. A `ParbeszedAblakMegsem` mindössze bezárja az ablakot. A két új eljárást a megfelelő gombokra történő kattintás eseményéhez kell rendelni. Ehhez váltsunk vissza a párbeszédablak szerkesztőre és a gombok Tulajdonságok lapjának második fülén az Inicializáláskor mezőt töltsük ki. Ehhez nincs szükség gépelésre, a mellette található gombbal tallózhatunk a makrók, azaz az eljárások között.
A közel kész modul `Main` makróját elindítva előugrik az ablak, és már képet is tudunk beszúrni. Viszont nem tündér a tündér, ha nincs több oldal, amin a ***Következő*** gomb folyamatos nyomkodásával nem lehet ugrálni, így hát még dolgoznunk kell egy kicsit. Ne szaladjunk azonban előre, hiszen már most is használható makróval van dolgunk. Tegyük még kényelmesebbé a használatát azzal, hogy kiteszünk az eszköztárra egy gombot, amivel azonnal indítható. Egyszerűen kattintsunk az eszköztáron jobb gombbal, és válasszuk a ***Testreszabás...*** pontot. Bal oldalon görgessünk az ***ElsoMakrom.sxw BASIC*** makrók elemig és bontsuk ki. Keressük meg a ***Standard*** könyvtár ***Module1*** moduljának `Main` makróját. Ezután keressük meg a jobb oldalon azt a pontot, ami alá szeretnénk beszúrni az új gombot. Ezután kattintsunk a ***Hozzáadás —>*** gombra. Jópofa ikonokhoz kattintsunk az ***Ikonok...*** gombon. Ellenőrizzük, hogy az új gomb melletti jelölőnégyzet be van-e jelölve. Ha minden rendben, kattintsunk az ***Ok***-ra, és élvezzük az egykattintásos makróindítás hihetetlen élményét.
Térjünk vissza tündérünkhöz és adjunk értelmet az `Elozo` és `Kovetkezo` gomboknak. Mint tudjuk, a beszúrt képet elhelyezhetjük egy, a bekezdéshez kötött horgonyhoz képest, illetve beszúrhatjuk úgy is, mintha egy karakter volna. Bízzuk ezt a döntést is a felhasználóra azáltal, hogy a tündér következő oldalán két választógombbal szembesítjük. Az egyikkel bekezdéshez horgonyozhatja a beszúrandó képet, a másikkal karakterként végezheti el a beszúrást.
Ehhez már oldalakra lesz szükségünk. A párbeszédablak szerkesztőben, ha bármely vezérlő Tulajdonságok lapját megnézzük, láthatjuk, hogy van egy ***Oldal (fázis)*** tulajdonsága. Ez jelenleg az összes elemre 0. Ez azt jelenti, hogy az összes oldalon elérhető az adott elem. Amennyiben ez az érték 1, akkor csak az 1. oldalon, ha 2, akkor csak a 2-on, stb. látható. A léptetés megvalósításához gombjaink ezen tulajdonságát 0-án hagyjuk, hogy mindig látszódjanak, a már meglévő címkét és fájlválasztót 1-re állítjuk, és felveszünk a 2. oldalra néhány vezérlőt.
Először is tehát állítsuk 1-re a címke és a fájlkiválasztó ***Oldal (fázis)*** tulajdonságát. Úgy tűnik, ettől még nem változott semmi. Most állítsuk át az egész ablak ***Oldal (fázis)*** tulajdonságát 2-re. Az előbb említett vezérlők eltűnnek, mintha sosem lettek volna ott. Ezután tegyünk fel egy címkét és két választógombot. A választógombok nevei legyenek `Bekezdeshez` és `Karakterkent`. A `Bekezdeshez` választógomb ***Tulajdonságok*** lapján az ***Állapot*** mezőt állítsuk arra, hogy ***Kijelölve***.
Az előbb elkészített vezérlők ***Oldal (fázis)*** tulajdonsága önműködően 2 lett. Ez nekünk jó is, viszont az ablak oldal tulajdonságát vissza kell állítanunk 1-re, hiszen azt szeretnénk, hogy az ablak létrehozásakor ez legyen az, amit a felhasználó meglát. Most újra neki kell esnünk a kódolásnak, ez viszont már nem lesz nehéz. Szükség van az `Elozo` és `Kovetkezo` gombok eljárásaira, és ki kell egészítenünk a `ParbeszedAblakBefejezes` rutint a horgony kezeléséhez. Lássuk tehát a teljes kódot:

```
Private parbeszedAblak as Variant

Sub Main

   parbeszedAblak = createUnoDialog
   ➥(DialogLibraries.Standard.Dialog1)
   parbeszedAblak.execute ()

End Sub

Sub ParbeszedAblakElozo

   parbeszedAblak.Model.Step = 1
   parbeszedAblak.Model.Elozo.Enabled = false
   parbeszedAblak.Model.Kovetkezo.Enabled = true

End Sub

Sub ParbeszedAblakKovetkezo

   parbeszedAblak.Model.Step = 2
   parbeszedAblak.Model.Elozo.Enabled = true
   parbeszedAblak.Model.Kovetkezo.Enabled = false

End Sub

Sub ParbeszedAblakMegsem

   parbeszedAblak.endExecute ()

End Sub

Sub ParbeszedAblakBefejezes

   Dim kepURL as String, horgony as Long

   parbeszedAblak.endExecute ()

   ‘ Eleresi utat fogunk kapni, ezt at kell
   ➥alakitani
   kepURL = ConvertToURL
   ➥(parbeszedAblak.Model.FileControl1.Text)
```

```
' Ha Allapot = Kijelolve, akkor State = 1

if parbeszedAblak.Model.Karakterkent.State = 1
➥then
   horgony = com.sun.star.text.
   ➥TextContentAnchorType.AS_CHARACTER
elseif parbeszedAblak.Model.Bekezdeshez.State = 1
➥then
   horgony = com.sun.star.text.
   ➥TextContentAnchorType.AT_PARAGRAPH
endif

dokumentum = ThisComponent
szoveg = dokumentum.getText ()

grafika = dokumentum.createInstance
➥("com.sun.star.text.GraphicObject")
grafika.GraphicURL = kepURL
grafika.AnchorType = horgony

lathatoKurzor = dokumentum.getCurrentController
➥().getViewCursor ()
szovegKurzor = szoveg.createTextCursorByRange
➥(lathatoKurzor.getStart ())

szoveg.insertTextContent (szovegKurzor.getStart
➥(), grafika, false)

End Sub
```

Természetesen az `Elozo` és `Kovetkezo` gombokhoz megfelelő eljárásokat még a gombok eseménykezelőinek meg kell adni, de ezt a harci feladatot már az Olvasóra bízom. Sikerült tehát közösen létrehoznunk egy tündért, amely egy képet szúr be a dokumentumba, és még a horgonyt is meg tudjuk adni a segítségével. Készítettünk hozzá gyorsindítót is az eszköztárra, így ezt a szenzációs makrót, mely a már beépített funkcióként elérhető képbeszúrást végzi el hibátlanul, hihetetlenül gyorsan érhetjük el. Érdemes megismerni olyan eszközöket is, melyekről elképzelhető, hogy nem sűrűn fogjuk előnyben részesíteni őket más megoldásokkal szemben. Azért el kell ismernünk, igen gyorsan össze lehetett dobni ezt a kis tündért, és a szükséges fejlesztőkörnyezet már telepítve volt a gépre egy irodai csomag formájában. Nem ebben a környezetben fogjuk megírni a világegyenletet megoldó programot, de kisebb feladatokra, különösen a táblázatkezelés területén, használható eszközre találhatunk a ***Basicben***. A ***http://api.openoffice.org/DevelopersGuide/DevelopersGuide.html*** címről elérhető több, mint 1000 oldalas leírás bőséges ismertetővel szolgál a témával kapcsolatban. A fenti példa is innen származik. Akit érdekelnek az ***OpenOffice.org*** programozóknak nyújtott szolgáltatásai, mindenképpen töltse le a leírást. Akár ***Java*** nyelven is készíthetők ugyanis olyan alkalmazások, melyek az ***OpenOffice.org*** lehetőségeit aknázzák ki. Érdemes egy-két pillantást vetni erre is.
Sok örömet a programozáshoz!

Fülöp Balázs



PHP5 – Új generáció (3. rész)

...avagy hogyan használjuk okosan az osztályokat és objektumokat PHP5-ben.

Az előző részben az objektumszemléletű fejlesztés igazi erősségét adó örökléssel, mint nagy témakörrel foglalkoztunk. Szó volt az ezekkel szorosan kapcsolatban álló ***elvont (abstract) osztályokról*** illetve ***felületekről (interface)*** és a bennük rejlő lehetőségekről, végezetül pedig megnéztünk néhány mágikus tagfüggvényt a teljesség igénye nélkül.
Most megnézzük, mire jók azok a bizonyos kivételek, miként másolhatjuk az objektumainkat értékeik szerint, illetve hogyan írhatunk elő objektumtípusokat paraméter gyanánt. Ne is húzzuk az időt, kezdjünk bele!

Kivételkezelés

PHP 5-ben is bevezették a *Java*, *C#* nyelvekből már jól ismert kivételeket. Ezek olyan felhasználói „hibaüzenetek", amelyeket a programkódban használhatjuk hibajelzések, különleges események kezelésére. Működésének lényege, hogy bizonyos feltétel teljesülése esetén mondjuk egy objektumból kivételt dobunk, amelyet azonban az objektum metódusát hívó programkód fog majd elkapni, lekezelni. Ez így homályos lehet, ezért nézzünk egy példát:

```
<?php
class Negyzet {
    private $oldal = 0;

    public function __construct($oldal) {
        $this->oldalHossztBeallit($oldal);
    }

    public function oldalHossztBeallit($ertek) {
        if (is_numeric($ertek)) {
            $this->oldal=$ertek;
        } else {
            throw new Exception("Értékadási hiba:
            ➥a megadott érték nem szám");
        }
    }

    public function terulete() {
        echo $this->oldal*$this->oldal;
    }
}
$negyzet1=new Negyzet(0);
try {
    $negyzet1->oldalHossztBeallit("asd");
    $negyzet1->terulete();
} catch (Exception $ex) {
    echo $ex->getMessage();
}
?>
```

A kulcs az `oldalHossztBeallit()` metódusban van. Jelen esetben, ha olyan értéket adunk meg, ami nem szám, akkor szeretnénk, ha erről értesítene bennünket az objektum, hogy a felhasználás pillanatában tudjuk, mi a hiba. A főprogramban a `try` blokk szolgál arra, hogy megpróbáljuk beállítani az oldalhosszt. Ez természetesen nem biztos, hogy sikerülni fog. A `catch` blokkban kezeljük le azt, ha esetleg a `try` blokk valamelyik parancsa nem sikerült, s az adott típusú kivétel keletkezett. Jelen esetben megpróbálunk karaktersorozatot adni értékül az oldalnak, ami nem fog sikerülni. Ekkor a `try` blokkban az utasítások végrehajtása megszakad, a futás átugrik a megfelelő `catch` blokkba (ahol az adott típusú kivételre várunk), és végrehajtja az ott található összes utasítást.
Egy `try` blokkot tetszőleges számú `catch` blokk követ. Mint már említettem is, abba a blokkba kerül a vezérlés a hiba esetén, amilyen típusú az objektum által dobott kivétel (jelenleg `Exception` típusú). Ez akkor lehet igazán hasznos, ha mi magunk is saját kivételtípusokat készítünk, és más más típusra máshogyan szeretnénk reagálni.
Természetesen egy blokkon belül több azonos kivételt előidéző utasítás is lehet (az oldalhossz beállítása mellé valami), azonban minden esetben csak egyetlen kivétel jön létre, hiszen utána azonnal a `catch` blokkban találjuk magunkat.
A módszer előnye az, hogy `if`-ek nélkül egyszerűen kezelhetjük a hibákat, ráadásul a használt objektum belső szerkezetének ismerete nélkül. Tegyük fel, hogy a négyzetnek beállítjuk még a színét is. Ha nem megfelelő színt adunk, akkor ott is kiírathatunk egy hibaüzenetet, s ilyenekből tetszőleges számú lehet. Ekkor azonban a felhasználás során alkalmazott `try-catch` szerkezet a világon semmit sem változik, csak végzi az egyszerű hibakezelési feladatot.
Természetesen nem szükséges minden esetben lekezelni a kivételt... nem kell tehát `try` blokkba foglalni. Ilyenkor, ha mégis bekövetkezik, végzetes hibával leáll a program futá-

sa, ezért célszerű mindig alkalmazni. Olyan eseteben hagyhatjuk el mégis, mint a fenti példa. Ha ugyanis a programból mindig ugyanazt a konstans számértéket állítjuk be, sohasem fog kivétel képződni. A dolog csak akkor érdekes, ha mondjuk a felhasználó által beolvasott értéket szeretnénk megadni oldalhosszként, amelyet nem ismerhetünk előre.

Típuselőírás, típusmeghatározás

A fentiekben már volt szó a `catch` blokkok kapcsán arról, hogy a kivétel, mint osztály típusával azonosítható egy ilyen kódrészlet. Az előző változatokhoz képest szokatlan a (típus változónév) formula, amely itt arra hivatott, hogy meghatározza, milyen típusú legyen az adott paraméter. Mivel a *PHP* egy gyengén típusos nyelv, ez az előírás csak azt erőlteti, hogy a paraméter objektum legyen, mely a típusként megadott nevű osztály egyik példánya. Ez az úgynevezett ***type hinting*** minden függvényre és metódusra alkalmazható.

```
function test(Negyzet $negyzet) {
        $negyzet->terulete();
}
```

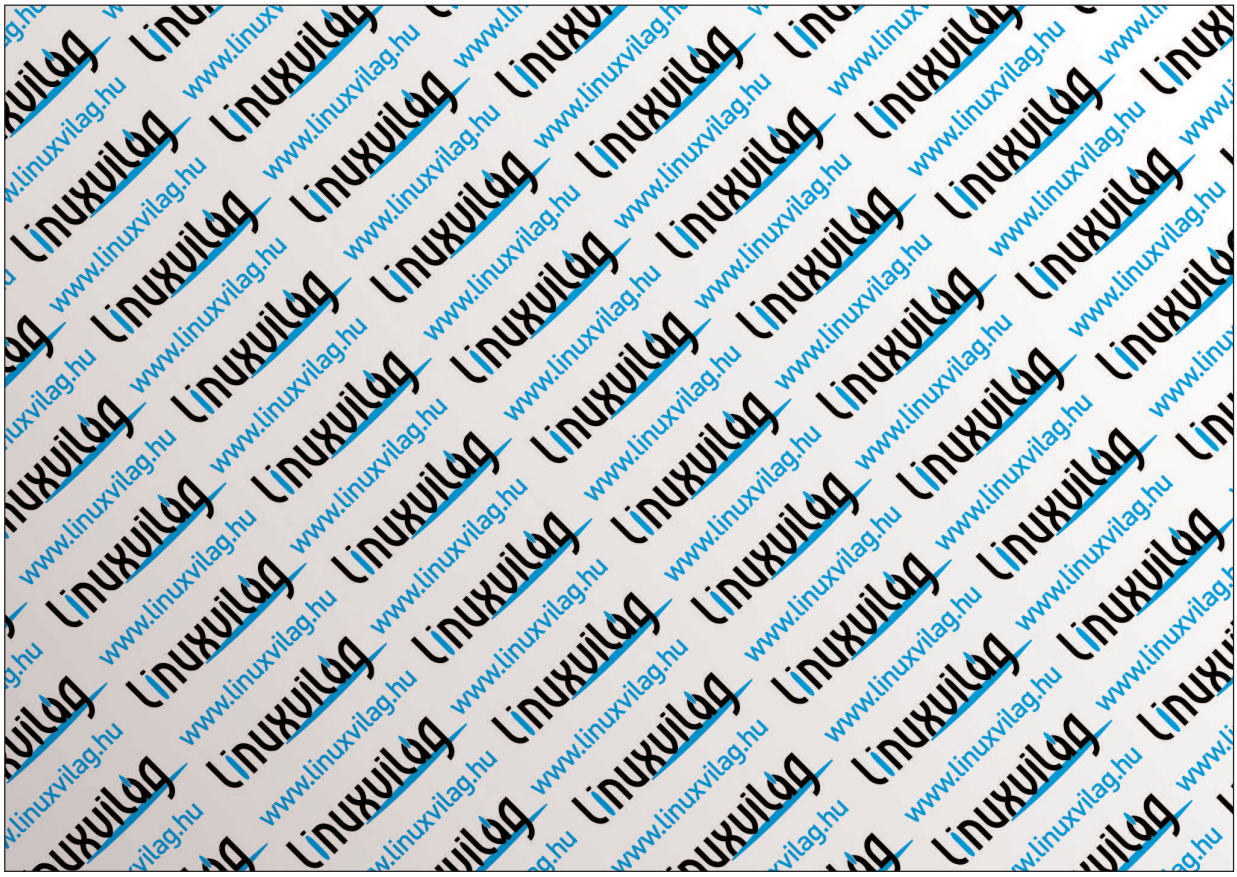
A fenti kódrészlet az előző példa kódját felhasználva szemlélteti az esetet. A test függvény csak a `Negyzet` osztály példányait fogadja paraméterül. Nem lehet más osztály példánya, egyáltalában csak osztályokat fogad paraméterül, és null sem lehet az értéke.
A fenti lehetőséget minden olyan esetben haszonnal alkalmazhatjuk, ahol előre meghatározott objektumpéldányokon kell műveletet végeznünk (így garantált, hogy létezik a paraméter megadott metódusa, stb.).

Objektumok másolása

Szó volt róla, hogy a *PHP 5* egyik legjelentősebb változtatása, hogy az objektumok átadása a régi érték szerinti megoldás helyett már cím szerinti. Ez azt jelenti, hogy `$a=$b` esetén mindkét változó ugyanarra a memóriaterületre fog mutatni (természetesen csak objektumok esetén), ezért ha az egyiknek bármilyen tulajdonsága megváltozik, valójában a másik is meg fog változni. Olyan ez, mintha két póráz lenne a kezünkben, amely ugyanazon kutyában végződik. Gyakran előfordulhat az a dolog, hogy mi valóban szeretnénk megduplázni úgy azt az objektumot, hogy valóban kettő legyen belőle, mert mindkettőn egymástól független műveleteket szeretnénk végezni, tehát kell nekünk két póráz, amely két különböző kutyában végződik. Még mindig a fenti négyzetes példánál maradva

```
$negyzet1 = new Negyzet(5);
$negyzet2 = clone $negyzet1;
$negyzet2->oldalHossztBeallit(3);

$negyzet1->terulete();
$negyzet2->terulete();
```

A fenti kódrészlet a futás után kiír egyszer 25-öt, majd 9-et, jól látszik tehát, hogy a két példány különböző. Ez olyan esetekben lehet hasznos, ahol van egy sok-sok tulajdonsá-



got tartalmazó objektumunk, be is vannak állítva ezek az adatok jól, és mi szeretnénk valamiért a rajta végzett műveletet „elágaztatni”, és az új objektumban csak bizonyos értékeket megváltoztatni.
Felvetődik itt azonban egy probléma, amelyet legjobb lesz, ha egy példán keresztül próbálok meg bemutatni:

```
<?php

class Ember {
    private $nev;
    private $szuletesiEv;

    public function __construct($nev,$szuletesiEv) {
        $this->setNev($nev);
        $this->setSzuletesiEv($szuletesiEv);
    }

    public function setSzuletesiEv($szuletesiEv) {
        $this->szuletesiEv=$szuletesiEv;
    }

    public function setNev($nev) {
        $this->nev=$nev;
    }

    public function getNev() {
        return $this->nev;
    }

    public function getSzuletesiEv() {
        return $this->szuletesiEv;
    }

}

class Anya extends Ember {
    private $gyermeke;

    public function setGyermeke(Ember $ember) {
        $this->gyermeke=$ember;
    }

    public function getGyermeke() {
        return $this->gyermeke;
    }
}

$ellie = new Anya(“Ellie”,1920);
$jockey = new Ember(“Jockey”,1950);
$ellie->setGyermeke($jockey);

$mary = clone $ellie;
$mary->setNev(“Mary”);
$ellie->getGyermeke()->setSzuletesiEv(1945);
echo $mary->getGyermeke()->getSzuletesiEv();
?>
```

A fenti kódrészlet az alábbi szituációt takarja: vannak `Emberek`, és vannak olyan speciális emberek, akik egyben `Anyák`, s van gyerekük, aki egy másik `Ember`. A példában létrehoztuk `Ellie`-t, és `Jockey`-t, mint embereket, majd azt mondtuk, hogy `Ellie` mama `Jockey` bácsi anyukája. Ez eddig rendben is volna.
Ezek után lemásoltuk `Ellie` anyukát `Mary` néven, majd `Ellie` fiának születési évét átállítottuk. Sajnálattal tapasztaltuk azonban az eredményt, hogy `Mary` fiának születési éve is megváltozott, amiből hajlamosak vagyunk arra következtetni, hogy a gyermekeik azonosak, pedig mi nem ezt szerettük volna.
A jelenség magyarázata az alábbi: az anyák gyermekükre referenciaként hivatkoznak, hisz látszik a kódból (`setGyermeke()`), hogy egyszerű egyenlőség operátorral „másoltuk” őket. Ez nem baj, ez az eredeti szándékunk, hisz így szép és helyes a megoldás. A baj abból adódik, hogy a `clone` hívás alapértelmezetten ún. ***sekély másolatot (shallow copy)*** készít, mindent egy az egyben átmásol az új objektumba, tehát referenciát is referenciába másol, így a két objektum $gyermeke tulajdonsága ugyanarra a példányra mutat. A kutyáknál maradva: hiába van nekünk két külön kutyánk a póráz végén, ha hozzájuk van kötözve egy harmadik közös kutya.
Nem mindig hasznos tehát, ha egy objektum egy az egyben másolódik (néha működésből adódóan sem célszerű – például nincs mindenre szükségünk). Erre megoldás lehet az, ha egy osztályban rögzítjük előre, hogy egy esetleges másolás esetén milyen értékek hogyan menjenek át a másolatba. Erre szolgál a `__clone()` nevű mágikus tagfüggvény, amely a `clone` utasítás kiadásakor hívódik meg az adott objektumra. Bármilyen utasítás, amit itt rögzítünk, végrehajtódik. A fenti példa javítása tehát: adjuk hozzá az Anya osztályhoz az alábbi metódust:

```
public function __clone() {
    $this->gyermeke = clone($this->gyermeke);
}
```

A hivatkozott referenciára is készítünk valódi másolatot. Fontos megjegyezni, hogy nem kell minden attribútumot átmásolnunk, azok a `__clone()` metódustól függetlenül automatikusan átmásolódnak, nekünk csak felül kell írnunk a nem kívánt értékeket.
További érdekesség, hogy ha megengedjük a `setGyermeke()` metódusban, hogy ne csak Ember, hanem Anya is lehessen gyerek (minek tulajdonsága referenciát tartalmaz), a fenti másolóalgoritmus akkor is hibátlanul működik, hisz a `__clone()` metódusban kiadott `clone` utasítás hatására a gyermekre (aki anya is lehet) szintén meghívódik annak `__clone()` metódusa, és ez így gyűrűzik egészen addig, amíg van hivatkozásunk a hivatkozott objektumban.
Már csak egy nagy témakör maradt hátra, nevezetesen a ***PHP 5*** új ***Reflection API***-ja, amely az osztályok, objektumok igen széles körű és részletes elemzésére szolgál – akár futásidőben is. A sorozat következő részében ezen a területen teszünk egy kisebb kirándulást.

Komáromi Zoltán

Számítógép hálózatok (14. rész)

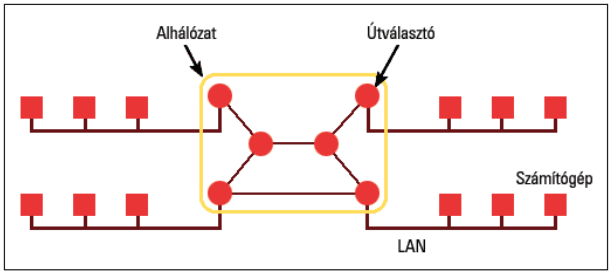
Virtuális áramkörök és datagram alapú alhálózatok működése, forgalomirányítás a hálózati rétegben.

Bevezetés

Az előző részben már elkezdtünk foglalkozni a hálózati réteggel, amelynek segítsége nélkül a hosztok képtelenek lennének megtalálni egymást a hálózaton. Arról is volt szó, hogy alapvetően kétféle elképzelés létezik arról, milyen szolgálatokkal is rendelkezzen a hálózati rétegnek. Az első elgondolás szerint a hálózati rétegnek összeköttetés alapú szolgálatot kell megvalósítani, amely nagyon hasonló egy telefonon történő kommunikációhoz. Valamelyik gép felhívja a másikat, kiépül a kapcsolat, majd megtörténik az adatcsere. Amikor mindkét gép befejezte a társalgást, a kapcsolatot lebontják. Ilyen elv szerint működnek például az ***ATM*** hálózatok.
Az Internet és protokolljai azonban egy másik szemléletet követnek: itt a szolgálat mindig összeköttetés nélküli.
Az ilyen típusú kommunikációt úgy képzelhetjük el, mintha levelezve, postai úton kommunikálnánk valakivel. Az elküldendő adatot „becsomagoljuk" egy úgynevezett datagramba, ráírjuk a címzettet és a feladót, majd útjára engedjük.
Amikor a hálózati réteg megvalósításáról beszélünk, akkor tulajdonképpen a kommunikációs alhálózat (vagy röviden alhálózat) működését vesszük szemügyre. Emlékezzünk vissza: az alhálózat az a hálózat, amely a gépeket, illetve helyi hálózatokat egymással összeköti. Ez a hálózat többnyire ***útvonalválasztókból (routerekből)*** áll (*1. ábra*).

Összeköttetés alapú hálózat

Az ilyen jellegű kommunikáció pontosan úgy működik, mint amikor telefonon beszélgetünk valakivel. Sőt, tulajdonképpen maga a telefonhálózat is összeköttetés alapú hálózat. Amikor két gép adatot kíván cserélni egymással, először létre kell hozniuk egy kapcsolatot. Amíg ez a kapcsolat él, addig tudnak egymásnak csomagokat küldözgetni. Az ilyen hálózatok megbízhatóak, tehát a csomagok nem vesznek el útközben, nem sérülnek meg és sorrendhelyesen érkeznek meg.
A gépek között kiépülő kapcsolatot ***virtuális áramkörnek (virtual circuit)*** nevezzük. Mivel általában több olyan alhálózati útvonal is létezik, amelyen keresztül eljuthatunk az egyik géptől a másikig, ezért a virtuális áramkörnek nem csak a két gépet kell meghatároznia, hanem egy hozzájuk tartozó útvonalat is. Ez az útvonal az összeköttetés felépítésekor kerül kiválasztásra, tehát minden elküldött csomag ugyanazon az útvonalon fog haladni.
Ehhez azonban a routereknek tudniuk kell, hogy a rajtuk átmenő csomagokat melyik kimenetükön keresztül küldjék



1. ábra

tovább. Így minden útválasztó rendelkezik egy táblázattal, amely hozzárendeli a rajta „keresztülmenő" virtuális áramköröket a megfelelő kimenettel. Amikor beérkezik egy csomag, akkor az útválasztó a fejlécből meg tudja állapítani, hogy az melyik virtuális áramkörön halad, így táblázata segítségével a megfelelő kimenetre képes irányítani.
A helyzetet az bonyolítja kicsit, hogy a virtuális áramköröknek nem lehet egy globálisan érvényes azonosítószámod adni, mivel minden gép maga választja ki azokat. Így a virtuális áramkörszámok csak helyi szinten érvényesek. Ha ez nem így lenne, akkor könnyen előfordulhatna az az eset, amikor két virtuális áramkör ugyanazt az azonosítót kapja, és ráadásul ugyanazon az útválasztón is megy keresztül. Ebben a helyzetben a router nem tudná eldönteni, hogy a beérkező csomagot melyik kimenetre is irányítsa.
Ha a vonal duplex, akkor időnként bekövetkezhet egy elég kellemetlen esemény, ami akkor fordul elő, ha két gép egy időben kíván kapcsolatot kiépíteni a másikkal. Ilyenkor a kapcsolat felépítésére irányuló kérelem mindkét irányban egy időben kezd el haladni az alhálózaton. Tegyük fel, hogy a routerek programja olyan, hogy új virtuális áramkör felépítésekor mindig a legkisebb szabad sorszámot rendelik az új kapcsolathoz. Amikor a két kérelem két szomszédos routerhez érkezik, akkor mind a ketten ugyanazt az azonosítót fogják kiosztani, tehát két különböző virtuális áramkörnek ugyanaz lesz a száma ugyanazon a fizikai vonalon. Ez azért baj, mert a routerek nem fogják tudni a beérkező csomagokról, hogy ez az egyik összeköttetésben egy előrehaladó, vagy a másikon egy visszafelé menő-e.
Amikor a két gép közötti kommunikáció befejeződik, akkor a virtuális áramkör lebontásra kerül.

Összeköttetés nélküli hálózatok

Míg az összeköttetés alapú hálózatoknál a hálózati réteg felel az összeköttetés megvalósításáért, addig az összeköttetés nélküli hálózatoknál ez egy felsőbb réteg (a szállítási réteg) feladata. Az Internet és annak protokolljai esetében a hálózati rétegre csak a neki átadott csomag célbajuttatása van bízva. Ehhez nem kell előzetesen kiépített kapcsolat, nem kell törődni a forgalomszabályozással és az sem biztos, hogy az átvitel hibátlan lesz. (Míg ezeket a feladatokat az összeköttetésen alapuló hálózatoknál az alhálózat végezte, addig itt ez a gépek feladata).
Mivel nincs összeköttetés, ezért az útvonal nincs előre kijelölve, hanem minden csomag esetén külön számítódik ki. Ezért a csomagoknak tartalmazniuk kell a forrás- és a célállomás címét. A csomagok egymástól függetlenül kerülnek továbbításra, ezért a fejlécben fel kell tüntetni a csomag sorszámát is, hiszen könnyen megeshet, hogy a célhoz nem az útrabocsátás sorrendjében érkeznek be a csomagok. (Mivel valószínűleg mindegyik más útvonalon keresztül ért célt). Számolni kell azzal a lehetőséggel is, hogy egyes csomagok soha sem érkeznek meg, vagy ami rosszabb, a vevő kétszer is megkapja ugyanazt a csomagot. Az összeköttetés mentes hálózatok tehát korántsem nevezhetők megbízhatónak. Az egymástól függetlenül továbbított adatdarabokat datagramoknak nevezzük. Még egyszer hangsúlyozzuk, hogy a datagramoknak soha nincs előre kijelölt útvonaluk. Egy ilyen hálózat kezelése mindenképpen bonyolultabb és munkaigényesebb feladat, viszont ezért cserébe sokkal jobban alkalmazkodik a véletlenszerűen jelentkező forgalomváltozásokra.
Az útválasztóknak az összeköttetés nélküli hálózatokban is szükségük van belső táblázatokra, csak ott most nem a virtuális áramköröket tartják nyílván (mivel ilyesmiről itt nem beszélhetünk), hanem a velük kapcsolatban lévő routerek címeit. A router a beérkező csomagot a célcíme alapján irányítja a megfelelő kimenetre. (Annak eldöntése, hogy melyik kimenetre kerüljön a csomag, egyáltalán nem egyszerű. Erről később bővebben szólunk majd a forgalomirányítási algoritmusok tárgyalásakor).

A két megoldás összehasonlítása

Nagyon nehéz kérdés, hogy melyik megoldás jobb a másiknál. Az biztos, hogy általánosan nem mondható el egyikről sem, hogy az jobb vagy rosszabb lenne. Sőt, azt sem egyszerű eldönteni, hogy egy konkrét helyzetben melyik alhálózati megvalósítás felelne meg leginkább az igényeknek. Most megpróbáljuk összeszedni mindkét hálózattípus előnyeit és hátrányait, illetve mikor jelenthet az egyik hatékonyabb megoldást a másiknál. De az itt leírtakat sem szabad örökérvényű törvénynek felfogni, mivel szélsőséges esetekben lehet, pont a másik típus lenne a legjobb választás.
Rögtön az első dolog, ami különbséget jelenthet a virtuális áramkörök és a datagramok között az a csomagok mérete. Az első esetben csak a virtuális áramkör azonosítószámát kell a csomagoknak szállítaniuk, amely lényegesen kisebb méretű mint a forrás- és a célcím. Ha kis csomagokkal dolgozunk, akkor ez a méretbeli eltérés jelentős lehet. Ugyanakkor az is igaz, hogy az összeköttetés alapú alhálózatoknál a routereknek nagyobb belső memóriára van szükség, hacsak nem akarjuk drasztikusan lecsökkenteni a virtuális áramkörök maximális számát.
A hálózat sebességét nézve is kompromisszumokra kell kényszerülnünk. Míg a virtuális áramkörök esetében az összeköttetés felépítése, illetve lebontása időt vesz igénybe, addig a datagram hálózatokban ez a késleltetés nem jelentkezik. Viszont a routernek sokkal tovább tart eldönteniük, hogy a csomagot melyik kimenetre továbbítsák.
Ha az összeköttetés várhatóan fennáll hetekig, hónapokig, esetleg évekig, akkor sokkal jobb választás a virtuális áramkör. Ha viszont csak időnként szeretnénk kis adatmennyiséget küldeni, tehát a kommunikáció úgymond tranzakció jellegű, akkor a datagram a hatékonyabb megoldás.
A torlódások elkerülését tekintve az összeköttetés alapú hálózatok némi előnnyel rendelkeznek. A szükséges erőforrásokat ***(értsd: routereket)*** már előre, a kapcsolat felépítésekor le lehet foglalni, így azok mindig rendelkezésre fognak állni, amikor csak szükség lesz rájuk. Az is igaz viszont, hogy a datagram alapú hálózatok sokkal dinamikusabban tudják kezelni a torlódásokat. Ha egy router nagyon leterhelt, akkor egy másikon keresztül mennek tovább a csomagok. Egy adott csomag útvonala tehát mindig az aktuális terhelési viszonyoktól függ.
A virtuális áramkörök ugyanakkor nagyon könnyen megszakadhatnak, nem kell hozzá más, mint hogy egy útválasztó akár csak egy másodpercre is üzemképtelenné váljon. Hiába indul újra egyből, a kapcsolat elveszik, és azt újra fel kell építeni. A datagram alapú hálózatoknál ez nem lehet probléma. Egy router kiesése csak azoknak a gépeknek jár kellemetlenséggel, akiknek a csomagjai éppen a kérdéses router memóriájában tartózkodtak.

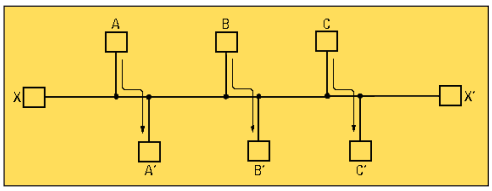
Összeköttetés megvalósítása datagramokkal, és összeköttetésmentes szolgálat virtuális áramkörökkel

Fontosnak tartjuk, hogy hangsúlyt helyezzünk a szolgálat típusa (összeköttetés alapú, illetve összeköttetés nélküli) és az alhálózat típusa (virtuális áramkörös vagy datagramos) között. A szolgálat típusa a felső réteg számára határozza meg, hogy pontosan mit is várhat el az adott rétegtől.
Az alhálózat típusa már a megvalósítás kérdéskörébe tartozik. Bizonyos esetekben szükség lehet egy datagramon alapuló összeköttetéses szolgálatra, vagy egy összeköttetés nélküli virtuális áramkörös megoldásra. Az utóbbira a legjobb példa az, ha valaki egy ***ATM*** hálózaton szeretne IP protokollt használni.

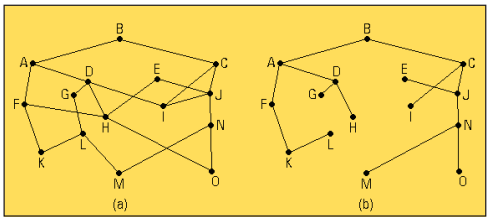
Forgalomirányító algoritmusok

A hálózati réteg megvalósításának legfontosabb kérdése az, hogy milyen útvonalon jutassa célba az alhálózaton keresztül menő csomagokat. A LAN-ok világában egyedül a switch-ek végeztek forgalomirányítás-szerű tevékenységet, de azt is csak a hálózaton lévő forgalom csökkentésére. Az adatszórásos hálózatoknál nincs szükség forgalomirányításra. Ha azonban a forrás és a cél különböző hálózaton van, akkor találni kell egy útvonalat, amelyen a csomag eljuthat rendeltetési helyére. Az is fontos, hogy ez az útvonal a lehető legoptimálisabb legyen.
Az útvonalat a ***forgalomirányító algoritmus*** határozza meg, amely része a hálózati szoftvernek. Talán nem meglepő, hogy a forgalomirányító algoritmus ***forgalomirányítást*** végez, amely nem jelent mást, mint eldönteni, hogy egy beérkező csomag melyik kimeneten folytassa útját. A virtuális áramkörökön alapuló hálózatoknál ezt csak az új kapcsolatok felépü-

lésénél kell elvégezni. A datagramos hálózatoknál minden beérkező csomagnál döntést kell hozni.
Sokféle forgalomirányítási módszer létezik, vannak rendkívül jól működők és vannak kevésbé hatékonyak. Mielőtt ezek működésébe részletesen is belemélyednénk, foglaljuk össze, milyen elvárásaink lehetnek egy ilyen algoritmussal szemben. Ezek pedig a következők: helyesség, egyszerűség, robusztusság, stabilitás, optimalitás és igazságosság.
Az nagyon jó dolog, ha az algoritmusunk helyesen működik, és nem hoz hibás döntéseket. Az sem árt, ha egyszerű, nem tartalmaz bonyolult számításokat, így kevesebb erőforrás szükségeltetik a megoldáshoz. Robusztusság alatt azt értjük, hogy az algoritmus ne legyen érzékeny az alhálózat topológiájában történő változásokra. Egy hálózat életében ugyanis naponta történhetnek váratlan fordulatok: áramszünet vagy valami hiba következtében egy útválasztó kiesik, majd hirtelen ismét megjavul, stb. Nehéz lenne az élet egy olyan hálózatban, amit mindig újra kéne indítani, ahányszor egy router meghibásodik.
Az igazságosság és az optimalitás alapvető követelménynek tűnhetnek, ám e két cél egymással ellentétes. Nyílván nem mindig a legigazságosabb megoldás a legoptimálisabb. Elképzelhető, hogy globálisan hatékonyabb eredményt érünk el, ha időnként egy-egy gép számára nem a legkedvezőbb megoldást nyújtjuk. A *2. ábrán* láthatunk egy példát, ami konfliktushoz vezet az igazságosság és az optimalitás között. Tegyük fel, hogy a A és A′, B és B′, C és C′ között olyan nagy a forgalom, hogy telíti az őket összekötő, az ábrán vízszintes helyzetben lévő vonalakat. Az optimális megoldás az lenne, hogy minél több gép tudjon egymással zavartalanul kommunikálni, tehát el kéne zárnunk az X és X′ közötti vonalat. Az X hoszt előtt ülő felhasználó viszont ezt nem tartaná igazságos elbánásnak.
A forgalomirányító algoritmusokat két nagy csoportra oszthatjuk. Az elsőbe az úgynevezett ***nem adaptív algoritmusok*** tartoznak. Ezek az egyszerűbb eljárások, amelyek a döntések meghozatalánál nem vesznek figyelembe mérési, becslési eredményeket, sőt még az aktuális forgalmi helyzettel sem törődnek. Inkább minden számítást előre, „offline" módon végeznek el, amelynek eredményeit a routerek a hálózat indításakor kapnak meg. Történjen tehát bármi, két pont között minden csomag ugyanazt az utat fogja bejárni. Ez az úgynevezett ***statikus forgalomirányítás*** módszere.
A második csoportba az ***adaptív algoritmusok*** tartoznak, amelyek felfigyelnek a topológiában, illetve a forgalmi viszonyokban történő változásokra, és ellentétben az előzőekkel, döntéseiket ezen változások figyelembevételével hozzák meg. Ezt nevezzük ***dinamikus forgalomirányításnak***.
Az adatív algoritmusok sok mindenben különbözhetnek egymástól. Egyesek mindig csak a szomszédos routerektől veszik az információkat, mások az alhálózatban lévő összes routerével kapcsolatban állnak. Különbözhetnek még abban is, hogy milyen módon próbálják az optimalitást megvalósítani. Például inkább a csomagkésleltetést kívánja-e csökkenteni, mint hogy a teljes alhálózat átbocsátóképességét a lehető legnagyobbra állítani. (A későbbiekben erre még részletesen is visszatérünk. Most csak azt jegyeznénk meg, hogy e két cél egymásnak ellentmond, így az algoritmusoknak megint valami kompromisszumos megoldást kell találnia).



2. ábra



3. ábra

Az optimalitási elv

Azt már tudjuk, hogy a forgalomirányító algoritmusoknak törekedniük kell arra, hogy a hálózat optimálisan működjön. De vajon mit jelent az, hogy optimális? Erre az úgynevezett ***optimalitási elv*** ad választ. Ennek a fogalom a bevezetése talán jobban megvilágítja a forgalomirányító algoritmusok működését. Most még nem foglalkozunk az olyan gyakorlati jellegű dolgokkal, mint például a hálózaton jelenlévő forgalom, vagy az alhálózat topológiája.
Az optimalitási elv a következő vallja: ha a B jelű router az A és a C router közötti optimális útvonalon helyezkedik el, akkor a B-től C-ig vezető optimális útvonal is része annak. (Ez egy nagyon egyszerűen bizonyítható tétel, ugyanis indirekt módon feltehetjük, hogy létezik egy jobb útvonal B és C között. Ekkor ezt összekapcsolhatjuk az A-ból B-be vezető útvonallal, így egy jobb A-C utat kapunk. Ez azonban ellentmondás, ugyanis az AC út az optimális). Ennek a tételnek egy érdekes következménye az, hogy az összes lehetséges forrástól egy adott célhoz tartó utak fát alkotnak, amelynek gyökere maga a cél. (A fa olyan gráf, amelyben nincs kör). Ezt a fát ***nyelőfának (sink tree)*** nevezzük. (*3. ábra*).
Észrevehető, hogy a nyelőfa nem feltétlenül egyedi, azaz nem csak egy optimális útvonal létezhet például az A router és a többi router között. A forgalomirányító algoritmusok feladata valójában pont az, hogy ezekre az utakra rábukkanjon.
A fák tulajdonságából következik az is, hogy minden csomag véges (korlátos számú) ugráson belül eléri célját. (Ugrás alatt most egy routeren történő keresztülhaladást értjük). Ez azonban csak a mi példánkban van így, a gyakorlatban minden bizonnyal nem áll fent. A routerek ugyanis elromolhatnak és hirtelen megjavulhatnak, így nem biztos, hogy például minden router ugyanúgy „képzeli el" a hálózat aktuális topológiáját. Gyakorlati kérdés az is, hogy az egyes routerek miként tudják beszerezni azokat az információkat, amellyel a nyelőfát felépíthetik. Erről a következő részben szólunk részletesen, amikor a gyakorlatban is használt forgalomirányító algoritmusokkal ismerkedünk. Ugyan az optimalitási elvnek és a nyelőfának közvetlen gyakorlati haszna nincs, mégis egyfajta viszonyítási alapként szolgálnak a való élet forgalomirányító algoritmusainak összevetése során.

Garzó András
garzo@interware.hu

Főzzünk Linuxszal: Elfelejtett biztonság

Jobb ha nem használjuk ugyanazt a jelszót több azonosítóhoz. Tekintve, hogy mennyi jelszóigényes kiszolgáló és weblap létezik, mit tehet ilyenkor egy biztonság-tudatos főszakács? Nézzük meg, hogyan tehetjük kényelmessé és biztonságossá a jelszó pásztorkodást.

Hol van az a borrendelés ***Henri Különleges Boraitól***, ***François***? Úgy tűnik kezdünk kifogyni a kedvenceimből. ***Henri*** általában otthon van ezen a téren. Nem adott neked visszaigazolást? Ah, kitűnő. Akkor megvan a rendelés? Nem? Hogy érted, hogy valahol biztonságban van? Most akkor meg van, vagy nincs? Értem. Úgy gondoltad, hogy fontos, ezért titkosítottad a rendelést és eldobtad az eredeti üzenetet. Had találjam ki, ***mon ami***, nem emlékszel a jelszóra amivel titkosítottad az üzenetet. Ahogy gondoltam. Rendben, mutasd milyen programot használtál.

Steganográfia, François? Saját arcképedbe rejtetted a borrendelést? Le vagyok nyűgözve! Ezzel a problémával egy kicsit később foglalkozunk, ***François***. Nincs túl sok időnk, vendégeink bármelyik pillanatban itt lehetnek. Ah, de hát már itt is vannak.

Üdvözlet, ***mes amis Chez Marcelnél***, a világ legfinomabb francia ***Linux*** éttermében és a világ legnagyszerűbb borospincéjében. Amely sajnos jelenleg lehet, hogy csak a második legjobb a világon. Úgy tűnik hűséges pincérem elkavarta a rendelést és nem akarta elmondani nekem. Igen, ***François***, tudom, hogy tudod hol van. Menj a pincébe és hozd fel a portugáliai 2000-es ***Douro***-t. Kiváló vörös, ***mes amis***, testes és erős bor, gyönyörű sötét gyümölcsillattal és egy csepp misztikummal. ***Vite***, ***François***!

Míg ***François*** felhozza a bort, elmondom hogyan próbálta meg biztos helyre tenni a borrendelést. Mint kiderült, ***Stefan Hetzl Steghide*** nevű programját használta, hogy a listát saját fényképébe rejtse (*1. ábra*).

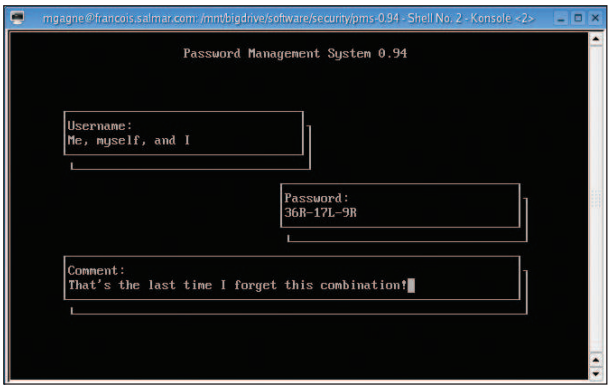
Ezt a módszert szteganográfiának nevezik. A módszerrel tetszőleges üzenetet rejthetünk egy másik üzenetbe (vagy mint ebben az esetben egy grafikus képbe). Tulajdonképpen, egy titkos üzenetekkel megtűzdelt képekkel teli weblapot is létrehozhatunk, és senki nem fog sejteni semmit. A ***Steghide***-ot a ***Steghide*** honlapjáról gyűjthetjük be (lásd a hálózati forrásokat). A kiadott futtatható állományokat könnyen meg fogjuk találni. Ha a ***Steghide***-ot forrásból szeretnénk fordítani, szükségünk lesz a ***libmhash***, ***libjpeg***, ***zlib*** és ***libmcrypt*** fejlesztői könyvtárakra. Ezek után már könnyű dolgunk van, a szokásos öt lépéses kitömörítés és fordítás menettel találkozunk:

1. ábra Ebbe a képbe rejtve valahol egy jókora borrendelést találunk

```
tar -xzvf steghide-0.5.1.tar.gz
cd steghide-0.5.1
./configure
make
su -c “make install”
```

A borrendelés elrejtése esetében ***François*** a következő parancsot használta a dokumentum képbe ágyazásához:

```
steghide embed -cf francois.jpg -ef wine_order.txt
```



2. ábra PMS nem csak jelszavakhoz használható. Akár a szekrénykódunkat is tárolhatjuk benne.

Ha már borról beszélünk, ***François*** visszatért. Légy oly kedves, ***mon ami***, és tölts a vendégeinknek. Nos, a parancs futtatása után egy jelszót kell megadnunk:

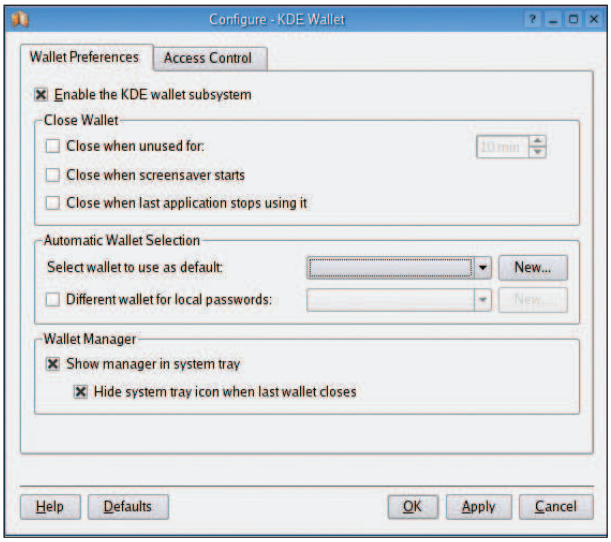
```
Enter passphrase:
Re-Enter passphrase:
embedding “wine_order.txt” in “francois.jpg”...
➥done
```

Eredményképpen a titkos üzenet elrejtését megelőző változattal azonos kinézetű képet kapunk, de a mérete megváltozik. Ha az adatokat ki akarjuk nyerni a képből, nekünk (vagy akinek a képet elküldtük) csak az `extract` paramétert kell megadnunk a következő paranccsal:

```
steghide extract -sf francois.jpg
Enter passphrase:
```

Amennyiben sikeresen megadtuk a helyes adatokat, a rejtett állomány a lemezre kerül. Nos, éppen ez az a pont ahol a dolgok elkezdenek rosszra fordulni. Miután elfelejtettük a jelszót, nincs többé módszer az információ kinyerésére.
A valós életben néhányan közülünk alkalmanként elveszítjük a kulcsunkat. Mások rendszeresen elvesztik, aminek következményeként egy vállalati feltaláló kijött a kulcscsomóra szerelhető csipogóval. Feltételezve, hogy a keresőt már nem veszítjük el, csak megnyomjuk a gombot és a kulcs magas hangon csipogva jelzi melyik díszpárna mögé csúszott be éppen.
A jelszavak esetében hasonló az alapötlet. A legegyszerűbb módszer leírni a jelszavakat valahová vagy egy szöveges állományba menteni őket. Ez azonban nem különösebben biztonságos. Ugyanakkor a jelszó és jelszöveg lista készítése egyre nagyobb értelmet nyer, ahogy jelszavak tucatjait, néha százait kell megjegyeznünk. Mennyivel egyszerűbb lenne, ha csak egyetlen jelszót kéne megjegyeznünk!
Itt kerülnek a képbe a jelszókezelők.
Az első ilyen amibe belefutottam ***Dennis Pries Password Management System (azaz PMS)*** rendszere volt. Ez azért tetszett, mert tisztán szöveges terminálablakban is futtatni lehetett, következésképpen bárhol legyünk is, egy héjbejelentkezésből használni tudjuk. A programot a ***SourceForge*** honlapján találhatjuk meg (lásd a forrásokat), ahol a forrást és a ***Debian*** csomagot egyaránt letölthetjük.

A ***PMS*** fordításához dupla kitömörítés és az öt lépéses fordítás szükséges. Először is csomagoljuk ki a tarolt és gzippelt állományt (`tar -xzvf pms-0.94.tar.gz`). Belekukkantva a forráskönyvtárba egy contrib alkönyvtárat láthatunk, ahol a forrásfájlból a kicsomagolás és fordítás szokásos öt lépésével elkészíthetjük a ***cdk***-t. A ***cdk*** telepítése után lépjünk vissza a ***PMS*** forráskönyvtárába, fordítsuk le és telepítsük.
A jelszókezelőt a `pms` paranccsal indíthatjuk. Első indításkor meg kell adnunk a mesterjelszót. Ez az egyetlen jelszó vagy jelszöveg, amit mostantól meg kell jegyeznünk, ezt viszont alaposan. Ha elfelejtjük a mesterkulcsot soha nem látjuk viszont a többi jelszavunkat. A ***PMS*** egyszerű menüjével gépneveket vehetünk fel, törölhetünk és nevezhetünk át. Ezek lesznek azok a helyek, ahová be kell jelentkeznünk. Először is vegyünk fel egy gépnevet (például, ***www.somewhere.dom***) majd fűzzünk megjegyzést hozzá (például: központi rendszer). Ezután ismét a főmenüben találjuk magunkat. Itt válasszuk a ***User Functions (Felhasználói Műveletek)*** pontot. Ebben a menüpontban vehetjük fel vagy törölhetjük az előző lépésben megadott géphez tartozó felhasználói azonosítókat. Egyúttal itt jeleníthetjük meg a felhasználóhoz tartozó elveszettnek hitt jelszavainkat is.
Mielőtt továbblépnénk, szeretnék rámutatni, hogy a gépnév és a felhasználói név akármi lehet. Gépnévként beírhatom azt is, hogy „iskolai zár”, felhasználónévnek, hogy „kombináció” jelszónak pedig magát a kombinációt. Bár eredetileg bejelentkezési információk őrzésére tervezték, a program más célra is jól használható (*2. ábra*).
A másik dolog amit mindig el szoktunk felejteni, a rengeteg meglátogatott weblaphoz rendelt jelszavak. Az on-line banki rendszerektől kezdve a hírújságokig ahol ingyenes azonosítóra van szükségünk a cikkek olvasásához, idővel irgalmatlan sok jelszó gyűlik össze. Aztán ott vannak az üzenetküldőkhöz és az e-mail bejegyzésekhez tartozó jelszavaink, az ***FTP*** helyek és még hosszan sorolhatnánk. Jelentősen leegyszerűsítené a dolgunkat, ha mindezt az információt valahogy átlátszóan tudnánk kezelni munka közben.
Van esetleg valamilyen eszköz ami beépül az asztalba?
A válasz természetesen igen. A ***KDE 3.2***, és a legújabb 3.3 verziójában a felhasználók beépített a jelszókezelővel rendelkeznek. ***George Staikos KDE Wallet Manager*** rendszeréről van szó, amely a `kwalletmanager` programot futtatja.
A program első indításakor nem készül tárca (wallet). Ugyanakkor a rendszertálcán egy apró tárca ikont találunk. Amennyiben a tárcakezelő ikonja még nincs nyitva, kattintsunk az ikonra, mire megnyílik egy (leginkább üres könyvtárra emlékeztető) üres doboz. Kattintsunk a ***Beállítások (Settings)*** gombra a menüben és válasszuk a ***Tárcabeállítások (Configure Wallet)*** pontot.
Új űrlapablak jelenik meg, ahol a legtöbb elem nem elérhető. Kattintsunk a ***KDE tárca alrendszer engedélyezése (Enable the KDE Wallet Subsystem)*** pontra. Most már néhány további lehetőséget is elérhetünk (*3. ábra*).
Figyeljük meg az ***Automatikus Tárcaválasztás (Automatic Wallet Selection)*** nevű középső részt. Meg kell adnunk melyik tárcát szeretnénk alapértelmezettként használni. Rögtön ez alatt kiválaszthatjuk a helyi jelszavakhoz használt tárcát (erről hamarosan még bővebben szó lesz).
Ha először futtatjuk a ***KDE Tárcát***, nem valószínű, hogy



3. ábra A KDE Tárcakezelő beállítása jelszótároláshoz

van létező tárcánk; Kattintsunk az ***Új (New)*** gombra és adjunk nevet a tárcánknak. Nyugodtan adhatjuk a saját nevünket ahogy én is tettem. Miután begépeltük a nevet és ráböktünk az ***OK***-ra, a megjelenő ***KDE tárcakezelő varázsló (KDE Wallet Manager Wizard)*** alap vagy kifinomult beállítási lehetőségei közül választhatunk, ahol az alap a javasolt mód. A kifinomult változatban kicsit több információs képernyőt találunk és a helyi jelszavainkhoz külön tárcát választhatunk. Én az alapot választottam és csak egy tárcát készítettem.

Bármelyiket is választottuk a varázsló egy idő után rákérdez a tárca megnyitásához szükséges mesterjelszóra.

Ez a szuperfelhasználó jelszó lesz az amit nem szeretnénk elfelejteni- az, amely az összes többi ajtaját nyitja. Jól válasszunk, és ne felejtsük el az ***„Igen, személyes adataimat is szeretném a KDE tárcában tartani" („Yes, I wish to use the KDE wallet to store my personal information")*** négyzetet megjelölni.

Ha a varázslóval végeztünk, majdnem készen is vagyunk. A felbukkanó üzenetablak tudatja velünk, hogy az alkalmazás (a varázsló) szeretné létrehozni az új tárcát. A kérelmet a tárcához tartozó jelszóval kell jóváhagynunk. Figyeljük meg ezt az űrlapot. Ehhez hasonlót fogunk látni minden ***KDE*** folyamatban amikor valamelyik alkalmazás jelszót keresve meg szeretné nyitni a tárcát. Amíg ki nem jelentkezünk, a tárca nyitva marad. Lépjünk be egy weblapra ahol jelszót és felhasználónevet kérnek tőlünk (például megnyithatjuk a banki elérésünket). Miután kitöltöttük az információkat és ráböktünk az ***Elküld*** vagy ***Enter*** gombra (űrlaptól függően), a ***KDE*** tárcakezelő ablaka jelenik meg figyelmeztetve, hogy egy alkalmazás (jelen esetben a ***Konqueror***) megpróbálta megnyitni az alapértelmezett tárcát (amit éppen most készítettünk). A *4. ábrán* láthatjuk mire gondoltam. Írjuk be a mesterjelszót és bökjünk a ***Folytatásra (Continue)***. Egy utolsó figyelmeztetést kapunk, miszerint a titkosított adatok elmentésre kerülnek, és ezt jóvá kell hagynunk. Kattintsunk az Igen gombra. Most vessünk egy pillantást a tálcára és látni fogjuk, hogy az ikon zárt tárcáról kissé nyitott tárcára változott.

4. ábra A tárca megnyitásához meg kell adnunk a mesterjelszót

E rendszer szépsége, hogy az adatok automatikusan beíródnak nekünk amikor legközelebb meglátogatjuk a lapot. Ez minden ***KDE*** alkalmazásra igaz, amely jelszót kér tőlünk, tehát például az üzenetküldőre is.

Egyetlen buktató van, mégpedig nem is kicsi. Mint említettem, csak egyetlen egyszer kell begépelnünk a mesterjelszót minden ***KDE*** folyamathoz, ami leegyszerűsíti a dolgokat. De vigyázat: most, hogy a rendszert képessé tettük a jelszavaink automatikus megadására, az asztalunk biztosítása igencsak fontossá válik. Ne feledjük el lezárni az asztalt mielőtt elmegyünk. Másik megoldás, hogy visszalépünk a ***KDE*** tárca beállítások űrlapjára és megismerkedünk a ***Tárcazárás (Close Wallet)*** lehetőségeivel. Beállíthatjuk, hogy egy megadott idő után automatikusan záruljon be, például, amikor a képernyő pihentető elindul (azaz amikor általában nem vagyunk ott) vagy amikor a tárcát használó utolsó alkalmazás is becsukódik. Ha így járunk el, egyel kevesebb dologra kell emlékeznünk.

A faliórára pillantva, ***mes amis***, úgy látszik ismét utolért bennünket a záróra. Mint láthattuk több lehetőségünk is van jelszavaink tárolására így nem kell tucatnyi titokzatos betű és számkombinációt megjegyeznünk. Talán, ha sikerül meggyőzni ***François***-t, hogy ilyen eszközöket használjon a jövőben, nem lesz több eltűnt rendelés. Addig is, abban biztos vagyok, hogy sikerül meggyőzni, hogy még egyszer töltse teli vendégeink poharát. És a bortartalékok miatt ne aggódjon senki. Személyesen gondoskodom róla, hogy a borospince színültig legyen mire legközelebb találkozunk. Addig is, ***mes amis***, egészségünkre. A ***votre santé! Bon appétit!***

Linux Journal 2005. január, 129. szám

A cikkhez tartozó források:
➲ ***www.linuxjournal.com/article/7860***

Marcel Gagné (mggagne@salmar.com) Mississaugaban él Ontario államban. Ő a szerzője a Moving to the Linux Business Desktop (ISBN 0-131-42192-1) című könyvnek amely a harmadik műve az Addison-Wesley gondozásában. A valós életben a rendszerintegrációval és hálózati tanácsadással foglalkozó Salmar Consulting, Inc. elnöke. Egyúttal pilóta, sci-fi és fantasy novellákat ír és silány origami T-Rex figurákat hajtogat.

Munka- és szolgálati viszony I. – Általános rész

A szoftverekkel kapcsolatos, már részleteiben érintett probléma a munkaviszonyban alkotott szoftverek esete.

Itt felmerülnek olyan kérdések, hogy mi minősül egyáltalán munkaviszonyban alkotott műnek? Mennyire tág a munkaszerződés hatásköre, milyen – a megítélést befolyásoló – szerepe lehet a program megalkotásához szükséges infrastruktúra rendelkezésre bocsátásának, van-e eltérés a kötött munkaidejű programozók alkotásai között illetve azok esetében, akik csak meghatározott óraszámot kötelesek ledolgozni, ám tetszőleges időpontokban (saját választása szerint akár hétvégén, vagy éjszakákba nyúlóan)? A szerzői jogi törvények általában véve munkaviszonyban alkotottnak minősítik azokat a műveket (így a szoftvereket is), melyek elkészítője és később vagyoni jogi jogosultja közötti, a munkavégzést rendező viszonyt a Munka törvénykönyve, a Közalkalmazotti illetve a Köztisztviselői törvény rögzíti. Ám az ezekhez pusztán hasonlító szerződési formák esetében nem állapítható meg munkaviszony léte, s így a művekre vonatkozó vagyoni jogosultságok automatikus átszállása sem. Ennek oka a szerzői jogi törvény szerzővédő értelmezési elve lehet.

Érdekes kérdést vet fel, ha a szoftver egy érvénytelen munkaszerződés keretei közt jön létre, itt a körülmények elbírálása lesz a döntő arra vonatkozóan, hogy a felhasználási jogok, legalább a jogügylet licenc szerződésnek való minősítéséből kiindulva átszálltak-e.

Ha kétség merül fel, a munkáltató köteles bizonyítani, hogy a program megalkotása munkaviszonyból fakadó kötelezettség volt, amennyiben erre nem képes, a szerző szabadon rendelkezhet művével.

Külön kérdéskört jelent az igénybe vett hardver esete. Míg a nyolcvanas évek elején elképzelhetetlennek tűnt a szoftveralkotás otthoni körülmények között, és feltétlenül nagy beruházást igényeltek az olyan számítógépek, melyek egy komolyabb program futtatására alkalmasak voltak, mára már lényegesen egyszerűbbé vált a megfelelő hardverháttér biztosítása, ám ugyanakkor előfordulhat, hogy a munkavállaló a munkájához kapcsolódó, ám nem munkaköri feladatai közé tartozó módon, munkahelyén kívül, saját hardveren a munkáltatója számára is hasznos szoftvert fejleszt. Amennyiben ez a szoftver összefügg a munkahelyén megismert know-how ismeretekkel, kötelessége lesz a munkáltató számára felkínálni a művet, míg az otthonában alkotott szoftverek esetében alapvetően az a feltételezés él, hogy a munkaviszonyon kívül alkotott a mű. Ugyanakkor a szoftverfejlesztés tipikusan az a terület, ahol a „távmunka” minden gond nélkül megvalósítható. Sajátos megoldást jelentene, ha a szabadalmi alkotások mintájára, elhatárolást nyerne a „szolgálati” illetve az „alkalmazotti találmányhoz” hasonlóan a „szolgálati” illetve „alkalmazotti mű” megkülönböztetése a szerzői jogban. Ezen felosztás szerint a munkaköri kötelességként alkotott szoftver esetében a munkáltató a vagyoni jogok megszerzésére, míg munkaviszonyból eredő kötelezettségen kívül fejlesztett szoftverekre, amennyiben annak hasznosítása a munkáltató tevékenységi körébe tartozik, csak felhasználási jogot nyerne a munkáltató.

Ugyancsak megfontolandó szoftverkészítők esetében a munkaidő fogalma, hiszen a hely meghatározásán túl az idő is fontos tényező. A munkaidőn túli teljesítményért az általános szabályok szerint túlóradíj illetné meg, amennyiben azonban ezt nem utalják számára, feltételezhető-e, hogy a műért cserében megfelelő ellenszolgáltatást kapott? Fontos figyelembe venni azt, hogy a munkáltató nem sajátíthatja ki a programozót, így annak is joga van – még abban az esetben is, ha ő az egyetlen, az adott témához értő szakember egy cégnél – a szabadsághoz, és ahhoz, hogy a szabad idejét saját elképzelései szerint töltse el.

A legszigorúbb elhatárolást mégis a munkaköri leírás jelenti, hiszen általában ez alapján ítélik meg a szoftver elkészítésénél releváns körülményeket. Ugyancsak munkaviszonyban alkotottnak minősül a mű, ha az elkészítésére a szerző utasítást kap, ám ilyenkor kérdéses, hogy az utasítás mennyiben terjeszkedhet túl az adott a munkakörön, hiszen például egy éjjeli őr is lehet jó programozó, de ez még nem jelenti, hogy a munkáltató őt egy szoftver elkészítésére utasíthatja. Ehhez kapcsolódó a német bíróság egy 1997-ből származó döntése, ahol a bíróság kimondta, hogy a technika fejlődéséből következik, hogy egyszer elérünk majd egy pontot, mikor egy felelősségteljes munkakört betöltő személytől elvárható lesz nem csak a programok futtatásának, hanem fejlesztésének képessége is, és attól kezdve a szoftveralkotás a munkaszerződések részét fogja képezni.

Ám már ezen állapotot megelőzően is megállapítható, hogy amennyiben a munkáltató jóváhagyásával és annak költségére a munkavállaló szoftvert fejleszt, az a munkaszerződési tevékenységi kör keretein belülinek minősül.

Az egyes jogokra speciálisan hat, amennyiben munkaviszonnyal összefüggően gyakorolják őket, bár nagy részüket már említettük, érdemesek még egy összefoglaló áttekintésre. Erre azonban a következő hónapban térünk ki részletesen.

Dr. Dudás Ágnes