Állandó vírusvédelem megvalósítása a ClamAV segítségével

Linux alatt vírusvédelem? Minek az nekünk? Linux alatt nincsenek is vírusok! Sajnos ez így nem igaz, mert hiába írják a vírusok 99 százalékát más operációs rendszerre, senki sem garantálhatja azt, hogy ez mindig így marad. Ha pedig Wine-t használunk, akkor az alkalmazások mellé jó eséllyel kaphatunk vírusokat is, melyek így képesek futni Linux alatt.

Amennyiben ez még mindig nem lett volna elég indok arra, hogy ***Linux*** alá vírusirtót telepítsünk, akkor nézzünk néhány olyan példát, amikor szükségünk lehet arra, hogy ***Linux*** alatt végezzünk vírusirtást.
Ha van egy ***NFS*** kiszolgálónk, amit közösen használnak linuxos és windowsos kliensek, akkor a Windows-on futó kliensek védelmének érdekében a kiszolgálón futtathatunk ***ClamAV*** vírusellenőrzést.
A másik példában linuxos kliensünkkel kapcsolódunk ***RDP***-n keresztül egy ***Windows*** terminálhoz, olyan módon, hogy egy adott könyvtárat a terminálon is és a kliensen is látunk. Ha el szeretnénk kerülni azt, hogy a megfertőzzük a kiszolgálót, akkor kliens oldalon (is) ellenőriznünk kell a fájlok tisztaságát.
A ***ClamAV*** kitűnő ingyenes eszközt kínál erre a feladatra, a probléma csak az, hogy nem ismeri a rezidens vírusellenőrzést, tehát a fájlokat nem akkor ellenőrzi le amikor éppen hozzáférünk, hanem ha beállítjuk, akkor bizonyos időközönként periodikusan. Ez egyrészt egy kis biztonsági rést hagy a vírusoknak, hiszen lehetőségük van két ellenőrzés között terjedni, másrészt pedig adott esetben többszörös pluszmunkával terheli a gépünket, ha például minden éjjel leellenőrizzük a teljes megosztott könyvtár tartalmát, mikor napközben csak tíz fájlt változtattunk meg.

Rezidens vírusvédelem Linux alatt

A telepítést az ***Xubuntu 6.06 beta 2*** verzióval végeztem el, de lényeges különbség nincsen a korábbi ***Xubuntu***, ***Ubuntu***, ***Debian*** verziók esetén sem. A ***Dazuko*** modul felelős azért, hogy értesítse a ***Clamavot*** az egyes fájlműveletekről. A ***Dazuko*** modult, így saját kernelünkhöz kell fordítani. Szerencsére ehhez nem kell az egész kernelt újrafordítani, viszont szükségünk lesz a kernel forráskódjára, a kernel fejlécekre és a kernel fordításhoz szükséges csomagokra is, mivel a ***Dazuko*** modul a változóit a kernel forrásából állapítja meg. Adjuk ki a következő parancsokat, hogy rendelkezésünkre álljanak a szükséges csomagok.

```
apt-get install gcc-4.0
➥gcc-4.0-base binutils cpp-4.0
➥libcc1
```

esetleg a ***gcc-4.0*** helyett választhatjuk a ***gcc-3.4***-et is,

```
apt-get install build-essential
➥bin86 kernel-package
➥libqt3-headers libqt3-mt-dev
apt-get install linux-source-
➥2.6.15 linux-headers-2.6.15
```

Lépjünk be a könyvtárba, ahová letöltöttük a rendszermag forrását

```
cd /usr/src
tar -xvjf linux-2.6.15.tar.bz2
```

Hozzunk létre egy szimbolikus linket, mely az aktuális kernel forráskódjára mutat.

```
ln -s /usr/src/linux-2.6.15
➥linux
```

Másoljuk be a jelenlegi kernel konfigurációs állományát a kernel forráskód könyvtárba

```
cp /boot/config-2.6.15
➥/usr/src/linux/.config
```

Amennyiben mégis új kernelt akarunk fordítani, akkor már csak a libncurses5-dev csomagra lesz szükségünk a `make menuconfig` parancs futtatásához. Telepítsük a ***clamav-freshclam***, ***clamav*** és a ***clamav-deamon*** csomagokat, de a ***dazuko*** csomagot ne az ***apt***-n keresztül telepítsük, mert az ott található már nagyon régi, ezért jobban járunk, ha a hivatalos oldalról töltjük le a legfrissebbet.

```
wget http://www.dazuko.org/
➥files/dazuko-source_2.2.1-
➥1_all.deb
```

```
dpkg -i http://www.dazuko.org/
➥files/dazuko-source_2.2.1-
➥1_all.deb
```

Valószínűleg a ***dazuko-source*** csomag függőségi hibába fog ütközni, ezért telepítenünk kell a ***module-assistant*** csomagot is, ekkor már a ***dazuko-source*** is hiba nélkül feltelepül. Szükségünk lesz még a debhelper csomagra is, különben a ***modules-assistant*** képtelen lefordítani a ***dazuko***-t és állandóan hibát fogunk kapni a fordítás során. A ***module-assistant*** kielégítő működéséhez először futtassuk le a `module-assistant prepare` parancsot, majd pedig készítsük el a ***dazuko*** betölthető kernelmodulunkat az

```
m-a a-i dazuko
```

paranccsal.
Amennyiben minden rendben van, akkor végre betölthetjük a ***dazuko*** modult, de előtte a ***capability*** modult ideiglenesen távolítsuk el a rendszerből.

```
rmmod capability
modprobe dazuko
modprobe capability
```

Ahhoz, hogy mindez teljesen automatikus legyen, hozzuk létre a */etc/modules.d/dazuko* fájlt a következő tartalommal:

```
install dazuko modprobe -r
➥capability;\
modprobe -i dazuko; \
modprobe -i capability
```

Ezennel a ***dazuko*** figyeli a fájlműveleteket, de az ***Ubuntuban*** lévő ***ClamAV*** nincsen felkészítve arra, hogy együttműködjön a ***Dazuko*** modullal, ezért kis módosításokkal újra kell fordítanunk azt. A dolog nem annyira bonyolult, mint amennyire annak tűnik.

```
apt-get install fakeroot
➥build-essential
apt-get build-dep clamav
apt-get source clamav
```

Ezután lépjünk be az így létrejött *clamav_xx.x* könyvtárba és a */debian/rules* fájlban a `./configure` kezdetű sorokból vegyük ki a `--disable-clamuko` opciót. Majd pedig fordítsuk újra a *ClamAV*-ot a következő utasítással:

```
dpkg-buildpackage -rfakeroot
➥-uc -us
```

A folyamat végeztével egy könyvtárral feljebb megtaláljuk a telepítéshez szükséges ***deb*** csomagok, melyeket a `dpkg -i *.deb` paranccsal telepíthetünk is. Most meg kell mondanunk a *ClamAV*-nak, hogy milyen esetekben ellenőrizze le a fájlokat, illetve azt is meg lehet mondani, hogy mely könyvtárakban ellenőrizze le őket. A */etc/clamav/clamd.conf* állományhoz írjuk hozzá a következő két sort, mely megmondja, hogy fájl hozzáférés és fájlmegnyitás esetén fusson le az ellenőrzés.

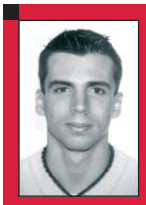
```
ClamukoScanOnAccess
ClamukoScanOnOpen
ClamukoIncludePath /home
```

A *Clamuko*, *Dazuko* együttműködéshez szükséges, hogy a *ClamAV root* jogosultságokkal fusson, így a *User* kezdetű sorban a *clamavot* írjuk át root-ra. Ha ezzel készen vagyunk, akkor már csak a teszt van hátra. Töltsük le az ***eicar*** antivírus tesztállományt (teszt szekvenciát) a ➲ http://www.eicar.org/download/eicar.com címről, majd pedig teszteljük le, hogy engedélyezi-e a rendszer a hozzáférést. Ha igen, akkor valószínűleg valamit nem jól csináltunk, ellenőrizzük le a lépéseket, hogy nem kaptunk-e valahol hibát. Amennyiben nem tudunk hozzáférni az adott fájlhoz (és amúgy a jogosultságunk meglenne hozzá), akkor működik a rezidens vírusvédelem *Linux* rendszerünkön.

Rezidens vírusvédelem Windows alatt

Aki egyáltalán nem találkozik *Windowsos* számítógépekkel, és nem is érdekli őket annak a lehetősége, hogy hogyan védjük meg a *Windowst* a vírusoktól tisztán szabad szoftverekkel, attól most elbúcsúznék, és köszönöm, hogy eddig velem tartott és végigolvasta a cikket, remélem hasznosnak találta.
Mivel a cikk témája a rezidens vírusellenőrzés *ClamAV* segítségével, ezért most kitérnék a *ClamAV Windows* alatt használható verziójára a *ClamWin*-re. Több dolog miatt is döntöttem amellett, hogy a *Linuxvilág* magazinban kitérjek erre a témára.
Mint magánembert sokan kerestek meg tanácsot kérve, hogy milyen programokat használjanak otthon (megértettem, hogy egyelőre nem szeretnének *Linuxra* váltani) ugyanakkor boldogan használnának biztonságosabb böngészőt, levelezőt, ingyenes és jogtiszta képszerkesztőt, hangvágót stb. Ezért mindenkinek az adott feladatra megfelelő szabad szoftvert ajánlom, esetleg telepítem fel a gépére. Pontosan ezért konzekvensen mindenkinek a *ClamAV*-ot ajánlottam mint vírusirtót, csak sajnos (eddig) hiányzott belőle a memória rezidens védelem. Mint rendszergazda munkám során találkozom *Windowsos* gépekkel is, és mindent megteszek azért, hogy a lehető legnagyobb biztonságot élvezzék ők is, de ezidáig nem tudtam megoldani azt, hogy a *ClamAV* csak azután engedje futni az alkalmazásokat, miután meggyőződött azok vírusmentességéről. Az esti vírusellenőrzés pedig lehet, hogy már csak eső után köpönyeg. Mint egyszerű user valahogy szerettem volna megoldani a családi pc védelmét is, szem előtt tartva azt, hogy az egyszerűség, a gyorsaság és az ingyenesség testesüljön meg a víruskereső szoftverben is, ha már kell otthonra egy *Windows* is.
Hogy a kép teljesebb legyen a *ClamAV*-nak létezik egy portja *Mac OS X* alá is. Ennek neve *ClamXav* mely egy letisztult, a *Mac OS X* kinézetéhez illeszkedő felületet nyújt a *ClamAV*-nak. Sajnos ha jól tudom, akkor egyelőre *Mac OS X* alatt a *ClamAV*-ot nem lehet rávenni a memória rezidens védelemre.
Szerezzük be a szükséges programokat a vírusellenőrzéshez, ha még nem tettük

volna meg. Szükségünk lesz tehát a *ClamWin*-re, melyet a ➲ *http://www.clamwin.com/* oldalról szerezhetünk be, illetve szükségünk lesz a *Winpooch* nevű okos kis alkalmazásra, melyet a ➲ http://winpooch.free.fr/home/index.php oldalon találhatunk meg.
A *Winpooch* egy remek szoftver mely a rosszindulatú szoftverek és trójai programok elleni védekezést segíti oly módon, hogy valós időben monitorozza az egyes folyamatok aktivitását a rendszerünkben. Segítségével engedélyezhetjük bizonyos programoknak, hogy adott könyvtárba írjanak, ugyanakkor egyéb könyvtárba megtilthatjuk az írás jogát. Ugyanígy megtehetjük azt is, hogy bizonyos programokat eltiltunk attól, hogy a regisztrációs adatbázisba írjanak, persze ekkor számolni kell azzal, hogy a program esetleg hibásan fog működni.
A program használatáról egy külön cikket is lehetne írni, de mi most csak a vírusellenőrzésre fogunk koncentrálni, azon belül is három különböző szabálytípust mutatok be.
Miután telepítettük a *Winpooch*-ot és a *ClamWin*-t is, láthatjuk, hogy a rendszer szinte minden egyes megmozdulására a *Winpooch* megkérdi tőlünk, hogy az adott műveletet engedélyezzük-e. Ha még nem zártuk ki magunkat a rendszerből :-), akkor mentsük el az alapértelmezett konfigurációs fájlt, és egy teljesen újat hozzunk létre helyette, mely kezdetben üres.
A program helyére írjunk egyszerűen csak egy * karaktert, mert a szabályunkat minden programra akarjuk érvényesíteni. A + ikonnal tudunk új szabályokat hozzáadni a jelenlegi konfigurációhoz.
Az első esetben ellenőrizzünk le minden *.doc* kiterjesztésű fájlt amikor azokat megnyitjuk. A szabályunk tehát a következő lesz, a *Reason* legördülő menüből válasszuk ki a *File::Read* opciót, a *Type* legördülő menüben válasszuk ki a *Wildcards* menüpontot, majd pedig adjuk meg a *Value* mezőnek a **.doc* kiterjesztést. A *Virus Scan* opciót feltétlenül válasszuk ki, majd mentsük el az első szabályunkat.
A második esetben egy adott könyvtár összes fájlját fogjuk leellenőrizni megnyitás előtt, például minden letöltött fájlt a *d:\downloads* könyvtárba töltünk le. A második szabály akkor annyiban fog különbözni az elsőtől, hogy a *Type* legördülő menüből a *Path with Wildcards* menőpontot válasszuk ki, és a *Value* mezőbe pedig írjuk be a *d:\downloads** karaktersort.
A harmadik szabály lesz talán a legértékesebb számunkra, ez fogja leellenőrizni a fájlokat még mielőtt azok futnának. A *Reason* legördülő menüből most válasszuk ki a *Sys::Execute* opciót, a *Type* legyen *Any Value*, értéke pedig * legyen. Az előzőekhez hasonlóan most is engedélyezzük a vírusellenőrzést.
Mindenkinek jó vírusirtást, de még inkább vírusmentességet kívánok.



Horváth Ernő
ernohorvath@gmail.com
24 éves, műszaki informatikus. Három évvel ezelőtt ismerkedett meg komolyabban a Linux rendszerekkel és emellett érdeklődik még a robotika és a biztonságtechnika iránt is. Ha lenne szabadideje sokat kirándulna, biciklizne és filmeket nézne.