Y36SPS – Semestrální práce

Linux based webhosting

Jan Matoušek
matouj3@fel.cvut.cz

Obsah

1. Úvod

Tato práce je postupem pro vytvoření webhostingového serveru pro poskytování virtuálních webserverů a na něj návazných služeb. Popsaný postup je funkční na kolejním serveru na Sinkuleho koleji v Dejvicích (http://web.sin.cvut.cz). Hlavní výhodou je autentizace a autorizace uživatelů uložených v mysql databázi pomocí modulu PAM. Požadovaným výstupem má být registrace uživatele přes webový formulář, výběr poskytovaných služeb a vytvoření jailového virtuálního webhostingu uzitatel.web.sin.cvut.cz (také web.sin.cvut.cz/~uzivatel), ssh přístupu k filesystému na tento webhosting, a databáze. Součástí této práce sice není webová aplikace napsaná mnou v PHP, která se stará o management účtů (z důvodu nekompletnosti), jsou zde popsány funkční požadavky, vstupy a výstupy takové aplikace.

2. Prerekvizity

Seznam potřebných balíků (pro distribuci Debian Lenny)

- libpam-mysql
- libnss-mysql
- libpam-runtime
- libpam-modules
- libpam0g
- mysql-client-5.0
- mysql-common
- mysql-server-5.0
- php5-mysql
- libnss-mysql
- apache2.2-common
- apache2-mpm-prefork
- apache2-utils
- libapache2-mod-php5
- openssh-server
- openssh-client
- phpmyadmin

3. Nastavení

1. PAM a NSS

Autorem této části je **Spencer Stirling**, uvádím český výtah a své postřehy. Adresa je zde:

http://www.spencerstirling.com/computergeek/mysqluser.html

PAM (Pluggable Authentication Module) slouží pro autentizaci (login uživatele), na něj navazuje NSS (Name Switch System) s autorizací (přidělení uživateli práv do systému). Protože cílem je uskladnit uživatele v databázi vytvoříme databázi (sql kód Spencera Stirlinga lze nalézt v dodatku na konci dokumentu). Upozorňuji, nikdy neskladujte v databázi roota! V případě, že by vypadla databáze se pak nedostanete do systému.

MySQL

Databáze obsahuje 3 tabulky: uživatelé, skupiny a uživatelé_skupiny, které jsou náhradou za /etc/passwd, /etc/groups a /etc/shadow.

Zároveň byl vytvořen mysql uživatel "nss" a "nss-shadow", s právy pro čtení a případně pro změny, je třeba si u něj nastavit heslo, které pak bude v nešifrované (plaintext) podobě uvedeno v některých souborech /etc/pam.d/common-* a /etc/nss-mysql*.conf

NSS

Nyní přidáme na každý řádek v /etc/nsswitch.conf "mysql". Tímto budeme směrovat přihlašovací requesty jak do souborů /etc/passwd, tak pomocí automatického mysql dotazu do mysql databáze.

```
passwd: compat mysql
group: compat mysql
shadow: compat mysql
```

Ještě je nutné nastavit přihlašovací údaje k databázi (uživatel nss a nss-shadow) v konfiguračním souboru modulů nss-mysql v /etc/nss-mysql.conf (uživatel nss) a /etc/nss-mysq-root.conf (uživatel nss-shadow).

Tyto soubory jsou prakticky nakonfigurované, stačí jen doplnit následující položky, zbytek souboru je velmi dobře okomentován.

```
shadow.db_user = nss;                # "nss-shadow" v nss-mysql-root.conf
shadow.db_password = tajneheslo;
```

Soubor nss-mysql-root.conf nastavíme čtení pouze rootovi , nss-mysql.conf rootovi a skupině.

```
chmod 0600 nss-mysql.conf          # pro root:shadow
chmod 0640 nss-mysql-root.conf  # pro  root:shadow
```

PAM

Nyní nastavíme tytéž přihlašovací údaje v /etc/pam.d/ pro soubory

common-auth:

```
auth   sufficient      pam_mysql.so verbose=2 user=nss passwd=heslo host=localhost
db=nss_mysql table=users usercolumn=user.user_name passwdcolumn=user.password crypt=0
sqllog=1 logtable=logs logmsgcolumn=msg logusercolumn=user logpidcolumn=pid
loghostcolumn=host logrhostcolumn=rhost logtimecolumn=logtime

auth    required        pam_unix.so nullok_secure
```

common-account:

```
account sufficient     pam_mysql.so verbose=2 user=nss passwd=heslo host=localhost
db=nss_mysql table=users usercolumn=user.user_name passwdcolumn=user.password crypt=0
sqllog=1 logtable=logs logmsgcolumn=msg logusercolumn=user logpidcolumn=pid
loghostcolumn=host logrhostcolumn=rhost logtimecolumn=logtime

account required       pam_unix.so
```

Autor zmiňuje jako možnost pro některé releasy common-session a common-passwd, v našem případě není potřeba. Pokud máte v plánu měnit tyto konfigurační soubory, pak velmi důrazně doporučuji dobře nastudovat stupňování optional/sufficient/required/... může se stát, že se při špatném nastavení budou moct uživatelé přihlašovat I bez hesla.

Přihlašování lze obohatit I o další možnosti jako je logování (viz kód nahoře napsaný italic písmem), tento kód přidávám jako bonus, je však nutné si přidat tabulku log s příslušnými sloupečky.

Nakonec nastavéme tyto soubory pro čtení pouze rootovi

```
chmod 0600 common-*       # pro root:root
```

Pro každou službu, ve které chcete využívat autentizaci a autorizaci uživatelů v databázi je třeba v /etc/pam.d/* includovat soubory common-*. V Debianu je to již defaultně nastavené pro většinu služeb.

```
@include common-auth

@include common-account
```

2. Jail

Vzhledem k množství bezpečnostních rizik, které vyplývají z plaintextového uskladňování hesel v konfiguračních souborech v /etc/ a celkovému zlepšení bezpečnosti systému je značně na místě izolovat uživatele od zbytku systému v chrootovaném jailu. Jail se jeví uživateli jako klasický root filesystem s /etc, /bin atd.. ačkoliv se jedná pouze o falešné adresáře s úzkým výběrem binárních souborů, knihoven a konfigurací.

V některých distribucích existuje sofistikovaný software pro tvorbu jailů, bohužel Debian Lenny nimi příliš neoplývá a poskytuje pouze základní část umožňující chrootovaný shell pro vybrané uživatele. (chrootovaný = uživatel vidí jako "/" root adresář, který je ve skutečnosti například "/home/uzivatel/" a ten má svoji strukturu /etc, /bin, /home, ...). O výběr softwaru jeho kopírování včetně závislostí se pak stará administrátor.

Typy Jailu

Jail lze postavit na mnoho způsobů – např. 1 jail, se všemi uživateli =

```
/home/jail/home/

/home/jail/bin/

/home/jail/home/petr/

/home/jail/home/pavel/
```

V tomto návodu se dále budeme zabývat tímto typem jailu. Jeho výhody jsou malé systémové

nároky. Nevýhody menší bezpečnost mezi uživateli (riziko, že si uživatelé budou moct vykrádat home), další nevýhodou je, že nemají uživatelé plný root přístup ke všem částem jailu.

Dalším typem je jail pro každého uživatele zvlášť.

```
/home/petr/bin/
/home/petr/home/petr/
/home/pavel/bin/
/home/pavel/home/pavel/
```

Jeho výhodou je velká indipencence uživatelů, bezpečnost, možnost root přístupu do jailovaného linuxu dokonce až možnost instalace vlastního softwaru přímo z repozitáře. Nevýhodou jsou vysoké systémové nároky.

Zřízení Jailu

Pro zřízení námi preferovaného jailu je třeba si zvolit adresář, který jako jail poslouží a nakopírovat do něj vybraný software včetně všech závislostí a knihoven. Pro vytvoření nejzákladnější kostry jailu a manipulace s jail uživateli se dá použít existující skript od **Wolfganga Fuschlbergera**, adresa skriptu je zde:

```
http://tuxtraining.com/files/make_chroot_jail.sh.html
```

nicméně tento skript je však určen pro případ, že nemáme uživatele v databázi nýbrž v /etc/passwd, oprava není nijak složitá. Skript se dá sám o sobě použít pro vybudování kostry Jailu a případně upravit pro přidávání vygenerovaných řádků do /etc/passwd, groups a shadow.

Tento skript vytvoří nový shell **/bin/chroot-shell**, ten přes sudo volá */usr/sbin/chroot* na složku s jailem. Je nutné tedy pro každého uživatele přidaného do databáze nastavit jako defaultní shell místo */bin/sh* shell */bin/chroot-shell* a zapsat následující řádek do **/etc/sudoers**, který jej opravňuje používat chroot.

```
franta      ALL=NOPASSWD: /usr/sbin/chroot, /bin/su - franta
```

Protože v našem případě se hlavní registrace děje v prostředí PHP, je nutné generovat jak uživatele franta, tak skupinu franta (nejlépe se stejným ID → GID = UID), a zároveň umístit uživatele franta jak do skupiny franta tak do skupiny users a to celé vložit jak do databáze, tak pomocí make_jail.sh do vnitřního prostředí jailu.

```
./make_jail.sh $name $group $passwd $shadow
```

Kde $name je jméno uživatele, $group je celý řádek do jail/etc/group, $passwd je řádek do jail/etc/passwd a $shadow je řádek do jail/etc/shadow. Nestačí totiž skladovat uživatele jen v databázi, protože uvnitř jailu to uživatel nepozná. Chrootovaný jail shell totiž potřebuje UID, GID a další informace pro úspěšné přihlášení a fungování uživatele.

Přidání softwaru do Jailu

Chcete-li přidat software do jailu, tak můžete narazit na problém. Po delším googlení jsem došel názoru, že neexistuje žádný standardizovaný a zaručený způsob jak to udělat. Proto jsem si napsal

svůj vlastní ne příliš elegantní, avšak ve většině případů funkční skriptík “addbin.sh”, jehož kód se nachází v dodatcích této práce. Je založen na příkazu *ldd*, který pro každý binární program vypisuje závislé knihovny a ty pak addbin.sh překopíruje do jailu. Je nutné pak ručně zkopírovat nastavení z /etc do jail/etc. Bohužel však ne vždy *ldd* vypíše kompletně všechny závislosti.

Skript se jednoduše umístí do rootu jailu (/home/jail/) zavolá jako “./addbin.sh /usr/bin/bc”, automaticky se pak zkopíruje /usr/bin/bc do jail/usr/bin/bc včetně závislých knihoven.

3. PHP

Administrační rozhraní

Jelikož cílem této práce je automatická tvorba webhostingu nejlépe ve formě registračního formuláře přes webový prohlížeč, tak tento požadavek stanovuje umístit největší podíl skriptovací části celého projektu právě do skriptů volaných přes web. Na výběr je prakticky cokoliv od JSP, Ruby, PHP, CGI, ... Mé osobě je nejbližší PHP, takže výsledkem je administrační rozhraní pro správu uživatelů, uživatelských skupin, logů a registrační rozhraní napsané v architektuře MVC (model – view – controller). Toto rozhraní volá ostatní bashové skripty pro ovládání zbytku systému.

Jelikož se mé PHP rozhraní nachází stále ve vývojové fázi a jeho kód není předmětem této práce protože nevysvětluje princip tvorby webhostingu, ale byl by předmětem tvorby webové aplikace, tak kód není přiložen k tomuto dokumentu.

Bezpečnost PHP prezentací

Je nutné poznamenat, že v námi zvoleném případě se uživatelské webové prezentace nacházejí v jail/home/franta/www a tuto prezentaci je možné skriptovat pouze v PHP, protože se však nachází všechny účty v jednom jailu v jednom homu a PHP umožňuje defaultně spouštět shellové skripty jako například

```
shell_exec(“less ../pepa/conf.db”);
```

a vyčíst tak jeho heslo do databáze. Tomuto jevu se nedá jednoduše zabránit pomocí zmeny práv, protože php skripty musí být čitelné pro skupinu www-data aby je mohl Apache servírovat ven. Tudíž onen *less ../pepa/conf.db* je spuštěn pod uživatelem www-data jak pro jail/home/franta tak pro jail/home/pepa.

Existují 2 řešení. Prví je zakázat v php.ini spouštění shellových skriptů přes shell_exec a jemu dalších asi 20 podobných příkazů (kompletní seznam se dá velmi jednoduše najít na googlu.). Druhým řešením je použití suPHP. Jedná se o PHP modul, který spouští PHP kompilaci jednotlivých uživatelských homů přímo pod účty jednotlivých uživatelů franta nebo pepa.

V našem případě není hosting využíván experty, kteří by toužili po spouštění shellových skriptů, proto bylo jednodušší zakázat onu sadu PHP příkazů v php.ini a až v případě, že si někdo zažádá o jejich aktivaci suPHP nainstalovat.

4. Apache

Zároveň při vkládání uživatele do databáze, tvorbě jeho home v jailu, zápisu jeho příslušných řádků v jail/etc/* se volá také skript pro tvorbu virtual hostingu v apache. Tento skript je volán z PHP a prakticky jen vygeneruje konfigurační soubor, jehož obsahem je adresa virtuálního serveru

a adresář, kde se nachází obsah uživatelova webu.

Skript se tvoří do /etc/apache/sites-enabled/franta.conf a jeho obsahem je:

```
<VirtualHost *:80>
    ServerName franta.web.sin.cvut.cz
    ServerAlias franta.sin.cvut.cz
    DocumentRoot /home/jail/home/franta/www

    <Directory /home/jail/home/franta/www>
        Options Indexes FollowSymLinks MultiViews
        AllowOverride All
        Order allow,deny
        allow from all
    </Directory>

    ErrorLog /var/log/apache2/user/franta_error.log
    CustomLog /var/log/apache2/user/franta_access.log combined
</VirtualHost>
```

Jelikož zde nepublikuji celý PHP kód, je zbytečné uvádět pouze jeho kousek, který tento konfigurák generuje. Nicméně je důležité uvést, že po odeslání registračního formuláře, uložení dat do databáze, vytvoření jail účtu a tohoto Apache konfiguráku se musí pro úspěšné dokončení restartovat server Apache. To bohužel z webu po odeslání formuláře nejde. Po delším googlení se nakonec ukázalo nejschůdnější nainstalovat Lighttpd server, rozjet ho na portu 81 a z něj pohodlně přes web přidávat uživatele do systému a zároveň restartovávat Apache, který se stará o samotný vitual hosting.

4.Závěr

Popsaný postup je nepochybně jedno z mnoha možných řešení webhostingu. Většinu zde uvedených informací jsem vygooglil. Pokud byl u informace uveden autor, tak jsem jej uvedl. Výsledné řešení nabízí spoustu možností pro rozšíření například o:

- automatickou tvorbu databáze a databázového účtu pro registrované uživatele
- možnost výběru z několika typů databází
- možnost výběru z několika typů skriptovacích webových jazyků
- možnost emailové chránky
- možnost ftp přístupu k souborům
- možnost omezovat a spravovat kvóty uživatelských účtů
- instalace suPHP pro bezpečnější jail mezi samotnými uživateli
- a mnoho dalších...

5. Dodatky

1. MySQL skript pro PAM

Skript pro vytvoření struktury tabulek v databázi včetně uživatelů a práv.

```
USE nss_mysql;
DROP TABLE IF EXISTS groups;
CREATE TABLE groups (
  group_id int(11) NOT NULL auto_increment primary key,
  group_name varchar(30) DEFAULT '' NOT NULL,
  status       char(1) DEFAULT 'A',
  group_password varchar(64) DEFAULT 'x' NOT NULL,
  gid int(11) NOT NULL
);
INSERT INTO groups VALUES (1,'users','A','x',100);
DROP TABLE IF EXISTS user;
CREATE TABLE user (
  user_id int(11) NOT NULL auto_increment primary key,
  user_name varchar(50) DEFAULT '' NOT NULL,
  realname varchar(32) DEFAULT '' NOT NULL,
  shell varchar(20) DEFAULT '/bin/sh' NOT NULL,
  password varchar(40) DEFAULT '' NOT NULL,
  status char(1) DEFAULT 'N' NOT NULL,
  uid int(11) NOT NULL,
  gid int(11) DEFAULT '65534' NOT NULL,
  homedir varchar(32) DEFAULT '/bin/sh' NOT NULL,
  lastchange varchar(50) NOT NULL default '',
  min int(11) NOT NULL default '0',
  max int(11) NOT NULL default '0',
  warn int(11) NOT NULL default '7',
  inact int(11) NOT NULL default '-1',
  expire int(11) NOT NULL default '-1'
);
DROP TABLE IF EXISTS user_group;
CREATE TABLE user_group (
  user_id int(11) DEFAULT '0' NOT NULL,
  group_id int(11) DEFAULT '0' NOT NULL
);
```

```
GRANT  select(user_name,user_id,uid,gid,realname,shell,homedir,status) on user to nss@localhost identi
GRANT  select(group_name,group_id,gid,group_password,status) on groups to nss@localhost identified by 0
GRANT  select(user_id,group_id) on user_group to nss@localhost identified by 'ieopurASDF';
GRANT select(user_name,password,user_id,uid,gid,realname,shell,homedir,status,lastchange,min,max,warn
GRANT update(user_name,password,user_id,uid,gid,realname,shell,homedir,status,lastchange,min,max,warn
FLUSH  PRIVILEGES;
```

2. addbin.sh

Skript pro přidávání softwaru do jailu včetně závislostí.

```
#!/bin/bash

if test $# -eq 0 ; then
echo usage: "${0##/*} [Dir name]"
exit
fi

recurse()
{
 echo "soubor: $1 "

 cp $1 `echo $1 | cut -c2-`  #zkopiruj binarku na prislusne misto,

 for file in `ldd $1 | cut -d' ' -f3` ; do
  echo "knihovna: $file "

  #kopiruj knihovny jednu po druhe
  if test -e $file ; then
   cp $file `echo $file | cut -c2-`
  fi
  recurse "$file"
  done
}

recurse `which $1`
```