Könnyű álmok (12. rész)

Az előző számban megjelent cikkünkben megkezdtük az ismerkedést az Interneten használt legfontosabb protokollokkal. Most ezt az utat járjuk tovább.

Először tekintsük át a levelezésnél használt legfontosabb protokollokat! A levelek küldésére és fogadására az `SMTP` (Simple Mail Transfer Protocol) szolgál. Az Interneten elhelyezkedő kiszolgálók e protokoll segítségével fogadják és továbbítják a leveleket. Nem minden számítógép rendelkezik azonban megfelelő erőforrásokkal (lemezterület, folytonos internetkapcsolat, állandó IP-cím) ahhoz, hogy kihasználhassa ezt a lehetőséget. Ezért két általánosan elterjedt protokoll is létezik, amelyek kisegítő szerepet játszanak. Az egyik a POP3 (Post Office Protocol – Version 3), amely a levelek letöltését teszi lehetővé, a másik pedig az IMAP4 (Internet Message Access Protocol – Version 4), ez még szélesebb körű szolgáltatást nyújt.

Mielőtt a protokollok ismertetésébe belekezdenénk, tekintsük át, hogy a levelezés milyen veszélyeket is jelent a hálózatunkra nézve. A legismertebb veszélyről már mindenki hallott: ez a vírusok terjedése. A vírusok a felhasználók képzetlenségét vagy kényelemszeretetét kihasználva terjednek, és károkat okoznak a számítógépeken. Sok elterjedt levelezőprogramban sajnos folyamatosan súlyos biztonsági hibákat találnak, amelyeket a támadók vagy a féregprogramok (worm) kihasználhatnak. Ilyenek lehetnek például az érvénytelen MIME-csatolások vagy a túl hosszú fejlécek hibás kezelése. Ezen hibák ellen a csomagszűrés-alapú megoldások nem nyújtanak védelmet.

A cikkben szereplő példákat ismét képzeletbeli tűzfalunkra írtuk, ahol az `eth0` csatoló a védett, míg az `eth1` csatoló az Internet felé néz. Az elozó cikkhez hasonlóan itt is csak az INPUT láncon mutatjuk be a beállításokat. Feltételezzük, hogy 2.2.x rendszermag esetén a nem SYN-es csomagokat a csomagszűrő szabályaink elfogadják. 2.4.x rendszermagsorozat esetén a rendszernek az ESTABLISHED állapotú csomagokat el kell fogadnia.

SMTP

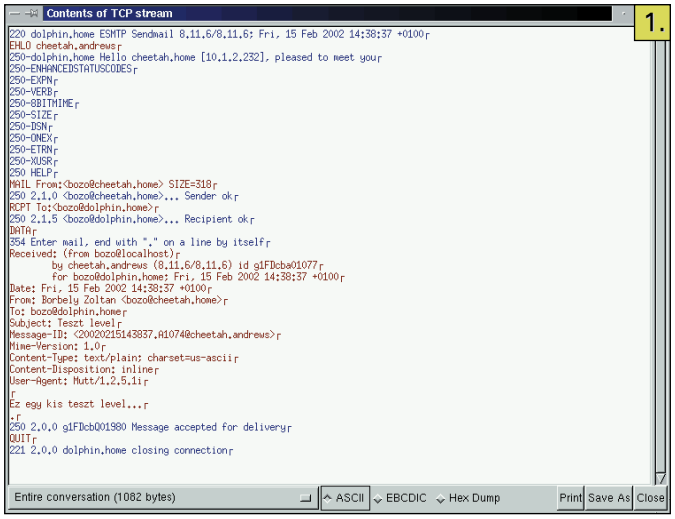
Az SMTP-protokoll egyike az Internet legrégibb protokolljainak, feladata a levelek továbbítása és fogadása. Az SMTP szövegalapú protokoll, ami TCP-t használ. A kiszolgáló bejegyzett kaput használ, ami a `25/tcp`. Az SMTP-protokoll `store & forward` (tárol és továbbít) alapú. Ez azt jelenti, hogy amelyik SMTP-kiszolgáló veszi a levelet, az vagy a felhasználó postaládájába helyezi, vagy tárolja és a címzett felé továbbítja. Amikor a kézbesítés sikertelen (például a címzett nem létezik, vagy a beállított időkorláton belül sem sikerült a levelet továbbítania), akkor a hibáról egy levelet küld a feladónak, majd a kézbesíthetetlen levelet eldobja. A hibalevél feladója egy különleges cím, a <> (azaz a cím üres). Sajnos, néhány vírus ellen védekezésnek „köszönhetően” sokan anélkül letiltották az ilyen levelek fogadását, hogy a következményekbe belegondoltak volna. Így nemcsak a vírus által küldött leveleket dobják el, hanem az MTA-k által küldött hibaleveleket is. A `store & forward` működés miatt az egyes SMTP-kiszolgálók (szaknyelven gyakran MTA-knak – Message Transfer Agentnek hívják őket) natív proxyként képesek működni. Amikor egy MTA levelet kíván küldeni, a DNS-kiszolgálótól megkérdezi az adott tartomány levelezőkiszolgálóinak címeit. Ezek a címek vagy MX [1.], vagy A rekordokban vannak tárolva. A címek közül választ egyet (egy tartomány általában több levelezőkiszolgálóval is rendelkezik, így az elsődleges kiszolgáló elérhetetlensége esetén a levelek nem a feladó SMTP-kiszolgálóján várakoznak), majd TCP-kapcsolatot kezdeményez a megadott levelezőkiszolgáló 25-ös kapujával, ahol egy MTA fogadja. Az elküldendő leveleket továbbítja, majd zárja a kapcsolatot. Erre láthatunk példát az *1. képen*.

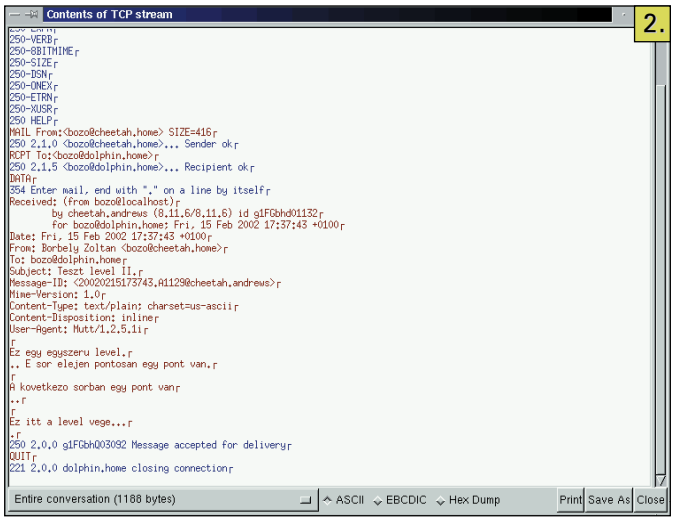
A kép az `etheral` program [2.] segítségével készült, és az ügyfél (***cheetah***) és a kiszolgáló (***dolphin***) között zajló TCP-forgalmat mutatja. Az ügyfélgép üzenetei pirossal, a kiszolgáló üzenetei kék színnel jelöltek. A protokoll menete a következő: az ügyfél TCP-kapcsolatot nyit a kiszolgáló 25-ös kapujára, ahol a kiszolgáló egy úgynevezett „greeting” üzenettel várja. Válaszul az ügyfél azonosítja magát, ami az EHLO vagy HELO üzenettel történhet, erre a kiszolgáló válaszol. Az EHLO üzenet esetén a kiszolgáló közli az általa támogatott protokollkiegészítéseket. E szakasz után következik a levelek tényleges elküldése. A levél küldését a `MAIL FROM` parancs jelzi (az SMTP-protokoll a kisbetűket és a nagybetűket a parancsokban nem különbözteti meg). A `MAIL FROM` parancs paramétere a `reverse path`, mely a küldő levélcíme. Mint látható, az SMTP MTA a küldőt elfogadta, ezt jelzi a 250 kódú válaszüzenet. Ezt követi a címzett(ek) megadása, jelenleg csak egy címzettet adtunk meg. A címzettek az `RCPT TO` parancs(ok) paramétereiben helyezkednek el. A ***dolphin*** gép elfogadta a címzettet. Miután a címzetteket megadtuk, a levél tartalmának átvitele következik. A levelet a `DATA` parancs vezeti be, és egészen addig tart, amíg egy sorban csak egy pont (`"."`) karakter áll. Mint láthatjuk, a levél első része a fejléc, ezt követi a levéltörzs. A fejlécet a törzstől egy üres sor választja el.

A levél lezárása után látható, hogy a kiszolgáló a levelet elfogadta. Mivel a ***cheetah*** gépnek most nincs több elküldendő levele, a `QUIT` paranccsal a kapcsolat bontását kezdeményezi. Ezután a TCP-kapcsolat lebomlik, a levéllel kapcsolatos további teendők (például a felhasználó postaládájába helyezés vagy a továbbküldés) már a ***dolphin*** gép feladata.

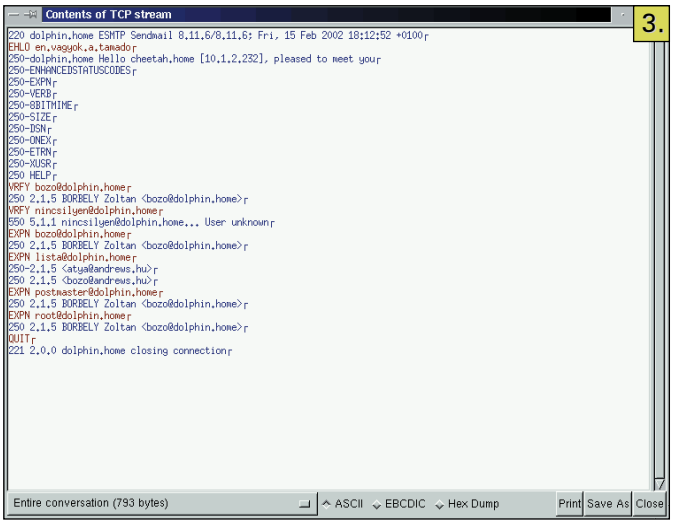
A `DATA` parancs után említettük, hogy a levél végét egy olyan sor jelzi, ami csak egy pontot tartalmaz. Mi történik akkor, ha olyan levelet írunk, ahol ez szintén előfordul? Erre láthatunk példát a *2. képen*. Itt jól látható, miként oldja meg ezt a gondot az SMTP-protokoll. A küldő program – amennyiben a sor elején pontot talál – még egy pontot szúr be elé. Így a sor elején álló pont a karaktertovábbítás folyamán mindig megkétszereződik. A vevő-MTA a sor elején álló pont felismerése után megvizsgálja, hogy sorvége következik-e (ez a levél végét jelentené), vagy más karakter található. Ha nem sorvége jelet talál, a plusz pontot eltávolítja. Ennek köszönhető, hogy

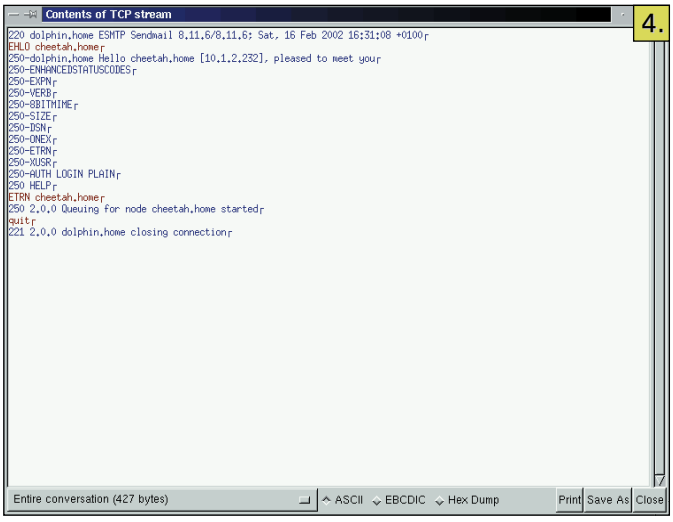
a címzett pontosan ugyanabban a formában kapja meg a levelet, ahogy a feladó azt megírta.
Most nézzük meg, hogy az általános levelezési kockázatok mellett milyen SMTP-re jellemző veszélyek leselkedhetnek még ránk. Az SMTP MTA-k egyedi parancsokkal bírnak ahhoz, hogy egy postaláda meglétét ellenőrizzék, vagy egy alias (nem valódi postaláda – a neki címzett levelet más címzett(ek) kapják





meg) tagjait felderítsék. Ezek hasznosak lehetnek hibakereséshez, azonban a támadók vagy a levélszemétcím-gyűjtők (akik az Internetről gyűjtik a levelezési címeket, amelyeket a szemetelők számára árulnak is) ugyancsak kihasználhatják. A postaláda létezésének ellenőrzése a támadók számára lehetővé teszi, hogy a rendszeren létező felhasználókat feltérképezzék. Ennek oka, hogy a felhasználói név általában megegyezik a postaláda nevével. Az alias kifejtése segítségével a támadó kiderítheti, hogy ki üzemelteti a számítógépet. Elég csak megnézni, ki kapja a `root` vagy `postmaster` leveleit (alapszabály, hogy a `root`-felhasználó nem levelezik, hiszen egy levelezőügyfél hibája végzetes következményekkel járhat a rendszer biztonságára nézve). A `VRFY` és `EXPN` parancsok kihasználására mutat példát a *3. kép*. Itt a `telnet` parancs segítségével kapcsolódtunk a ***dolphin*** levelezőkiszolgálójára, majd kézzel adtuk ki a parancsokat. Ezeket a parancsokat a kiszolgálóban célszerű kikapcsolni. Minden korszerű MTA képes e parancsok letiltására. Mint a protokollból is látható, az elektronikus levelek hamisítása roppant egyszerű feladat. Soha ne bízzunk meg egy hagyományos elektronikus levélben! A hamisítás ellen a legegyszerűbb és legjobb megoldás a levelek digitális aláírása. A legtöbb linuxos levelezőprogram (MUA – Mail User Agent) képes együttműködni valamilyen fejlett titkosító programmal (például gpg, pgp stb.). Napjainkban egyre nagyobb gondot jelentenek a szemetelők (spammer), akik kéretlen reklámjaikkal folyamatosan bombáznak bennünket. Tökéletes megoldás sajnos nem létezik ellenük, most csak arra vonatkozóan adunk néhány tippet, miként akadályozhatjuk meg, hogy a reklámokat a gépünkön keresztül küldjék. Az Interneten az SMTP MTA-k régi időkben bárkitől bárki számára elfogadtak levelet. Ezt a szemetelők alaposan ki is használták. Manapság ez a beállítás veszélyessé vált, hiszen a szemetelők felfedezik és reklámleveleiket az áldozat kiszolgálóján át továbbítják, ami jelentős hálózati terhelést és tekintélyveszteséget okozhat.





Az ilyen MTA-kat a szaknyelv `open relay`-nek hívja. A levéltovábbítást csak a védett hálózat számára célszerű engedélyezni, illetve azonosításhoz kell kötni.
Az Internet elterjedésével megjelentek olyan kisebb cégek, akik SMTP-alapú levéltovábbítást (számukra a POP3 vagy az IMAP4 nem jelentett megoldást) igényeltek, de nem rendelkeztek állandó internetkapcsolattal. Az ilyen szükséglet kielégítésére fejlesztették ki az ETRN-protokollbővítést, ami a `TURN` parancs (az ügyfél és a kiszolgáló viszonyának megfordítása) korszerűbb és biztonságosabb változata. Amennyiben a levél hagyományosan nem továbbítható, az SMTP MTA várakozási sorá-

ban marad, és a kiszolgáló meghatározott időközönként ismételt továbbítást kísérel meg. Az ideiglenes kapcsolattal rendelkező gépek számára (például telefonos kapcsolat) a továbbítási kísérletek közötti több óra túl hosszú, a próbálkozások idejének csökkentése pedig a kiszolgálóknak jelent felesleges terhelést. Az `ETRN` parancs célja, hogy az ügyfélgép Internetre csatlakozásakor a megadott kiszolgálókhoz fordul, és kéri őket, hogy kezdeményezzék a nekik szánt levelek azonnali továbbítását. Erre láthatunk példát a *4. képen*. Ahogy a képen is látszik, az ügyfél (***cheetah***) arra kéri a kiszolgálót (***dolphin***), hogy a ***cheetah.home*** tartományra vonatkozó leveleket küldje el számára. A levelek továbbítása a hagyományos SMTP-protokollon, egy független TCP-kapcsolaton át történik. Az `ETRN` használatához általában a DNS-be bejegyzett név szükséges, így csak az állandó IP-című ügyfelek használhatják.

A korszerű SMTP MTA-k egyre fejlettebb módszerekkel rendelkeznek: lehetővé teszik az azonosítást, valamint a köztük levő csatorna titkosítását is. Az azonosítás a kapcsolódó MTA vagy felhasználó kilétének megállapítására szolgál, célja pusztán a levél továbbíthatóságának igazolása. Ez azonban semmilyen védelmet nem jelent a levelek hamisításával szemben.

A levelek illetéktelen elolvasása ellen a csatorna titkosítása nem tökéletes megoldás, hiszen csak az adott csatornára vonatkozik. Amennyiben a levél végcélja nem ez az MTA, a levél a következő továbbítási lépésnél akár kódolatlanul is közeledhet.

A levél a köztes MTA-kon és a postaládában eredeti formájában található meg, tehát az így nyújtott védelem nem azonos magának a levélnek a kódolásával. A csatorna titkosításának elsődleges célja, hogy az azonosítási adatok ne kódolatlanul kerüljenek a hálózatra.

Az SMTP-levelezést nem célszerű pusztán csomagszűrés segítségével védeni. A legjobb módszer, ha a DMZ-ben (a DMZ – a DeMilitarized Zone határvédelmi fogalom: olyan alhálózatot hívunk így, amely sem védett hálózatunknak, sem az Internetnek nem része, lásd még Linuxvilág, 5. szám 40. oldal.) elhelyezzük a levelek továbbítását végző kiszolgálót, amely nem tartalmaz helyi kézbesítést (azaz nincsenek rajta postaládák). Az ilyen kiszolgáló célja, hogy az Internetről beérkező leveleket a cég belső levél kiszolgálóinak ossza szét, és a belülről jövő leveleket az Internetre továbbítsa. Ezen a kiszolgálón fontos a levelek fejlécét és tartalmát is vizsgálni, így a támadások egy része (például túl hosszú a levélfejléc vagy vírusos a levél) kiszűrhető. Hasznos, ha az SMTP MTA elkülönítőben (jail) foglal helyet. Az elkülönítők (jail) célja, hogy az esetlegesen bejutott támadót megakadályozza a kiszolgáló egyéb alrendszereinek támadásában. Ilyen környezet kialakításáról sorozatunk későbbi részeiben fogunk szót ejteni.

Kis cégek esetén nem mindig engedhető meg egy csak erre a célra fenntartott gép használata. Ilyenkor az SMTP-továbbítási feladatot a tűzfalra is bízhatjuk, de a beállításoknál fokozott gonddal kell eljárnunk.

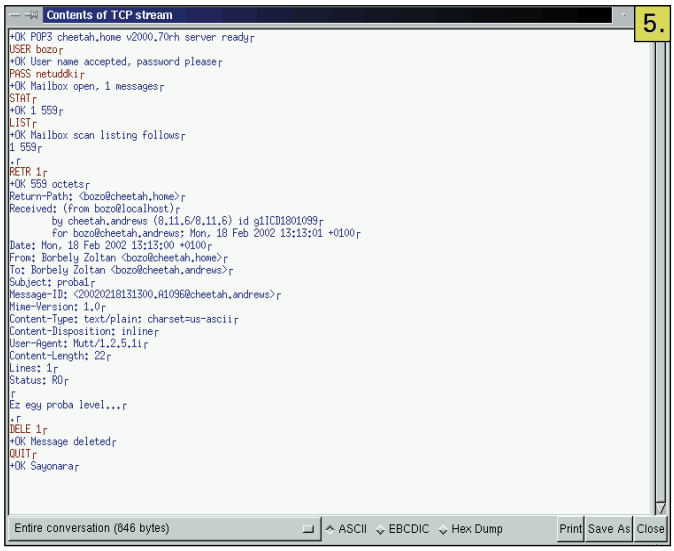
A helyi postafiókokba történő kézbesítés és egyéb felhasználói szolgáltatások (például *~/.forward* jellegű állomány, ahol a levél célja megváltoztatható vagy program indítható) szinte semmilyen esetben sem célszerűek. A jogokat a levéltovábbításhoz az elengedhetetlen minimumra kell csökkenteni, az MTA-t pedig elkülönítőbe (jail) kell zárni.

Ha feltételezzük, hogy a tűzfalunkon fut az SMTP MTA, a csomagszűrőben engedélyezni kell a `25/tcp` kapu elérését. Ezt a védett hálózat számára elérhetővé kell tenni, hogy a tűzfalon át SMTP-vel levelet kaphasson.

Amennyiben a külvilágból a leveleket SMTP-protokollon át kapjuk (állandó IP-címmel rendelkezünk), akkor az onnan érkező SMTP-kapcsolatokat is engedélyezni kell. Ha a külvilágból POP3- vagy IMAP4-protokoll segítségével töltjük le leveleinket, szerencsés, ha a kiszolgálónk SMTP-kapuja kívülről nem érhető el. Amennyiben a gép levélfogadást és -továbbítást nem végez (csak leveleket küld), tanácsos, hogy az adott gépen ne fusson MTA. A levelek küldésekor a legtöbb MUA a */usr/lib/**sendmail*** programot elindítva továbbítja a levelet. Az MTA-nak nem feltétlenül a ***Sendmailnek*** kell lennie – kompatibilitási okokból szinte bármelyik MTA tartalmaz programot ezen a néven. A név egyszerűen „történelmi” okokból alakult így.

Fontos figyelmet fordítanunk egy apróságra, amit sokan elfelednek: a levelet fogadó MTA-k sok esetben visszakapcsolódnak a küldő `auth`-kapujára (más néven `ident`, ez a `113/tcp` kapu), amely a kapcsolatot létesítő felhasználó kilétének megállapítására szolgál. A protokoll nagyon régi és egyáltalán nem biztonságos. Nem biztos, hogy egy lehetséges támadóra tartozna, hogy egy hálózati kapcsolatot kezdeményező program milyen felhasználóként fut. Ha azonban az ilyen kapcsolatokat a `DENY`-szabály segítségével eldobjuk, a levél továbbítása előtt a címzett MTA hosszan várakozhat. Ezért célszerű, hogy az `auth`-kapura érkező kéréseket utasítsuk el (lehetőleg `TCP RST`-csomaggal).

Ezt kétféle módon tehetjük meg: vagy a csomagszűrőből utasítjuk el (ez a biztosabb), de az RST-elutasítás használatára



csak a 2.4.x rendszermagsorozat esetén nyílik lehetőségünk, vagy meggyőződünk arról, hogy gépünkön nem fut az `auth`-szolgáltatás, és az `auth`-kapura beérkező kapcsolatokat elfogadjuk. A második megoldás azért lehet veszélyesebb, mert a szolgáltatás a későbbiek során „véletlenül” (például valamelyik későbbi csomag telepítésfüggő csomagként felhozza) települhet és elindulhat.

Amennyiben nem ragaszkodunk a TCP RST-elutasításhoz (így a kapcsolódó gép láthatja, hogy a TCP kaput csomagszűrő védi), úgy használhatjuk az ipchains REJECT akcióját is. Az alábbi példabeállítással csak a védett hálóról fogadunk el leveleket.

2.2.x rendszermagsorozatnál:

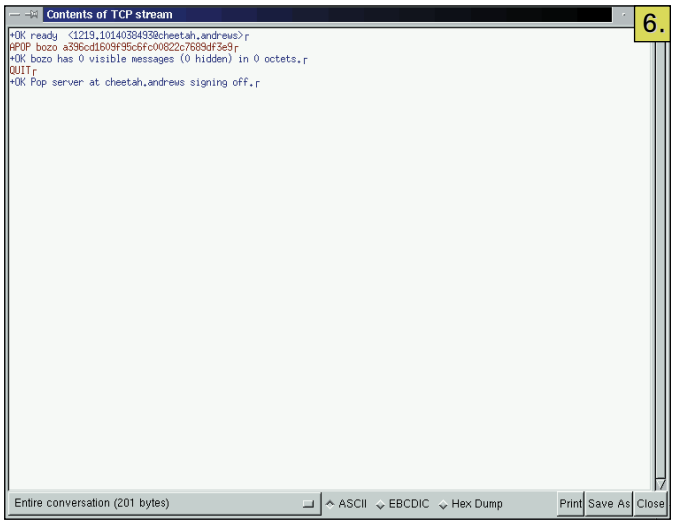
```
ipchains -A input -i ! eth0 -d 0.0.0.0/0
➥smtp -p tcp -j DENY
ipchains -A input -i eth0 -d
➥tuzfal_belso_ip_cime smtp -p tcp -j ACCEPT
ipchains -A input -d 0.0.0.0/0 auth -p tcp
➥-j REJECT
```

2.4.x rendszermagsorozatnál:

```
iptables -A INPUT -p tcp -i ! eth0 --dport
➥smtp -j DROP
iptables -A INPUT -p tcp -i eth0 -d
➥tuzfal_belso_ip_cime --dport smtp -j ACCEPT
iptables -A INPUT -p tcp --dport auth
➥-j REJECT --reject-with tcp-reset
```

POP3

A POP3-protokoll lényege, hogy egyszerű ügyfélgépek (például asztali számítógépek) számára is lehetővé tegye a levelezést (a POP3-protokoll nem támogatja levelek küldését).
A levél ilyenkor SMTP-protokollon keresztül egy kiszolgálóra érkezik, ahol a felhasználó postaládájába kerül. A POP3-protokoll célja, hogy a felhasználó postaládájában levő leveleket letöltse, majd letörölje onnan. A protokoll által nyújtott szolgáltatások köre igen behatárolt, a fő cél az egyszerű megvalósíthatóság volt. Aki szélesebb körű támogatást szeretne



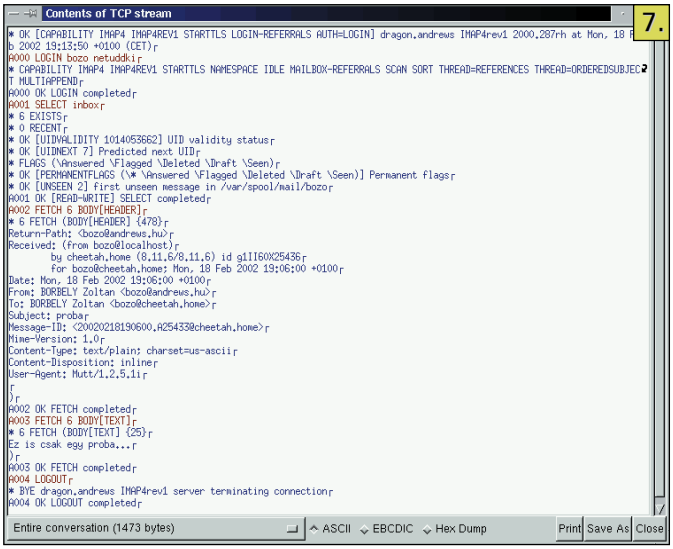
6.

(például mappák kezelése), az IMAP4-protokollt használja.
Az *5. képen* egy a POP3-ügyfél és -kiszolgáló közti jellemző párbeszédet mutatunk be.
Ebben a példában a ***cheetah*** nevű gép a kiszolgáló (kék színnel), míg a ***dolphin*** az ügyfél (piros színnel jelölve). A kiszolgáló itt is üdvözlő sorral fogadja az ügyfelet, minden válasz egy állapotjelzővel (`+OK`, `-ERR`) kezdődik, és ezt követi az üzenet.
A belépés a `USER` és `PASS` parancsokkal történt. Mint látható, ez esetben a felhasználó neve és jelszava minden védelem nélkül kerül továbbításra. Belépés után a `STAT` paranccsal megnéztük, hogy hány levél vár ránk (1 levél 559 bájt hosszan). Ezután a `LIST` parancs hatására a levelek listáját is megmutatja. A sorszámozás eggyel kezdődik, a sorszámok kiosztását a POP3-kiszolgáló a belépéskor végzi el. Ezekután letöltjük az első levelet, majd le is töröljük. A törlés ilyenkor még nem hajtódik végre, csak miután kilépünk (`QUIT`). Fontos megfigyelni, hogy a levél hogyan kerül továbbításra. Ha jobban megnézzük, itt is az SMTP-protokoll esetén megismert levélvégejelzés használatos.
A fejlettebb POP-kiszolgálók pusztán a levél elejének letöltésére is lehetőséget adnak. Ez akkor lehet hasznos, ha modemes kapcsolattal rendelkezünk, és nem szeretnénk az olykor nagyméretű levélszemeteket vagy egyéb kéretlen leveleket letölteni. Néhány ügyfél lehetővé teszi, hogy a levélfejlécek letöltése után leveleinket törölhessük. Ennek használatához az ügyfélnek és a kiszolgálónak támogatnia kell a `TOP` parancsot.

A jelszó nyílt elküldése (főleg az Interneten át) nem szerencsés. Kikerülésére több megoldás is született: az egyik az `APOP` parancs használata, amelyre a *6. képen* láthatunk példát. Itt a kiszolgáló üdvözlő üzenetében egy rejtvény található (a példában ez a `<1219.1014038493@cheetah.andrews>` karaktersorozat). A cél, hogy erre az `APOP` parancsban a megfelelő választ adjunk. Az `APOP` parancsban el kell küldeni a felhasználó nevét, valamint a választ, ami a rejtvény és a jelszó egymás után írásából származó karaktersorozat MD5-értéke karakterlánc formában. A válasz így tehát kiszámítható (amennyiben a jelszó „netuddki”):

```
$ echo -n '<1219.1014038493@cheetah.andrews>
➥netuddki' | md5sum
a396cd1609f95c6fc00822c7689df3e9  -
```

Ezenkívül az IMAP4-protokollnál bevezetett többféle azonosítási módszer használata is lehetséges. Ezekre az IMAP4-protokoll ismertetésekor térünk ki. A POP3-protokoll bővítése a TLS (Transport Layer Security – az SSL utódja) használatát is lehetővé teszi. Célja, hogy a csatorna a jelszó vagy a levél lehallgatásától, valamint az egyéb támadásoktól védve legyen.
A POP3-protokoll egyszerűsége ellenére a kiszolgálómegvaló-



7.

sítások sajnos sokszor súlyos biztonsági hibákat tartalmaznak. Amennyiben belső POP3-kiszolgálót kívánunk üzemeltetni, az elérhetőség körét célszerű a szükségesre korlátozni. Ne feledkezzünk meg arról, hogy a cikk elején vázolt általános levelezési gondok a POP3-protokollt ugyanúgy érintik. A protokoll csomagszűrése egyszerű: a kiszolgálók szabványos `110/tcp`-kapun várnak az ügyfelekre. Ezenkívül használhatják még a `995/tcp`-kaput is, ami POP3S, ami SSL-es védelmet jelent. A POP3S szükségessége fokozatosan csökken, mivel a szabványos változat is képes már TLS-t használni.
Amennyiben a belső ügyfélgépek szeretnék az Interneten elhelyezett POP3-kiszolgálókat elérni, ellenőrizzük a kiszolgálót, valamint az ügyfélprogramot, hogy támogatja-e a titkosítást (TLS vagy POP3S), illetve milyen azonosítást használ.
Az alábbi példa-beállítás lehetővé teszi, hogy a védett gépek elérhessék a külső POP3- és POP3S-kiszolgálókat.

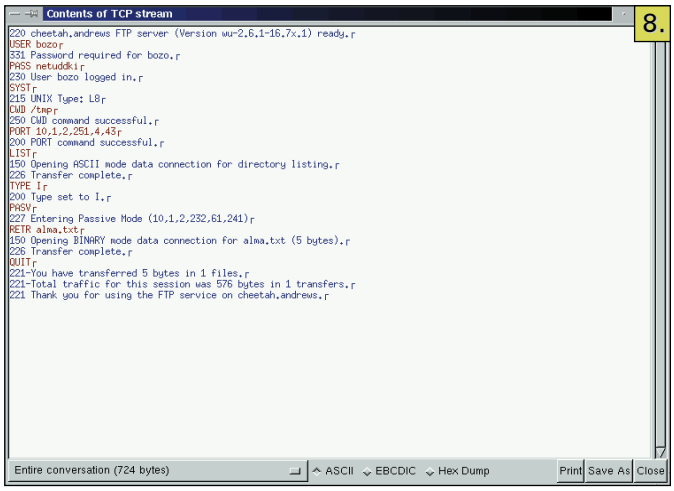
2.2.x rendszermagsorozatnál:

```
ipchains -A input -i eth0 -d 0.0.0.0/0 pop3
➥-p tcp -j ACCEPT
```

```
ipchains -A input -i eth0 -d 0.0.0.0/0 995
➥-p tcp -j ACCEPT
```

2.4.x rendszermagsorozatnál:

```
iptables -p tcp -A INPUT -i eth0 --dport pop3
➥-j ACCEPT
iptables -p tcp -A INPUT -i eth0 --dport 995
➥-j ACCEPT
```



IMAP4

Az IMAP4-protokoll célja, hogy a felhasználó több helyről is kezelhesse a postaládáit. Ez a POP3-protokollal szemben nem csak a levelek letöltésére és törlésére készült. A levelek ilyen esetben a kiszolgálón maradnak, a felhasználó ott végezhet velük műveleteket. A protokoll csak a levelek kezelését végzi, a küldésre az SMTP-protokoll használható.

A protokoll meglehetősen bonyolult, ezért csak ízelítőt adunk a működéséből. A TCP-kapcsolat felépítése után az ügyfelet egy üdvözlősor várja, amelyben a kiszolgáló által támogatott kiegészítések is fel vannak sorolva. Az ügyfél ezután kérést küld a kiszolgálónak. Minden kérés elején egy egyedi azonosító helyezkedik el, ezt követi a parancs, valamint a parancshoz tartozó paraméterek. A *7. képen* ezek jól láthatók, valamint az is, hogy miként lehet egy IMAP-kiszolgálóra bejelentkezni. Mint a kép is mutatja, a jelszó ez esetben is kódolatlanul halad át a hálózaton, amely komoly biztonsági hibát jelenthet. Ez ellen többféle módon is védekezhetünk, az egyik lehetőség a megfelelő azonosítási módszer használata. A fejlettebb IMAP-kiszolgálók a `LOGIN` parancson kívül lehetővé teszik az `AUTHENTICATE` parancs használatát, amelynek segítségével számos azonosítási módszer áll a rendelkezésünkre (például Kerberos és többféle challenge-response azonosítás). Ezenkívül a kapcsolatot magát is védhetjük titkosítási eszközökkel: elterjedt módszer a kapcsolat SSL-be ágyazása (IMAPS), és egyre több kiszolgáló támogatja a TLS-t mint protokollbővítést.

Az IMAP-protokollt csomagszűrővel védeni egyszerű, bár a protokoll összetettsége miatt a kiszolgálóprogram kiválasztására fokozottabban ügyelni kell. Az IMAP-kiszolgálók alapesetben ügyfeleikre a `143/tcp`-kapun várakoznak . Az `"IMAP over SSL"` a `993/tcp`-kaput használja.

Az alábbi példa beállítás lehetővé teszi, hogy a védett gépek elérhessék a külső IMAP4- és IMAP4S-kiszolgálókat.

2.2.x rendszermagsorozatnál:

```
ipchains -A input -i eth0 -d 0.0.0.0/0 imap
➥-p tcp -j ACCEPT
ipchains -A input -i eth0 -d 0.0.0.0/0 993
➥-p tcp -j ACCEPT
```

2.4.x rendszermagsorozatnál:

```
iptables -p tcp -A INPUT -i eth0 --dport imap
➥-j ACCEPT
iptables -p tcp -A INPUT -i eth0 --dport 993
➥-j ACCEPT
```

Állományok mozgatása

A levelezés áttekintése után nézzük meg, miként szokták szállítani a különböző állományokat az Interneten. E szempontból főképpen két protokoll népszerű: az FTP és a HTTP. Az általános kérdésekről (például vírusok és trójai programok) természetesen itt sem szabad megfeledkeznünk, bár kivédésükre most nem térünk ki. A HTTP protokollról és a Web veszélyeiről későbbi írásunkban szólunk részletesebben, most az FTP protokollt vizsgáljuk meg.

FTP

Az FTP szintén az Internet egyik régi alapprotokollja, hiszen az állományok mozgatásának szükségessége már a kezdetek kezdetén felmerült. A protokoll célja, hogy állományok mozgatását tegye lehetővé két számítógép között, a rendszerek eltérő sajátosságainak figyelembevételével. Mit is értünk ez alatt? Még az olyan egyszerű dolgok, mint a szöveges állományok is különbözőek lehetnek a rendszerek között. A DOS/Windows-alapú rendszereken a szövegállományok sorait `CR-LF` bájtsorozat határolja, míg a Unix-alapú rendszerek csak egy `LF` karaktert használnak a sor végének jelölésére. Léteznek olyan rendszerek is, amelyek a sorok végét egy `CR` karakterrel jelzik. A régebbi időkben ennél különösebb eltérések is léteztek a rendszerek között, de az ilyen rendszerek az idők folyamán háttérbe szorultak.

Az FTP (a legtöbb protokolltól eltérően) két független csatornát használ. Az egyik a parancscsatorna, ahol az ügyfélprogram utasíthatja a kiszolgálót, a másik az adatcsatorna, ahol az állományok és a könyvtárlisták közlekednek. Míg a parancscsatorna a kapcsolat ideje alatt fennmarad, addig az adatcsatorna szükség esetén megnyílik, majd az állomány- vagy könyvtárlista átvitele után lebomlik. Az adatcsatorna felépítésére két módszer létezik. Alapesetben a kiszolgáló a kezdeményező fél. Ekkor az ügyfél egy dinamikus TCP-kaput hoz létre, amelynek címét tudatja a kiszolgálóval. A kiszolgáló a `20`-as TCP-kapuról kapcsolódik az ügyfél megadott kapujára – ezt hívják aktív módnak. A másik mód esetén a `PASV` parancs hatására a kiszolgálóprogram egy dinamikus kaput hoz létre. A létrehozott kapu címét a válaszban megadja. Ebben az esetben az ügyfélgép kezdeményezi az adatcsatorna felépítését a kiszolgáló felé. Szaknyelven ezt passzív módnak hívják. A *8. képen* mind az aktív, mind a passzív módra láthatunk példát. A */tmp* alkönyvtár listáját aktív módban töltöttük le, az ***alma.txt*** állományt pedig passzív módban. Mind a `PORT`, mind a `PASV` parancsnál a címet az alábbi módon kell értelmezni: az első négy bájt az IP-cím, az utolsó kettő a kapu címe két bájtra bontva. A nagyobb helyiértékű bájt szerepel elöl. A képen ezek szerint az ügyfélgép a `PORT` paranccsal azt jelentette be, hogy a `10.1.2.251:1067` TCP-kapuján vár a kiszolgáló által küldött adatra, a `PASV` parancs válaszából kitalálható, hogy az ügyfélnek az adatért a `10.1.2.232:15857` TCP-kapura kell csatlakozni.

Mint a példából is jól látható, a teljes protokoll (és benne a jelszó) szövegalapú, és a jelszó is nyílt szövegként kerül további-

tásra. Ez súlyosan veszélyeztetheti a rendszereink biztonságát. Az eddig említett protokollokkal (SMTP, POP3, IMAP4) szemben az FTP protokoll biztonsági kiegészítése ugyan megtörtént, de nem terjedt el széles körben. Az FTP protokollal kapcsolatban egyéb biztonsági gondok is felmerülnek. Képzeljük el, hogy egy támadó állományt helyezhet el az FTP-kiszolgálón, és egy megtámadandó rendszer megadott kapujára (például a `telnet`-kapura) állományátvitelt kezdeményez. Amennyiben a támadó az állományt megfelelően állítja össze, úgy az „állományátvitel" leple alatt be tud jelentkezni az adott gépre, és parancsokat tud kiadni, hiába nem lenne képes közvetlenül az adott szolgáltatásra csatlakozni. Ezt a támadási formát a „FTP bounce attack" névre keresztelték. Megakadályozására az FTP-kiszolgálók aktív mód esetén megkövetelik, hogy a célcím a parancscsatorna forráscíme, és a célkapu 1023 feletti legyen. Egy másik lehetséges támadási forma az állományok ellopása. Ezt passzív módú kapcsolatok esetén lehet kihasználni. Ilyenkor a támadó az FTP-kiszolgáló dinamikus kapuira próbál kapcsolódni. Amennyiben a kiszolgáló a passzív átvitel során folyamatosan növekvő kapukat használ, a támadó a pásztázandó tartományt egy könnyen végigpróbálható kis szeletre tudja korlátozni. Ebben az esetben a támadó más felhasználók állományaihoz is hozzáférhet.
Az adatkapcsolat forráscíme a parancscsatorna forráscímével ugyanebben az esetben is vizsgálható lenne, de ezt az FTP-kiszolgálók nem mindegyike teszi meg. Mindkét galibát a külön adatcsatorna jelenléte okozza, ám az ezáltal okozott nehézségek sora ezzel még nem ért véget. Egy FTP-ügyfelet vagy -kiszolgálót hagyományos csomagszűrővel védeni rémálom. Amennyiben FTP-ügyfeleket kell védeni, megfelelő megoldás lehet a passzív mód, hiszen ilyenkor mind a parancscsatorna, mind az adatcsatorna kimenő kapcsolat. Ne felejtsük el azonban, hogy az adatcsatornának mind a forrás-, mind a célkapuja dinamikus, így szinte tetszőleges TCP-kapcsolatot ki kell engednünk (minden kapcsolatot, amelynek forrás- és célkapuja 1024 feletti).
Az aktív mód engedélyezése még veszélyesebb. Ebben az esetben a kívülről érkező csomagokat be kell engedni, ha forráskapujuk 20-as és célkapujuk 1023-nál nagyobb. Ne feledjük, hogy ebben a tartományban helyezkednek el az olyan programok, mint az NFS-kiszolgáló vagy az X-kiszolgáló. Ennek hatására a támadó a 20-as forráskapuról kapcsolatot képes kezdeményezni többek között ezekre az alkalmazásokra is.
Állapottartó csomagszűrőkkel (SPF – Stateful Packet Filter) a helyzet sokat javult, hiszen ezek figyelik a parancscsatornát, és az adatcsatorna létesítésére vonatkozó szabályokat maguk helyezik és távolítják el. Szerencsére a 2.2-es Linux-rendszermagsorozat (masquerading) része ilyen, 2.4-es rendszermag esetén pedig a Netfilter hagyományos forgalomirányítás esetén is képes a kezelésére. Amennyiben az FTP protokollt Linux-tűzfalunk csomagszűrőjével szeretnénk védeni, ne felejtsük el betölteni az FTP-protokollt támogató modulokat (2.2-es sorozat esetén az `ip_masq_ftp.o`, 2.4-es sorozat esetén a `ip_conntrack_ftp.o`, és NAT használata esetén `ip_nat_ftp.o` modulokat).
Az SPF-tűzfalak sem mindig védenek meg azonban az FTP protokoll hibáitól. Az FTP protokoll kellő védelme csak alkalmazásszintű tűzfalakkal lehetséges. Amennyiben csak anonim FTP-t kívánunk használni (eléggé gyakori eset), célszerű lehet egy FTP proxy használata, például a Squid gyorstáré. A Squid (lásd még Linuxvilág 2–3. szám, 74. oldal) webböngészővel egyszerűen használható, valamint az FTP- mellett a HTTP protokollt is támogatja. Külön előnye, hogy gyorstárként is működik, így használatával értékes hálózati sávszélességet nyerhetünk. Az ilyen nagyméretű programokat célszerű elkülönítőbe (jail) zárni, és amennyiben lehetséges, külön gépre elhelyezni. Az FTP beállítás a 2.2.x rendszermag esetén jelentősen eltér masquerading használatakor. Mivel masquerading használatakor a beállítás sokkal egyszerűbb, most csak a hagyományos megoldást mutatjuk meg (amikor nincs szükség címfordításra).

2.2.x rendszermagsorozatnál:

```
ipchains -A input -i eth0 -d 0.0.0.0/0 ftp
➥-p tcp -j ACCEPT
```

amennyiben a passzív módot szeretnénk engedélyezni:

```
ipchains -A input -i eth0 -s 0.0.0.0/0 1024:
➥-s 0.0.0.0/0 1024: -p tcp -j ACCEPT
```

amennyiben az aktív módot szeretnénk engedélyezni:

```
ipchains -A input -i eth1 -s 0.0.0.0/0 20
➥-d 0.0.0.0/0 1024: -p tcp -j ACCEPT
```

2.4.x rendszermagsorozatnál:

```
iptables -A INPUT -p tcp -i eth0 --dport ftp
➥-j ACCEPT
iptables -A INPUT -p tcp -i eth0 -m state
➥--state RELATED -j ACCEPT
```

Áttekintés

Az előző fejezetekben néhány gyakran használt internetes protokoll tulajdonságaival, valamint veszélyeivel ismerkedtünk meg. Nem szabad azonban elfelejteni, hogy a programokban lévő megvalósítási hibák külön veszélyt jelentenek. E programok sajnos általában rendszergazdai jogosultságokkal futnak, hogy 1024 alatti kaput foglalhassanak maguknak, a felhasználót azonosíthassák a rendszer adatbázisaiból, vagy átválthassanak a felhasználó jogaira. Emiatt egy a futtatásának korai állapotban sikeresen megtámadott program (mielőtt még rendszergazdai jogosultságait eldobhatta volna) igen súlyosan veszélyezteti a rendszer egészének biztonságát. A rendszergazda ellen még az elkülönítők (jail) sem mindig jelentenek tökéletes védelmet.

A fenti protokollokkal kapcsolatban érdemes elolvasni a 30. CD-n (Magazin/Konnyu) található irodalomjegyzékben lévő írásokat.

Hivatkozások

[1.] Linuxvilág: 2002. 1–2 szám, 50. oldal – Könnyű álmok: A protokollok (11. rész)
[2.] Linuxvilág: 2001. 8. szám, 56. oldal – Könnyű álmok: Hálózati forgalom vizsgálata (8. rész)

Borbély Zoltán (bozo@andrews.hu), okleveles mérnök-informatikus. Főként Linuxon futó számítógépes biztonsági rendszerek tervezésével és fejlesztésével foglalkozik. A 1.0.9-es rendszermag ideje óta linuxozik. Szabadidejét barátaival tölti.

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