Formális biztonsági modellek hálózati aspektusai

Magosányi Árpád <mag@lme.linux.hu>

2001.03.10

Kivonat

A formális biztonsági modellekkel kapcsolatos kutatás egyik kevéssé kutatott területe azok hálózatbiztonsági aspektusai. Erről a jelenlegi tűzfalpiac kétségbeejtő helyzete ékesen tanúskodik. Az előadás röviden bemutatja a legelterjedtebb formális modelleket, részletesebben elemezve a hálózatbiztonság szempontjából fontosabb modelleket, megadva a Bell-LaPadula modell egy a hálózati határvédelem nézőpontjába jól illeszkedő kiterjesztését. Lazításként megtekintjük a modell egy gyakorlati megvalósítását a Zorp tűzfalszoftver segítségével.

Tartalomjegyzék

1 Formális modellek

1.1 Diszkrécionális hozzáférésvezérlés

A diszkrécionális hozzáférésvezérlés (Discretionary Access Control, DAC) a gyakorlatban legtöbbször a mindenki által ismert *hozzáférésvezérlési listákat (access control list, ACL)* [1] jelenti. A dolog lényege az, hogy az egyes *objektumokhoz* (pl állomány,, könyvtár, adatbázis tábla, socket, shared memory szegmens) való hozzáférést a *szubjektumok* (felhasználók, illetve a nevükben futó folyamatok) számára az objektum tulajdonosa állítja be. A hozzáférésvezérlési listában implicit vagy explicit módon szerepel az objektum, a szubjektum és az engedélyezett illetve tiltott művelet.

A diszkrécionális hozzáférésvezérlés hátulütője az, hogy nem tudja garantálni az adatok bizalmasságát. Az ismert gondolatkísérlet szerint létezik a szigorúan bizalmas információ, amely az O_1 objektumban foglal helyet, Az O_1 objektum tulajdonosa S_1, aki beállítja annak hozzáférésvezérlési listáját oly módon, hogy az objektumhoz csak S_1-nek és S_2-nek legyen olvasási joga. Azonban S_2 (akár rosszindulatból, akár egy trójai "segítségével") az O_1 objektumban lévő információt elolvassa, majd kiírja az O_2 objektumba, amelyhez olvasási jogot ad S_3 részére. Ez a művelet akkor is elvégezhető, ha az információ megismerése S_3 számára kifejezetten tilos.

1.2 Kötelező hozzáférésvezérlés

A fenti szituáció elkerülésére hozták létre a *kötelező hozzáférésvezérlési modelleket (Mandatory Access Control, MAC)*. A modellek közös tulajdonsága, hogy nem az objektumokkal végezhető műveletekre, hanem az azokban tárolt információ áramlására fektetik a hangsúlyt.

Mivel a hangsúly az információáramláson van, a modellek tiszta formájukban csak az írás és olvasás műveletre tesznek megkötéseket.

Fontos alapelve még ezeknek a modelleknek, hogy az információt azok felhasználhatósági körét meghatározó *domainekbe* sorolják. Az információ besorolását az objektumok biztonsági attribútumaival, annak *címkéivel* írják le. A szubjektumokhoz is tartoznak címkék, amikből a szubjektum által elérhető domainek vezethetőek le.

Az egyes modellek közötti különbség tulajdonképpen a címkék és a domainek összerendelésében, és az engedélyezett műveletekben térnek el. Egy írási vagy olvasási művelet engedélyezettségét az $L_S \times L_O$ halmazon értelmezett reláció adja meg, ahol L_S a szubjektumok, L_O pedig az objektumok címkéinek halmaza. A két halmaz általában egyenlő, hiszen a szubjektum-szubjektum adatáramlásokat is kezelni kell. Ilyenkor az $L_{S_1} \times L_{S_2}$ halmazon értelmezzük általában ugyanazt a relációt.

Most pedig lássunk példákat ezekre a modellekre:

1.2.1 A Bell-LaPadula modell[6]

Ez a modell a klasszikus MAC modell, gyakran MAC alatt ennek a modellnek egyik vagy másik változatát kell érteni.

A bizalmasság kezelése

Az objektumokhoz és szubjektumokhoz hozzárendeljük az azok *biztonsági szintjét ((Hierarchical) security level)* jelző címkéket. A címkéken értelmezünk egy teljes rendezési relációt ($\geq$), és a következő relációk írják le az olvasás illetve írás engedélyezettségét:

- az olvasás engedélyezett, ha $L_S \geq L_O$
- az írás engedélyezett, ha $L_O \geq L_S$ (A gyakorlati implementációk általában csak az egyenlőséget engedik meg.)

Tehát az információ mindig csak "fölfelé" terjedhet. Ennek a modellnek a célja, hogy bizalmas információ soha ne kerülhessen alacsonyabb biztonsági szintre. A modellt figyelmesen szemügyre véve az embernek támad valami kényelmetlen érzése attól, hogy egy alacsony biztonsági szinten lévő szubjektum felül tud írni egy nála magasabb biztonsági szinten lévő objektumot. Ez így van, a modell ebben a formájában csak az adatok bizalmasságával foglalkozik, azok integritása nem szempont. Az integritás kezelését ACL-ekkel lehet megoldani, vagy a modell következő kiegészítésével:

Integritási címkék

Az objektumoknak a biztonsági szintjükön kívül még egy címkéjük van. Nevezzük ezt *integritási címkének (integrity label)*. Az integritási címke tartalma 0 vagy több *(nemhierarchikus) kategória ((Non-hierarchical) categories)*. A kategóriák közötti rendezési reláció megfelel a halmazelméleti részhalmaz fogalmának. Az írásra és az olvasásra ugyanaz a szabály vonatkozik:

- az írás/olvasás engedélyezett, ha $L_S \supseteq L_O$

A kategóriák kiosztásának szabályai általában az adat felhasználásának célját, a felhasználó szervezeti hovatartozását tükrözik.

A fenti két modell ötvözése (amikor a biztonsági szintek és a kategóriák együttesen határozzák meg a hozzáférés engedélyezését) a klasszikus Bell-LaPadula modell. Ebben a formájában definiálja a TCSEC[9] B1 osztálya valamint a CC[10] Labeled Security Protection Profile-ja (LSPP[11]) is. Természetesen a modellnek számos kiterjesztése létezik. Ezek egyike a szubjektum címkéjének meghatározásáról szól:

1.2.2 A Bell-LaPadula modell dinamikus címkékkel

A felhasználók biztonsági szintje általában az általuk éppen végzett feladattól függ. A gyakorlatban a felhasználóhoz általában egy maximális biztonsági szintet rendelnek amit elérhet, és vagy azon, vagy annál alacsonyabb szinten dolgozik. Az aktuális biztonsági szint megváltoztatásához be kell lépnie, vagy azzal felhasználói szmpontból egyenértékű műveletet kell végeznie. Ennek elkerülésére találták ki azt, hogy a szubjektumoknak két biztonsági szintje van: egy minimális és egy maximális. Ezek eredetileg a felhasználó minimális és maximális biztonsági szintjére vannak beállítva, és az objektumokhoz való hozzáféréstúl függően változnak: olvasáskor a minimális biztonsági szint nő fel az olvasott objektum minimális biztonsági szintjére, íráskor a maximális biztonsági szint csökken le az írásra megnyitott objektum biztonsági szintjére.

1.2.3 Biba integritási modellje[7]

Az esetek nagy részében az adatok bizalmassága kevés jelentőséggel bír, viszont a rendszer és adatainak integritása nagyon fontos szempont. Erre készült a Biba modell. Ez a modell kísértetiesen emlékeztet a Bell-LaPadula modellre, csak a biztonsági szinteket integritási szintekre, a $\geq$ operátort pedig $\leq$ -re kell cserélni. Tehát a Bell-LaPadula modell biztonsági szintjei fordított sorrendben értelmezve a Biba modell integritási szintjeivé válnak. Vagyis egy Bell-Lapadula implementáció fordított kiosztású címkékkel Biba implementációvá válik.

1.2.4 Low Water Mark modell[12]

Ez a modell is az adatok integritására összpontosít. A lényeg az hogy az objektumokhoz és szubjektumokhoz rendelt címke azok "tisztaságát" jelképezi. Minél kisebb a címke, az objektum annál "koszosabb". Az új egyedek "objektumok és szubjektumok" címkéjüket attól az egyedtől öröklik, amelyből származnak. Az objektumok címkéje nem változik, a szubjektumok címkéje pedig a legkisebb integritású olvasott egyed címkéje lesz. Írni csak ugyanolyan vagy kisebb címkéjű objektumba lehet. Ha egy szubjektum címkéje csökken, minden objektumot amit írásra tart nyitva és már nincs jogosultsága írni, "eldob".

1.2.5 Kínai fal modell[13]

A kínai fal modell arra a helyzetre keresi a megoldást, amikor pl egy konzultáns cég munkatársai több egymással versengő cégnek is végeznek munkát. Ilyenkor nem szabad hogy ugyanaz a munkatárs több, ugyanabban a szektorban tevékenykedő cég számára is dolgozzon. Ezt a kihívást *COI (Conflict Of Interest)* csoportok definiálásával oldja meg a modell. Minden szubjektumhoz tartozik egy nelemű cimke, amely minden COI csoporthoz maximum egy elemet tartalmazhat. Egy címkézett objektumhoz akkor lehet hozzáférni olvasásra, ha a szubjektum címkéjében az objektum cimkéi közül minden COI classnál ugyanaz, vagy üres címke van. Ha egy szubjektum hozzáfért olvasásra egy objektumhoz, a szubjektum címkéje a továbbiakban a két címke uniója lesz.

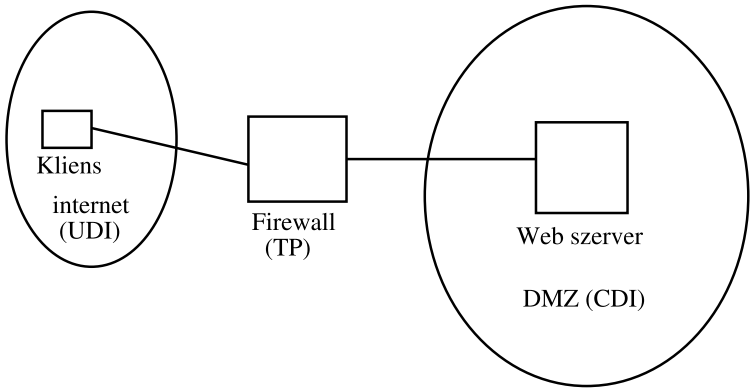
A szubjektum címkék két bejelentkezés között megörződnek.

1.2.6 Clark-Wilson modell[16]

A Clark-Wilson modell lényege, hogy *a védett adatokhoz csak védett procedúrákon keresztül lehet hozzáférni*.

Vannak *védett adatok (Constrained Data Item; CDI)* és *nem védett adatok (Unconstrained Data Item; UDI)* UDI-ból vagy a CDI-k egy osztályából egy másik CDI osztályba való transzformációt *Transzformációs Procedúrák (Transformation Procedure; TP)* végzik. A TPknek igazoltaknak kell lenniük ahhoz hogy a tranzakciót elvégezhessék. Az operációs rendszernek biztosítani kell, hogy a CDI-k csak az azokat manipuláló TP-ken keresztül legyenek elérhetőek, csak azok számára a felhasználók számára, akik jogosultak a TP-t, és azon keresztül a CDI-t elérni.

A Clark-Wilson modellre példa lehet az alábbi ábrán látható felépítés:



A fenti ábrán egy szokásos elrendezés látható. Egy web szerver amit tűzfallal védünk a *csúnya gonosz crackerek* ellen. Itt az internet, az azon megtalálható adatok, illetve az internetről jövő kérések UDI-nak számítanak. A DMZ-ben lévő adatok, az oda bekerülő http kérések pedig CDI-nek. A kérések helyes formájára hozásáért a firewallon futó http proxy felelős, ami a modellben a TP szerepét tölti be.

A példából is nyilvánvaló a Clark-Wilson modell nagy hiányossága; semmilyen útmutatást nem ad arra, hogy a Trusted procedure mitől lesz Trusted. Amint a tűzfalpiac ismerőinek az nyilvánvaló, a kereskedelmi tűzfalak közül szinte nincs is olyan, amivel lehetséges lenne a http kérések olyan szintű szűrése ami egy web szerver védelmét ilyen szituációban megfelelő módon el tudná látni.

1.2.7 Privacy Modell[14]

A privacy modell egy kicsit hasonlít a Bell-LaPadula DoD kiegészítéséhez, de az integritási címkéket teljesen másra használja, nevezzük hát őket privacy címkéknek:

Ezek a címkék egy-egy felhasználási körét írják le az adatoknak. A felhasználóknak van egy engedélyezett halmaza a privacy labelekből, de egyszerre csak egy ilyen címkét használhat. Hogy éppen melyiket, egy TP-vel adhatja meg. A dolog lényege a privát információ megőrzése.

Nézzük egy példán a dolog lényegét: Van egy kórház, ahol a betegek személyes adatait őrzik. Ezek három kategóriába tartoznak: pénzügyi adat, a kezeléshez használható adat, kutatáshoz használható adat. A betegek megmondhatják, hogy az adataikat milyen célokra lehet használni. Tegyük fel hogy egy orvos kutatni akar: beállítja, hogy ő most kutat, és ekkor csak a kutatásra használható adatokhoz férhet hozzá. Ha gyógyítani akar, akkor meg csak a gyógyításhoz használhatóakhoz.

1.2.8 Háló modellek[5][8]

Az eddig ismertetett modelleknek, a Clark-Wilson modell kivételével közös tulajdonsága volt az, hogy ezek háló modellek, vagy némi matematikai trükközés után háló modellekké alakíthatóak. A háló modellek a következő szabályoknak felelnek meg:

- A domainek halmaza véges
- Ezen a halmazon létezik egy részleges rendezési operátor
- A domainek halmazának van alsó határa a rendezési operátorra nézve
- Létezik egy "osztálykombinációs" operátor, ami azt mondja meg hogy két különböző domainba tartozó adat kombinálásakor mi lesz a kombinált adat domainje. Ez az operátor egy teljesen definiált legkisebb felső határ operátor.

Az ebből levonható tanulság pedig az, hogy elméletileg bármely modell használható bármely másik modell emulálására. A hangsúly az elméletileg kitételen van.

1.2.9 Type enforcement[3]

A kötelező hozzáférésvezérlési modellek egy jó általános esetének tűnik a Type Enforcement (TE) modell. Ez a modell annyit mond, hogy a szubjektumokhoz domaint, az objektumokhoz típust rendelünk, és a konfigurációból kiderül hogy a domainek hogyan érhetik el a típusokat, illetve más domineket. Ezen kívül automatikos domain-átmeneteket lehet előírni. Ez a modell megfelelően általános, de az implementálásához szükség van a döntési réteg különválasztására. Az eddigi eredményekből úgy tűnik, hogy ezzel a modellel a gyakorlatban is lehet implementálni a többi modellt.

1.2.10 Role Based Access Control (RBAC)[4]

A role-based access control lényege annyi, hogy meghatározza egy szubjektum számára, hogy mely domainekbe hajthat végre egyáltalán átmenetet.

1.3 Hálózati alkalmazás

A fenti modellek közül hálózati alkalmazásért leginkább a Clark-Wilson és a Bell-LaPadula modell kiált.

A Clark-Wilson modell baja -mint azt láthattuk- az hogy a TP-k kívánatos tulajdonságairól annyit tud mondani, hogy biztosítsák azt hogy a CDI-k formája megfelelő legyen. A TP-k minőségének biztosítása implementációs részlet. A hálózati határvédelem által ebből leszűrhető tanulság annyi, hogy biztosítsuk az eszközöket arra hogy a hálózati forgalom minden protokollelemét flexibilisen kezelni lehessen. Ezt a gyakorlatban megfelelően flexibilis, moduláris proxy architektúrával és a döntési réteg szétválasztásával lehet elérni. Ilyen tulajdonságokkal a piacon jelenleg *egyetlen tűzalszoftver, a Zorp[2]* rendelkezik.

A Bell-Lapadula modell hálózati alkalmazhatóságának vizsgálatánál az első szembeötlő probléma az, hogy egy távoli eljáráshívást (itt nem kell bonyolult dolgokra gondolni, pl az FTP protokoll DIR parancsa is az), illetve a státusz visszaadását minek tekintsük. A tranzakció nyilvánvalóan adatáramlással jár, de normális körülmények között a cél nyilvánvalóan nem az adatáramlás. A tranzakcióval kapcsolatos aggályok sem elsősorban a bizalmassággal, hanem az integritással kapcsolatosak. Ezen túl ha adatáramlásnak tekintjük az ilyen jellegű tranzakciókat, akkor azt vesszük észre hogy ezek mindig egy írásból és utána egy olvasásból állanak. Amennyiben a két fél biztonsági szintje nem ugyanaz, legalább az egyik művelet tilos lesz, tehát semmiféle magasabb szintű kommunikáció nem képzelhető el.

A megoldás az, hogy az írás és olvasás művelet mellett bevezetjük a *vezérlés (control)* és *visszajelzés (feedback)* műveleteket. Az írásra és az olvasásra a Bell-LaPadula modell szabályait, a vezérlésre és a visszajelzésre pedig a Biba modell szabályait alkalmazzuk, a következőképpen:

- írás: $L_S1 \leq L_S2$

- olvasás: $L_S1 \geq L_S2$
- vezérlés: $L_S1 \geq L_S2$
- visszajelzés: $L_S1 \leq L_S2$
- És természetesen $C_S1$nek részhalmaza C_S2.

Ahol L a biztonsági szintet, C a nem-hierarchikus kategóriákat, S1 a kliens, S2 a szerver oldalt jelenti.

A fenti modell nem mond semmit arról az esetről, amikor a vezérlés és visszajelzés műveletet használják félre információ kiszivárogtatásához. Az olyan csatornát, amin ezt meg lehet tenni *fedett csatornának (covert channel)*[15] nevezzük. A fedett csatornák minimalizálása egy fontos, ám igen nehéz feladat. Ha azt vizsgáljuk, hogy milyen technikákat és eszközöket lehet erre a célra használni, rá fogunk jönni, hogy a fedett csatornától mentesség tulajdonképpen gyakorlati szempontból is a hálózati protokoll jólformált viselkedésének a kritériuma. A Clark-Wilson modell hálózatvédelmi kihatásait pedig föntebb tárgyaltuk.

1.3.1 A példa

Mielőtt belemennénk a példa értelmezésébe, tekintsünk át néhány hálózati határvédelemmel illetve a Zorp tűzfal működésével kapcsolatos alapfogalmat:

Zóna A zóna tulajdonképpen hálózati erőforrások halmaza. A hálózati erőforrásukat címük azonosítja (tt lehet akár IP címre, akár URL-re, akár hálózatcímre gondolni netmaszkostul). Azonos zónába azokat az erőforrásokat soroljuk, amelyeket valamilyen szempontból azonosan kívánunk kezelni. A kezelés módját az objektum biztonsági attribútumai, címkéi határozzák meg. Tehát a zóna lényege az, hogy segítségével a hálózati erőforrások által képviselt objektumok és szubjektumok, valamint a címkék közötti összerendelést tudjuk meghatározni. Kényelmi és menedzselhetőségi szempontból a zónákat egy fába rendezhetjük, ahol a címkék öröklődésének módját öröklődési szabályok határozzák meg.

Service A service a tűzfalszoftver egy olyan entitása, amely a hálózati forgalom valamely részhalmazát ellenőrzi. Vagyis a kliens és a szerver közé beékelődvén elvégzi a hozzáférésvezérlés és a forgalom jólformáltságának ellenőrzése című feladatot.

Listener A listener a tűzfalszoftver olyan entitása, amely valamely hálózati forgalomra figyel. A listener dönt arról hogy a hálózati kapcsolatot melyik servie-nek adja át.

Chainer A chainer a tűzfalszoftver azon entitása, amely arról dönt, hogy a kapcsolat tulajdonképpen hol is végződjön. A tűzfal konfigurációjától függően ugyanis egy adott kapcsolat gyakran nem arra a címre irányul a valóságban, amiről a kliens tud. Ennek biztonsági, kényelmi és címgazdálkodási okai egyaránt vannak.

Miután ezeket jól áttekintettük, lássuk a konfigot. A következő konfigról tudni kell, hogy a 0.5-ös Zorp alaphoz készült, és bevezet illetve átdefiniál néhány modult. Az érdekesség kedvéért viszont került bele néhány proxy definíció, ami a 0.7-es alapot feltételezi. A vonatkozó modulok forrása a publikációhoz tartozó examples könyvtárban van.

Fontos azt is tudni, hogy ez deszkamodell-implementáció, valami ehhez hasonló vagy ettől teljesen különböző a Zorp valamely későbbi verziójában várható, illetve ebből kiindulva bárki összeütheti a sajátját.

A modell céljai egy új zónaabsztrakció és a fönnt említett MAC modell kipróbálása volt.

Tehát lássuk:

Először beolvassuk a szükséges modulokat. A saját dolgaink a local.proxies modulban lesznek

```
from Zorp.Chainer import TransparentChainer
from Zorp.Zone import Zone,acladd
from Zorp.LabelRange import LabelRange
from Zorp.Label import Label
from Zorp.Glue import Instance
from local.proxies import *
```

Definiáljuk a biztonsági szinteket. A SecurityLevels lista a szintek sorrendjét jelöli.

```
#Hierarchical Security Levels
secret='secret'
topsecret='topsecret'
public='public'
confidental='confidental'
SecurityLevels=[secret, topsecret, confidental, public ]
```

A kategóriák

```
#Non-Hierarchical Categories
webeditor='Web editor role'
```

```
webadmin='OS administrator in the public web server'
user='Plain user. Can reach the internet.'
luser='External user. Can reach the web servers in DMZ.'
```

A zónák. Van internet,intranet és DMZ. Az intranetben van két, a DMZ-ben ‘ kitüntetett gép.

```
#Zones
#

#internet
Zone(name='internet',address='*://0.0.0.0/0//',
client_labels=LabelRange((public,[]),(public,[])),
labels=Label(public,[]),
adminparent='')
#the intranet
Zone(name='intranet',address='*://192.168.0.0/15//',
client_labels=LabelRange((confidental,[luser]),
(confidental,[user,luser])),
labels=Label(confidental,[user]),
adminparent='')
#demilitarized zone
Zone(name='dmz',address='http://192.168.128.0/24//',
client_labels=LabelRange((public,[]),(public,[])),
labels=Label(public,[luser]),
adminparent='')
#a special machine in the DMZ,
# which can be reached only by the administrational roles
Zone(name='dmz_main',address='*://192.168.128.2//',
labels=Label(public,[webadmin,webeditor]),
adminparent='dmz')
#the webserver's administrator's machine
Zone(name='intranet_webadmin',address='*://192.168.1.2//',
client_labels=LabelRange((confidental,[]),
(confidental,[webadmin])),
adminparent='intranet')
# the web editor's machine
Zone(name='intranet_webeditor',address='*://192.168.2.1//',
client_labels=LabelRange((confidental,[]),
(confidental,[webeditor])),
adminparent='intranet')
```

Így fogjuk átmapelni a DMZ kívülről látható címeit a belső címekre. A MappingChainer a proxies modulban van.

```
def virtualipmap(ip):
        return (
  SockAddr.inet_ntoa((SockAddr.inet_aton(ip) & 0xff)
  + SockAddr.inet_aton('192.168.128.2')))
```

Itt vannak az instanciák

```
#Zorp instances
Instance(name='http_internet',proto=['http'],proxy=InterHttpProxy,
chainer=TransparentChainer, listenaddress = '192.168.0.1',
 usrport = 80, listenport = 3001, chainerparam = (None,1))

Instance(name='http_inter_dmz',proto=['http'],proxy=DMZHttpProxy,
chainer=MappingChainer, listenaddress = '1.2.3.4',
 usrport = 80, listenport = 3002, chainerparam = virtualipmap)

Instance(name='http_intra_dmz',proto=['http'],proxy=DMZHttpProxy,
chainer=MappingChainer, listenaddress = '192.168.0.1',
 usrport = 80, listenport = 3003, chainerparam = virtualipmap)

Instance(name='http_webedit',proto=['webedit'],proxy=EditorHttpProxy,
chainer=DirectedChainer, listenaddress = '192.168.0.1',
 usrport = 8080, listenport = 3004, chainerparam = (192.168.128.2,0))
```

És lássuk a proxies.py modult. Ez a modul elsőre kevésbé olvasmányosabb. Míg a fenti modul a Bell-Lapadula modellnek megfelelő konfiguációval foglalkozott, ez a Clark-Wilson-al kapcsolatos témákat járja körül. Ezért is van az olvasmányosság hiánya: rengeteg, a hálózati protokoll tudását igénylő dolgot lehetne és kellene beállítani. A lenti példák épp csak a felszínt borzolva adnak számot a lehetőségekről.

```
#we will give this function to the mapping chainer
def virtualipmap(ip):
        return (
  SockAddr.inet_ntoa((SockAddr.inet_aton(ip) & 0xff) +
  SockAddr.inet_aton('192.168.128.2')))

#Okay, one possible implementation of the mapping chainer
```

```
class MappingChainer(AbstractChainer):
def __init__(self,function):
self.function=function
def connectServer(self,session,host,port):
local=session.client_address
origremote=getTransparentDestination(session.client_fd)
remote=self.function(origremote)
return self.establishConnection(session,local,remote)


#
# A http proxy, which
# -rewrites our user agent
# -denies URLs which match with the regexes in DenyRegex
# and does not match with those in AllowRegex
# in the real world we would read them from a file
# -drops cookies
#

DenyRegex=['sex','porno']
AllowRegex=['sextant','sexton']

def compileregexlist(list):
ret=[]
for i in list:
ret.append(re.compile(i))

Deny=compileregexlist(DenyRegex)
Allow=compileregexlist(AllowRegex)

class InterHttpProxy(HttpProxy):

        def config(self):
         HttpProxy.config(self)
                self.transparent_mode = TRUE
                self.request_headers["User-Agent"] =
(Http.HTTP_CHANGE_VALUE, "Lynx/2.8.3rel.1")
self.request_headers["Cookie"] = (HTTP_DROP)
self.response_headers["Set-Cookie"] = (HTTP_DROP)
                self.request["GET"] = \
(Http.HTTP_POLICY, self.filterURL)
```

```
                self.request["HEAD"] = \
(Http.HTTP_POLICY, self.filterURL)
                self.request["POST"] = \
(Http.HTTP_POLICY, self.filterURL)

def filterURL(self, method, url, version):
        log("http.info", 3, "%s: %s: %s" %
(self.session.session_id,method, url))
for rexd in Deny:
if rexd.search(url):
for rexa in Allow:
if rexa.search(url):
return Z_ACCEPT
self.error_info = \
"This URL is in our deny list"
return Z_REJECT
return Z_ACCEPT


#
#
# This proxy protects the web servers in the DMZ
#  -Drops all request headers
# -lowercase URL
#  -filters URLs to only contain [a-z/\?\.] and do not contain ..
# -filters URLs for maximum length of 256 bytes
# -denies POST requests
#
dotdot=re.compile('\.\.')
allowedchars=re.compile('^[a-z/\?\.]*$')
class DMZHttpProxy(HttpProxy):

        def config(self):
         HttpProxy.config(self)
                self.transparent_mode = FALSE
self.request_headers["*"] = (HTTP_DROP)
                self.request["GET"] = \
(Http.HTTP_POLICY, self.filterURL)
                self.request["HEAD"] = \
(Http.HTTP_POLICY, self.filterURL)
                self.request["POST"] = (HTTP_DROP)
```

```
def filterURL(self, method, url, version):
        log("http.info", 3, "%s: GET: %s" % (
self.session.session_id, url))
if len(url)>256:
return Z_REJECT
url=string.lower(url)
self.request_url=url
if dotdot.search(url):
return Z_REJECT
if not allowedchars.search(url):
return Z_REJECT
return Z_ACCEPT

#
#
# Http proxy for the editor
#
# Same as the above, but enables POST

class EditorHttpProxy(DMZHttpProxy):
def config(self):
EditorHttpProxy.config(self)
self.request["POST"] = (Http.HTTP_POLICY, self.filterURL)
```

Hivatkozások

[1] Trusted unix working group (trusix) rationale for selecting access control list features for the unix (r) system (http://www.radium.ncsc.mil/tpep/library/rainbow/NCSC-TG-020-A.txt)

[2] Zorp homepage (http://www.balabit.hu/zorp/)

[3] Badger, et al. "Practical Domain and Type Enforcement for UNIX" 1995 IEEE Conference on Security and Privacy (http://www.nai.com/nai_labs/asp_set/environments/dte_acccon.asp)

[4] R. S. Sandhu, et al. "Role-Based Access Control Models", IEEE Computer 29(2): 38-47, IEEE Press, 1996.(http://citeseer.nj.nec.com/barkley97role.html)

[5] Ravi S. Sandhu. Lattice-based access control models. IEEE Computer, 26(11):9–19, November 1993 (http://heavenly.nj.nec.com/did/87719)

[6] Bell, D.E. and LaPadula, L.J. "Secure Computer Systems: Mathematical Foundations and Model." M74-244, Mitre Corporation, Bedford, Massachusetts (1975). (Also available through National Technical Information Service, Springfield, Va., NTIS AD771543.)

[7] Biba, K.J. "Integrity Considerations for Secure Computer Systems." Mitre TR-3153, Mitre Corporation, Bedford, Massachusetts, (1977). (Also available through National Technical Information Service, Springfield, Va., NTIS AD-A039324.)

[8] Denning, D.E. "A Lattice Model of Secure Information Flow." Communications of ACM 19(5):236-243 (1976).

[9] DoD Trusted Computer System Evaluation Criteria, 26 December 1985 (Supercedes CSC-STD-001-83, dtd 15 Aug 83). (http://www.radium.ncsc.mil/tpep/library/rainbow/5200.28-STD.html)

[10] Common Criteria (http://www.radium.ncsc.mil/tpep/library/ccitse/ccitse.html)

[11] Labeled Security Protection Profile (Version 1.b) (http://www.radium.ncsc.mil/tpep/library/protection_profiles/LSPP-1.b.pdf)

[12] LOMAC: Low Water-Mark Integrity Protection for COTS Environments, Timothy Fraser NAI Labs tfraser@nai.com 3060 Washington Road Glenwood, MD 21738, USA (ftp://ftp.tislabs.com/pub/lomac/)

[13] Brewer, D.F.C and Nash, M.J. "The Chinese Wall Security Policy." Proceedings IEEE Symposium on Security and Privacy, 215-228 (1989).

[14] Simone Fischer-Hübner, Amon Ott: "From a Formal Privacy Model to its Implementation" for the National Information Systems Security Conference (NISSC 98) (http://www.rsbac.de/niss98.htm)

[15] Covert Channels — Here to Stay? (1994) Reprint Department Navy Naval Research Laboratory Ira S. Moskowitz Myong H. Kang (http://citeseer.nj.nec.com/moskowitz94covert.html)

[16] D. C. Clark and D. R. Wilson, "A comparison of Commercial and Military Computer Security Policies", IEEE Symposium on Security and Privacy, 1987.