EMTP, EGY ÚJ LEVELEZÕ PROTOKOLL ÉS IMPLEMENTÁCIÓJA

Iványi Tibor, *ivanyit@tigris.klte.hu*
Csukás Levente, *csukasl@fox.klte.hu*
Kossuth Lajos Tudományegyetem Informatikai és Számító Központ

Abstract

The well known SMTP (Simple Mail Transfer Protocol) based mailing systems have security holes. Since the client side can be a simple telnet client, everyone can connect to SMTP servers on port 25. With this method unnamed letters can be sent. To eliminate this problem we define a new protocol, Endpoint Mail Transfer Protocol. This protocol connects the two communicating endpoint directly and includes security checking of letters. It automatically determines the sender's data, uses a privileged port for communication and incereases the security for both the client programs and the server daemons.

Bevezetés

A jól ismert és széles körben használt SMTP [RFC821] levelez õ protokoll egy biztonsági szempontból történõ kibõvítését kívánjuk definiálni. Az SMTP kliens-szerver modellel mûködik, méghozzá olymódon, hogy a levelet küld õ oldal a kliens, a levelet fogadó oldal a szerver szerepét játssza. Ez azt jelenti, hogy a levelet küldõ process csatlakozik a levelet fogadó oldalhoz, annak átadja a feladó és a levél adatait, amikre válaszol a szerver oldal. A hiányosság abban áll, hogy a szerver oldal ellen õrzés nélkül "elhiszi", amit neki a kliens mond. Továbbá, mivel a kliens lehet a szintén jól ismert *telnet* program is, látható, hogy a névtelen, illetve hamis feladóval küldött levelek küldése lehetséges és rendkívül egyszer û.

Az EMTP (Endpoint Mail Transfer Protocol) ezen a hiányosságon próbál meg segíteni. Kibõvített biztonsági rendszere segítségével a hamis feladójú levelek felismerését és kiszelektálását teszi lehetõvé. Mint neve is mutatja a leveleket közvetlenül a feladó és a címzett gépe között továbbítja.

Az EMTP célja tehát elektronikus levelek továbbítása megbízható és biztonságos módon.

1. Kommunikációs modell

1.1 Komponensek

Az EMTP rendszer több komponensbõl áll. A teljes feladatrendszer komponensekre való bontásánál alapvetõ szempont, hogy

– funkcionálisan különbözõ részek különbözõ komponensbe kerüljenek,
– ne legyen túl sok komponens.

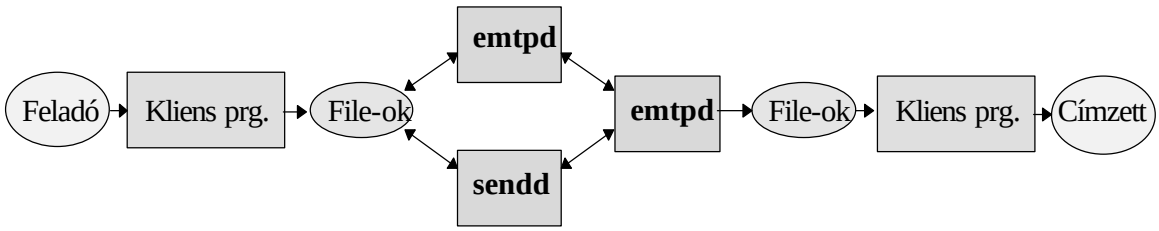
Ezek szerint az EMTP alapvetõen három komponensre bontható :

1. emtpd daemon
2. sendd daemon
3. kliens programok

Az emtpd feladata a levelek átvétele a fogadó gépen, továbbá a biztonsági egyeztetés. A sendd a levél queue kezeléséért és a levelek elküldéséért felel õs. A kliens programok a felhasználó felé szolgáltatásokat — például levelek küldése, olvasása — nyújtanak.

1.2 Kapcsolat a komponensek között

A következõ ábrán szemléltetjük, hogy az egyes komponensek hogyan kapcsolódnak egymáshoz, illetve a levelezésben résztvevõ más objektumokhoz.



1. ábra
EMTP komponensek kapcsolata

Ezen az ábrán látható, hogy a felhasználó csak a kliens programmal áll kapcsolatban. A kliens program a file-rendszer egy bizonyos file-jához fûzi a feladó levelét, a továbbításhoz szükséges információkkal kiegészítve. Ezen file alapján a sendd továbbítja a levelet a címzett gépére. Itt az emtpd daemon megkezdi a levél átvételét, biztonsági egyeztetésre csatlakozik a feladó emtpd-jéhez, majd a felhasználó postaládájába teszi az átvett levelet. Ezt a címzett egy kliens program segítségével elolvashatja, válaszolhat rá. A most következõ részben részletesen áttekintjük az egyes komponensek mûködését, funkcióját.

2. Mûködés

A részegységek mûködésének leírását fordított sorrendben végezzük, ez a sorrend a levél küldése során a tevékenységek idõbeli sorrendje.

2.1 Kliens programok

A kliens programok elsõsorban a felhasználóval való kapcsolattartást szolgálják. Ebbõl eredõen tudniuk kell a már ismert levelez õ kliens szolgáltatásokat, például levelek olvasása a felhasználó postaládájából, levelek küldése. Más levelez õ programok által nyújtott szolgáltatások is beépíthet õk a kliensekbe, például ún. attachment-ek küldése, mint a *pine*, vagy *pmail* programokban. Természetesen a kliensek nagyon széles skálán mozoghatnak, a parancssorból vezérelhet õtõl a multimédia leveleket is kezelni tudó grafikus levelez õ kliensekig.

Az EMTP kliensek nagyon fontos feladata, hogy a felhasználó nevét nem kérik be a feladó címének létrehozásához, hanem azt automatikusan a futás során megállapítják. Ha szükséges adott felhasználók neveihez ún. alias-okat rendelni a kliensnek képesnek kell lennie, hogy a megállapított felhasználói név alapján az alias nevet, ha létezik, azonosítsa és használja a címben. Természetesen a biztonságosság megköveteli, hogy az alias adatbázist csak privilegizált felhasználók tudják módosítani. A felhasználó nevének automatikus megállapítása lehetetlenné teszi a klienst használónak a más felhasználói név alatt történ õ levél küldését.

A kliens a levél átvétele után a levelet és a feladó nevét (felhasználói név, vagy alias) hozzáfûzi egy file-hoz, amit írni és olvasni természetesen csak privilegizált jogokkal lehet.

2.2 Sendd daemon

A sendd daemon gyakorlatilag a levél átvitelénél a kliens szerepét tölti be kommunikációs szempontból. A levelezõ kliens programok által az úgynevezett *append-file*-hoz fûzött leveleket ebbõl a file-ból kiveszi, berendezi õket az általa menedzselt queue-ba és továbbítja a címzett gépe felé.

A sendd két egymástól független ciklusban kering.

– Queue Read ciklus

Ebben a ciklusban a sendd konfigurálható idõközönként elolvassa az append-file-t, és ha az nem üres, akkor egyesével kiveszi a leveleket és a queue-ba teszi õket. A queue felépítése a következõ : minden egyes levél kap egy üzenet azonosítót, amit az

idõ*sorszám

alakban állítunk elõ. Itt az idõ másodpercekben mérendõ, a sorszám pedig a sendd indulásától számítva a levél sorszáma a továbbítandó levelek között. (Az implementáció során Unix fölött az idõ meghatározása történhet például a time() C függvény segítségével.) Erre az üzenet azonosítóra azért van szükség, hogy minden egyes levelet egyértelmûen azonosítani tudjunk. Látható, hogy pusztán a feladó és a címzett nem elegendõ a levél egyértelmû azonosításához. Pusztán az idõ is kevés, mert elképzelhetõ, hogy egyetlen másodperc alatt ugyanazon feladó ugyanazon címzettnek két levelet is küld. A sorszám önmagában szintén kevés, mert ha a sendd-t valamilyen okból újra kell indítani, több azonos sorszámú levél létezhet ugyanahhoz a feladó-címzett párhoz.

Minden egyes levélbõl a queue-ban két file lesz. Az egyik tartalmazza a levél szövegét, a másik csak egy számot, hogy a sendd hányszor próbálta már meg továbbítani a levelet. Ha ez egy konfigurálható értéknél nagyobbra nõ, akkor valamilyen módon hibajelzést ad a sendd.

Ezen kívül a sendd ebben a ciklusban a meglevõ levelek alapján további file-okhoz is hozzányúl. A biztonsági egyeztetéshez szükséges file-okba beírja az adott, új levelek adatait. A mostani megvalósításban mindaddig, amíg az összes queue-ban levõ levél kézbesítése nem sikerült, az összes levél a queue-ban marad, csak egy külön helyen megjegyzi a sendd, hogy az adott levelek közül melyikkel kell még foglalkoznia.

– Letter Send ciklus

A sendd ebben a ciklusban szintén konfigurálható idõközönként leellenõrzi, hogy mely levelek nincsenek még továbbítva a queue-ból. Minden egyes levélhez egy külön folyamatot indít, a továbbiakban a gyermek foglalkozik a levéllel. A gyermek megvizsgálja, hogy a próbálkozások száma alapján kell-e még a levéllel foglalkoznia. Ha igen, akkor kapcsolódik a címzett gép emtpd daemonjához és a levelet továbbítja. Ha ez sikerült, jelzi a queue-ban, hogy az adott levéllel további tennivaló nincs.

Ha a szülõ úgy találja, hogy a queue-ban levõ összes levél sikeresen elment, törli a queue-ból az összes bejegyzést. Megjegyezhet õ, hogy a queue ezzel a módszerrel történõ kezelés mellett esetleg túl nagy méretûre nõhet. Ha a tapasztalat ezt igazolja, akkor a queue méretének ésszerûen kis értéken tartásához a sendd vagy rendszeres idõközönként, vagy ha a queue mérete egy adott értéket meghalad, újrarendezheti a queue-t.

2.3 Emtpd daemon

Ez a daemon a levél átvételéért, annak a címzett postaládájába írásáért, illetve a biztonsági egyeztetéséért felelõs. A következõkben lépésrõl lépésre áttekintjük a daemon mûködését, együtt a protokoll utasításainak leírásával.

– Miután a feladó gépe sikeresen kapcsolódott a címzett emtpd-hez, a sendd küld egy **letter send** parancsot az emtpd-nek, a következõ formában :

ls|<sname>|<aname>**@**<machine>|<messgid>

A parancsokban a szeparátor karakter a |.

– Az **ls** a letter send parancs kétbetûs kódja.
– <sname> a feladó felhasználói neve, vagy alias-a.
– <aname> a címzett felhasználói neve, vagy alias-a.
– <machine> a címzett gép IP neve.
– <messgid> a levél azonosítója, a sendd-nél leírt formátummal.

Látható, hogy az utolsó elõtti mezõ a klasszikus email címe a címzettnek.

– Az címzett emtpd küld egy **request for identification** parancsot a feladó gépén futó emtpd-nek, a
rid|<sname>|<aname>**@**<machine>|<messgid>
formában, ahol a **rid** a request for identification parancs kódja, a többi mezõ az ls-nél leírtakkal egyezik meg. Látható, hogy a feladó gép a saját címét nem küldi át. A címzett emtpd a feladó gépének nevét a kapcsolat elfogadásakor határozza meg. Innen tudja, hogy hova kell csatlakozni a rid kéréssel. Megjegyezzük, hogy a tervek szerint az összes EMTP-vel levelezni kívánó gépnek name service-ben szereplõnek kell lennie.

– A feladó gépén futó emtpd a címzett rid kérésének megfelelõ küldési szándékot leellenõrzi a queue-ban. Ha az adott levélhez tartozó bejegyzés szerepel, akkor visszaküld egy **identification ok** választ, egyébként egy **identification not ok**-t.

– A címzett emtpd, ha pozitív választ kapott, leellenõrzi, hogy létezik-e a címzett a gépen, a rendszer szintû azonosító file-ok alapján, illetve az alias file alapján. Ha létezik, akkor küld egy **header data ok** üzenetet, ellenkezõ esetben egy **header data not ok**-t. Ez ellentétes az SMTP-vel, hiszen ott a levél elõbb átkerül a címzett gépre, csak aztán ellenõrzik, hogy létezik-e egyáltalán a felhasználó. Az EMTP megoldása kisebb hálózati forgalmat generál, hiszen ha eleve nem lehetséges a kézbesítés, nincs nagy mennyiségû átvitel.

– Ha ok-t küldött, a feladó sendd átküldi a levelet. Ha az átvétel sikerült, a címzett emtpd küld egy **letter acknowledged** választ és bontja a kapcsolatot. A feladó csak abban az esetben értékeli a levél átvitelét sikeresnek, ha a letter acknowledged választ megkapta.

– A címzett emtpd az átvett levelet hozzáfûzi a címzett postaládájához.

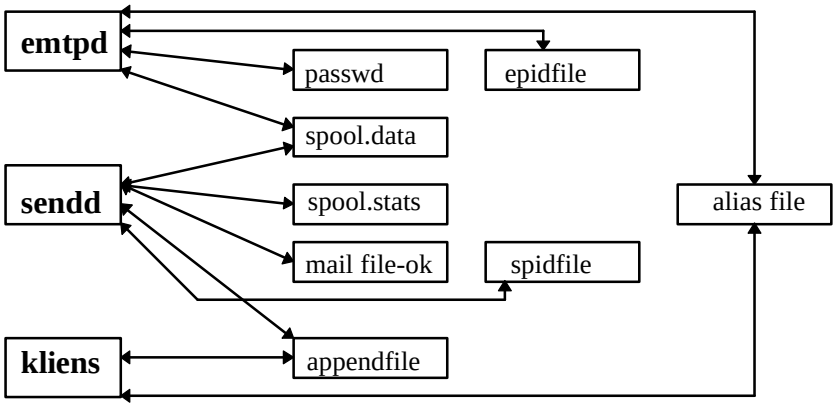
2.4 File-átvitel

A levél átvitele során az emtp egy primitív ftp-szerû protokollt használ a sendd és emtpd között. Sorban karakterenként továbbítódik a szöveg, a file végét **##** elküldése jelzi. (Természetesen a küldés során minden # jel egy (pl.) *#-space* párral helyettesítõdik.) A késõbbi verziók támogatni fogják a levelek átküldés elõtti tömörítését is.

3. Implementáció

Mi az implementációt Unix fölött, C-ben végeztük. Az emtpd-hez szükséges egy TCP [RFC 793] port szám. A biztonságosság miatt egy privilegizált portot használtunk. A hivatalosan még ki nem adott port számokról az RFC 1060-ban találhatunk információt.

A következõ ábrán illusztráljuk, hogy az implementáció során milyen file-okkal dolgoztunk.



2. ábra
Emtp implementációban használt file-ok.

– *passwd* a Unix-ban ismert /etc/passwd,
– *epidfile* egy szöveges file, amibe az emtpd imdulásakor beleírja a process id-ját,
– *spool.data* a küldésre el õkészített levelek fejrészei,
– *spool.stats* a queue-ban lev õ levelek állapotának tárolásához,
– *mail file-ok* az említett két-két file levelenként,
– *spidfile* a sendd process id-ja,
– *appendfile* a kliensek leveleit tároló file.

A klienseket setuid móddal kell megírni root user-re, hogy írni tudjanak az append file-hoz.

4. Biztonság

Mindenekelõtt megjegyezzük, hogy a rendszert nem a privilegizált user-ek (pl. Unix-ban root) ellen kívánjuk védeni, hanem a normál jogosultságú felhasználók rosszindulatú tevékenységei ellen.

A biztonsági kérdéseknél három dologra térünk ki. Elõször annak a kérdését vizsgáljuk, hogy miért nem lehetséges emulációkkal kijátszani az EMTP védelmét.

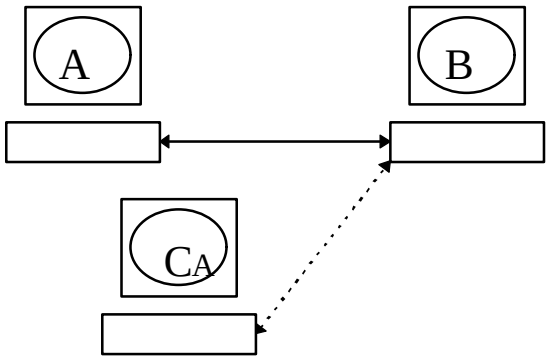
Tegyük fel, hogy egy adott gépen még nem mûködik az EMTP rendszer. Ekkor egy normál user megpróbálhat leemulálni egy emtpd daemon-t, amit õ írt és hamis leveleket továbbít. Erre azonban nincs lehetõsége, mivel a rendszerben az emtpd privilegizált portot használ, amit egy normál user nem tud magának leallokálni. Máshol viszont hiába próbál emtpd-t emulálni, hiszen a túlsó oldal a feladó privilegizált portjához fog csatlakozni a biztonsági egyeztetés miatt.

A kliens emulálása sem mûködik, hiszen a kliensnek setuid root-tal ellátottnak kell lennie, ezt normál user szintén nem tudja elérni.

A sendd emulálható, csakhogy a címzett emtpd a feladó emtpd-hez fog csatlakozni, ami a queue-ban nem fogja megtalálni a hamis sendd-vel küldött levelet, így a hamis levél a biztonsági egyeztetésen elbukik.

Ezeken túlmenõen egy normál user nem tud ugyanannak a gépnek egy másik felhasználója nevében levelet küldeni, hiszen a kliens a futtató személy kilétét automatikusan állapítja meg.

Lássuk végül a következõ elrendezést :



3. ábra
Hamis EMTP kommunikáció a C gépr õl

Ha itt a **C** az **A** nevében próbál meg kommunikálni a **B** emtpd daemon-nal, akkor az azért nem sikerülhet, mert a **B** a kapcsolat felvétele során megállapítja, hogy a **C** kapcsolódott hozzá és nem az **A.** Így tehát a másik gép egyik felhasználója nevében történ õ levél küldés sem sikerülhet.

Hivatkozások :

RFC 821	SIMPLE MAIL TRANSFER PROTOCOL, Johnathan B. Postel Information Sciences Institute, University of Southern California
RFC 793	TRANSMISSION CONTROL PROTOCOL Johnathan B. Postel Information Sciences Institute, University of Southern California
RFC 1060	ASSIGNED NUMBERS J. Postel, J. Reynolds