Az SQL és az Nmap szövetsége: hatékonyság és kényelem

Ha az Nmap-et használjuk a hálózaton lévő gépek biztonságának ellenőrzésére, akkor érdemes a MySQL-re bíznunk a trendek és változások nyomon követését.

Levelet váltottam nemrég valakivel, akinek rendszeres kapupásztázást kell végeznie a hálózatán a sebezhető pontok felderítése végett. Az általa használt kapupásztázó segédprogram az ***Nmap***, ami azonban maga nem teszi könnyen kezelhetővé az általa szolgáltatott adatokat. Ez bizony már egy teljesen más terület. Néhány héttel később elkészült az ***Nmaphez*** egy olyan folt, amely lehetővé tette az eredmények ***MySQL*** adatbázisban történő rögzítését. Bár az ***Nmap*** támogatja a mind a géppel elemezhető, mind pedig az ***XML*** kimeneti formátumot, az ***SQL***-adatbázisba történő naplózás kényelme és hatékonysága messze meghaladja az ***XML***, de még a géppel elemezhető kimeneti formátum képességeit is. Hogy csak egy dolgot említsek, az ***nmapsql*** nem tartalmaz külön lépést a parancshéjban a kimenet utófeldolgozó egységbe való áttöltésére.
Az ***nmapsql*** egy közvetlen folt a ***Fyodor*** által megalkotott tiszteletre méltó ***Nmap*** hálózatpásztázó program 3.48-as változatához. (Amikor ezt írom, már megjelent az ***Nmap 3.50***-es változata, amelyhez egy frissített ***nmapsql*** is letölthető a program honlapjáról.) A folt tartalmaz ***MySQL***-támogatást, de az eredmények egyszerű átvitelén kívül sokkal többre is képes: célmegjelölést, pásztázó-kijelölést, és egyszerű összehasonlító kiértékelést (trending) is lehetővé tesz. Az adatok ***SQL***-adatbázisba való áttöltésével egy sereg új feladat elvégezhető. Az ***nmapsql*** a ➲ ***sourceforge.net/projects/nmapsql*** címről tölthető le. A program az adatkezeléshez jelenleg a ***MySQL*** ügyfélprogramjának felhasználói felületét alkalmazza.
Mivel a hálózati biztonságért felelős rendszergazdák nem feltétlenül adatzsonglőrök, az ***nmapsql*** tervezésekor fontos szempont volt az egyszerű kezelhetőség. A folt működése elég egyszerű ahhoz, hogy a hálózatvizsgálat eredményének legfontosabb információi egyetlen táblából kinyerhetőek legyenek. Az egyszerűség az oka annak is, hogy az IP-címek tárolása formázatlan szövegként és nem `inet_aton()` jelölésben történik. Ismerem a szövegkezelés teljesítményben megmutatkozó hátrányait, de most a kis adathalmazok kényelmes kezelésének bemutatására összpontosítunk. Nagy adathalmazok esetén, amikor a sebesség kérdése kritikus szempont, a célmegjelölő címkék, futási és pásztázó-azonosítók állnak rendelkezésre a numerikus keresés számára.

A cikkben először egy ***SQL***-t használó hálózatpásztázással foglalkozunk, amellyel egy hálózat nyitott kapuit és élő célgépeit derítjük fel. Ezután az ***SQL***-ben összegyűjtött adatokra vetünk egy pillantást és megvizsgáljuk, milyen módszerekkel lehet a kapott eredményeket összehasonlítani.

A beállítási lehetőségek

Az ***nmapsql*** tevékenységét az aktuális felhasználó saját könyvtárában lévő ***~/nmapsql.rc*** fájl beolvasásával kezdi. Ez azt jelenti, hogy ha az `nmapsql` futtatása előtt a `su -` parancsot használjuk a rendszergazdai jogosultság megszerzéséhez, a beolvasott fájl a ***/root/nmapsql.rc*** lesz. Egyelőre mindössze négy tétel szerepel az ***nmapsql.rc*** fájlban, mindegyik külön sorban, `adat=érték` formátumban, ahogy sok más programnál is láthatjuk. A tételek a következők:

```
server=localhost,
db=nmaplog,
user=nmap,
passwd=scanamanga
```

A `server` annak a gépnek a ***DNS***-neve, amelyen a ***MySQL*** fut, a `db` pedig ezen a kiszolgálón található adatbázis neve. A `user` és `password` tételek az adatbázishoz való csatlakozáshoz szükséges felhasználói név és jelszó. Az itt szereplő felhasználónak az adatbázison legalább `SELECT`, `INSERT` és `UPDATE` jogosultságokkal kell rendelkeznie.
Az ***Nmap*** által biztosított parancssori lehetőségeket az ***nmapsql*** négy további paraméterrel egészíti ki:

```
--mysql
--runid
--targetid
--scannerid
```

Ha az ***nmapsql*** futtatható állományát ezek nélkül a paraméterek nélkül indítjuk, a működése teljesen megegyezik az ***Nmap*** működésével. Ezen beállítások egyike sincs hatással az ***Nmap*** létező paramétereire, így semmi akadálya annak,

1. lista Az Nmap kimenetének egy részlete

```
Starting nmap 3.48 (
http://www.insecure.org/nmap/ )
at 2003-12-14 10:00 SGT
Insufficient responses for TCP sequencing (1),
OS detection may be less accurate
Interesting ports on 192.168.10.0:
(The 1656 ports scanned but not shown below are
in state: closed)
PORT    STATE SERVICE VERSION
80/tcp open  http?
Device type: print server
Running: Linksys embedded
OS details: Linksys EtherFast print server

Interesting ports on wap.hasnains.com
(192.168.10.1):
(The 1654 ports scanned but not shown below are
in state: closed)
PORT    STATE SERVICE VERSION
21/tcp open  ftp?
23/tcp open  telnet?
80/tcp open  http
Device type: broadband router
Running: Zyxel ZyNOS
OS details: ZyXEL Prestige 700/Netgear MA314
broadband router
```

hogy ugyanannak a hálózatvizsgálatnak az eredményét egyidejűleg SQL-adatbázisba és gépileg elemezhető kimenetre irányítsuk.
A `--mysql` kapcsolót a többi ***nmapsql*** kapcsoló nélkül használva a parancssorban olyan ***MySQL***-naplózás indul el, amelynek során minden címke- és azonosító-kijelölés önműködően történik. Az összes többi ***nmapsql*** paraméter feltételezi a `--mysql` használatát. Az önműködő kijelölés során a folyamat veszi a megfelelő tábla legnagyobb értékét és egyesével halad az értékeken.
A pásztázó azonosító megadásának lehetőségét bevezető `--scanner-id xxx` paraméter, amelyben az `xxx` az azonosító értékét jelöli, olyan helyzetek esetében alkalmazhatók, amikor több pásztázó kerül üzembe helyezésre, például egy több alhálózattal rendelkező környezetben. A pásztázó-azonosító (`scanner ID`) és a futásidejű azonosító (`runtime ID`) a `portstat` táblában tárolódik lehetővé téve a pásztázást végző gép számára az eredményhalmazok különválasztását. Az alábbi lekérdezés használatával így egyszerűen különválaszthatnánk például a tízes pásztázó eredményeit:

```
mysql> select * from portstat
    -> where scannerid = 10 and runid = 100;
```

A `--run-id xxx` paraméter az ***nmapsql*** adott futtatásához rendelt azonosító megadására szolgál. Amennyiben nem használjuk ezt a lehetőséget, a rendszer önműködően hozzárendelt azonosítót használ. Ha a megadott futtatás-azonosító már szerepel az adatbázisban, ismételten felhasználásra kerül. Ez a lehetőség biztosítja, hogy különböző futtatások eredményeit kényelmesen csoportosíthassuk egy futtatási azonosító alatt.
A futtatás-azonosító (`runtime ID`) és a hozzá kapcsolódó információk tárolására a `runlist` tábla biztosít helyet. A használt táblák ismertetését „Az nmaplog által használt táblák” című kiemelt részben láthatjuk. A vizsgálat befejeztével frissül a futásidejű információk egy része, köztük a parancssorban megadott összes lehetséges célpontok és a működő állapotban találtak száma. Hasonlóan tárolódik a pásztázó azonosítója (`scanner ID`) és az ehhez kapcsolódó információk a scanners táblában.
Minden vizsgált célpont egy címkét kap, az információ pedig a `targets` táblába kerül. A futási listához hasonlóan a `targets` tábla is két szakaszban kerül feltöltésre. Az első szakasz az IP-címeket és – amennyiben feloldható – a gépneveket gyűjti be, a második szakaszban pedig az `os_guessed` oszlop feltöltése történik. Pillanatnyilag nem tárolódik egy fel nem ismert operációs rendszer ujjlenyomat-információja, de a jövőben változhat a helyzet.
A `targets` táblában sosem jönnek létre kettőződések. Tapasztalatom szerint csak abban az esetben lehetnek kettőzött IP-alhálózatok, amennyiben egyik ügyféltől a másikhoz helyezzük át a rendszert. Ebben az esetben minden ügyfél esetén különböző adatbázisra van szükség.
A célazonosítókat (`target ID`) pillanatnyilag még nem használja a rendszer, de a felhasználónak a parancssorban lehetősége van saját célazonosító meghatározására. Amennyiben a megadott azonosító már létezik, az egyediség megőrzése érdekében a rendszer figyelmen kívül hagyja és egy rendszer által előállított azonosítót használ helyette.
Ha a parancssorban a `--target-id` után megadott célazonosító még nem szerepel a `targets` táblában, hozzárendelődik az adott cél IP-címéhez. Ha több rendszer számára történik a célmegadás, az első cél kapja meg a megadott célazonosítót, a rá következőkhöz pedig az eggyel növekvő azonosítók fognak tartozni.

Az alapok

Az ***nmapsql*** rögzíti a naplóban a futtatás dátumát és időpontját, az ***Nmap***-et futtató felhasználó nevét, a futtató gép nevét, és a futtatáshoz rendelt azonosító számot. A két utóbbi adat lehetővé teszi, hogy az ***nmapsql***-t nagyobb rendszerekben is alkalmazzuk és alapot képeznek az egyes vizsgálatok összehasonlítására. A futtatás-azonosító (`runid`, ***runtime ID***) az adathalmazon belül mindig egyedi. Ha a megadott cél azonos, két vizsgálati eredmény között a futtatás-azonosító alapján tehetünk különbséget, lehetőség van azonban arra is, hogy több vizsgálati eredményt csoportosítsunk oly módon, hogy a pásztázáshoz a `--run-id` kapcsoló segítségével egyetlen futtatás-azonosítót adunk meg. Vizsgáljuk meg például az `nmapsql` alábbi indítását:

```
$ nmap -A --mysql --runid 100 192.168.10.1/24
```

A parancs a naplózási lehetőséget engedélyező `--mysql` kapcsolóval indítja el az ***Nmap***-et, a futtatás-azonosító értékének

2. lista Egy adott pásztázás eredménye

target_ip	d	t	port	protocol	state	runid
192.168.10.0	2003-12-14	10:00:37	80	tcp	open	100
192.168.10.1	2003-12-14	10:00:37	21	tcp	open	100
192.168.10.1	2003-12-14	10:00:37	23	tcp	open	100
192.168.10.1	2003-12-14	10:00:37	80	tcp	open	100
192.168.10.44	2003-12-14	10:00:37	22	tcp	open	100
192.168.10.44	2003-12-14	10:00:37	111	tcp	open	100
192.168.10.44	2003-12-14	10:00:37	3306	tcp	open	100
192.168.10.44	2003-12-14	10:00:37	6000	tcp	open	100
192.168.10.64	2003-12-14	10:00:37	135	tcp	open	100
192.168.10.64	2003-12-14	10:00:37	139	tcp	open	100
192.168.10.64	2003-12-14	10:00:37	445	tcp	open	100
192.168.10.64	2003-12-14	10:00:37	1024	tcp	open	100
192.168.10.64	2003-12-14	10:00:37	1025	tcp	open	100
192.168.10.64	2003-12-14	10:00:37	1031	tcp	open	100
192.168.10.64	2003-12-14	10:00:37	5000	tcp	open	100
192.168.10.64	2003-12-14	10:00:37	5101	tcp	open	100
192.168.10.64	2003-12-14	10:00:37	6000	tcp	open	100
192.168.10.65	2003-12-14	10:00:37	135	tcp	open	100
192.168.10.65	2003-12-14	10:00:37	139	tcp	open	100
192.168.10.65	2003-12-14	10:00:37	445	tcp	open	100
192.168.10.65	2003-12-14	10:00:37	1025	tcp	open	100
192.168.10.65	2003-12-14	10:00:37	5000	tcp	open	100

100-at ad meg és megvizsgálja a 192.168.10.1/24 hálózatot. Amennyiben ez az `nmapsql` első futtatása, egy összehasonlítási alap jön létre a hálózat számára, amelyhez a későbbi futtatások eredményeit hasonlíthatjuk. Az ***nmapsql*** emellett önműködően létrehoz a futtató gép számára egy bejegyzést, ebben az esetben a `192.168.10.44` értékkel, amelyet hozzárendel a `scanners` tábla egy `scanner_id` mezőjéhez. Az *1. listán* a futtatás konzolkimenetének részlete látható.

A cél meghatározása ebben az esetben a teljes C-osztályú alhálózatot jelenti. Az ***nmapsql*** önműködően egyedi azonosítókat rendel a hálózaton talált minden működő géphez, és további információkat is tárol a `hoststats` táblában. Önmagában már ez a tábla is kiinduló eszköz lehet egy kapupásztázás-összehasonlításhoz.

Vizsgáljuk meg röviden, hogy milyen adatok kerültek rögzítésre. Ehhez először is bejelentkezünk a ***MySQL*** ügyfélprogramjába és csatlakozunk az ***nmapsql.rc*** fájlban megadott adatbázishoz. Ezután a következő lekérdezést futtatjuk:

```
$ mysql nmaplog -p
mysql> select target_ip, d, t, port, protocol,
    -> state, runid from portstat
    -> order by target_ip, d, t ;
```

A lekérdezés a *2. listán* látható táblázatot adja eredményül, egy tetszetős listát, amelyben az adatok a cél IP-címe, a dátum és idő szerint rendezett sorrendben jelennek meg. Látható, hogy a `runid` oszlop tartalma minden esetben 100, ahogy azt a parancssorban megadtuk.

A következő lekérdezés futtatásával a *3. listán* látható, a `192.168.10.44` cím nyitott kapuira vonatkozó információt kapjuk:

```
mysql> select target_ip, d, t, port, protocol,
    -> state, runid from portstat
    -> where target_ip = ´192.168.10.44´
    -> order by d, t;
```

Ha a kapott négy sort hozzáillesztjük az 1. lista `192.168.10.44` címre vonatkozó szakaszához, azonnal felismerhetjük a köztük fennálló kapcsolatot. Látszik, hogy a `portstat` tábla önmagában is tartalmazza az ***Nmap*** által kapupásztázásra vonatkozóan szolgáltatott összes információt. Természetesen ha már több futtatási eredménnyel rendelkeznénk, a fenti lekérdezés a `192.168.10.44` címre vonatkozó minden adatot kilistázná.

Tegyük fel, hogy két nap, egy hét, esetleg egy hónap elteltével újra megvizsgáljuk a hálózatot és az eredményeket vizuálisan is össze szeretnénk hasonlítani. A `runlist` táblára vetett első pillantásra látható, hogy a két nappal későbbi vizsgálatnak a 102-es futtatás-azonosító felel meg. Ennek az információnak a felhasználásával már írhatjuk is a lekérdezésünket:

```
mysql> select target_ip, d,t,port, protocol,
    -> state from portstat
    -> where target_ip = ´192.168.10.44´
    -> and runid = 102 order by d,t,port;
```

3. lista Egy adott gép vizsgálatának eredményei

```
+-----------------+-------------+-----------+---------+-----------+---------+----------+
| target_ip       | d           | t         | port    | protocol  | state   | runid    |
+-----------------+-------------+-----------+---------+-----------+---------+----------+
| 192.168.10.44   | 2003-12-14  | 10:00:37  |     22  | tcp       | open    | 100      |
| 192.168.10.44   | 2003-12-14  | 10:00:37  |    111  | tcp       | open    | 100      |
| 192.168.10.44   | 2003-12-14  | 10:00:37  |   3306  | tcp       | open    | 100      |
| 192.168.10.44   | 2003-12-14  | 10:00:37  |   6000  | tcp       | open    | 100      |
+-----------------+-------------+-----------+---------+-----------+---------+----------+
```

4. lista A 192.168.10.44 cím eredményei egy két nappal később folytatott vizsgálat után

```
+--------------+------------+----------+---------+-----------+--------+
| target_ip    | d          | t        | port    | protocol  | state  |
+--------------+------------+----------+---------+-----------+--------+
| 192.168.10.44 | 2003-12-16 | 00:47:16 |     22  | tcp       | open   |
| 192.168.10.44 | 2003-12-16 | 00:47:16 |    111  | tcp       | open   |
| 192.168.10.44 | 2003-12-16 | 00:47:16 |   3306  | tcp       | open   |
| 192.168.10.44 | 2003-12-16 | 00:47:16 |   6000  | tcp       | open   |
+--------------+------------+----------+---------+-----------+--------+
```

Könnyen összehasonlíthatjuk a *4.* és *5. lista* eredményeit és kiszűrhetjük a különbségeket. Természetesen a két adathalmaz összehasonlítását egy program is elvégezheti, amely összefoglalja számunkra a különbségeket.
A korábban említett `hoststat` tábla minden működő gép információit tartalmazza. Az `open_ports` oszlop egyszerű összegzést nyújt a nyitott kapuk számáról. Ahhoz, hogy a célgépünk nyitott kapujának egy soros információit megkapjuk, a következő lekérdezést kell futtatnunk:

```
mysql> select ip,d,t,open_ports, ports_scanned,
    -> runid from hoststats where order by ip, d, t;
```

(Az `order by` záradékot a továbbfejlesztés érdekében írtuk a lekérdezéshez.) Az `open_ports` oszlopot a `date/time` és runid oszlopokkal együtt vizsgálva vázlatos képet kapunk a nyitott kapuk állapotának egy biztonyos időn belüli változásairól. A targets tábla az összes előforduló célgéppel kapcsolatos információt tartalmazza, IP-címenként egy-egy sort. Ez az egyetlen tábla, amelyben a gép neve (amennyiben meghatározható) és az ***Nmap*** által feltételezett operációs rendszer tárolásra kerül. Nézzük, milyen információkat nyerhetünk a vizsgált címről:

```
mysql> select  * from targets
    -> where ip = ´192.168.10.44´;
```

Láthatjuk, hogy az `OS_guessed` mező tartalma jelen esetben `Linux Kernel 2.4.0-2.5.20`, a `hostname` (gépnév) oszlop pedig az `ophelia.hasnains.com` szöveget tartalmazza (szeretem Shakespeare tragikus hősnőit). Most, hogy már lényegében minden részlet a birtokunkban van, írjunk egy olyan lekérdezést, amely összegyűjtve tartalmazza a vizsgálat tárgyát képező gépre vonatkozó összes adatot.

```
mysql> select r.runid, r.d, r.t, t.ip, t.host,
    -> t.os_guessed, p.port, p.protocol,
       ➥p.service,
    -> p.state, p.fullversion from runlist r,
    -> targets t, portstat p
    -> where r.runid = 100 and p.target_ip = t.ip
    -> and p.runid = r.runid
    -> order by r.runid, r.d, r.t, t.ip;
```

Helyszűke miatt nem mutatjuk meg a lekérdezés kimenetét, próbáljuk ki nyugodtan saját magunk. Használhatnánk jelentéskészítő programot is az eredmény célpontok szerinti csoportosításához. Ha tetszetősebb kimenetet szeretnénk, hatékonyabb fegyvereket kell bevetnünk, például a ***PHP*** vagy a ***Perl*** tehet megfelelő erre a célra.
A jelentések közül az egyik leghasznosabb az egyes célpontok nyitott kapuinak változását szemléltető kimutatás. Vegyük például azt az esetet, amikor a vizsgált gépen bezáródott a 111/TCP kapu, de a 23/TCP nyitott állapotba került. Ebben az esetben a hoststats tábla open_ports oszlopa még mindig négy kaput fog mutatni, pedig a részletekben változás következett be. A megfelelő program könnyedén kiválogathatja a különbsége(ke)t egy jelentés számára.

Hasznos kimutatások

Az alábbi, leggyakrabban használt lekérdezés arra a kérdésre ad választ, hogy egy bizonyos célpont esetén mely kapuk vannak nyitott állapotban:

```
mysql> select d, t, port, protocol, state,
    -> fullversion from portstat
    -> where target_ip =  ´192.168.10.44´
    -> order by d,t,port;
```

5. lista Egy adott gépre vonatkozó összesített eredmények

```
+------------+----------+--------------+---------------------+---------------------------+------+----------+---------+-------+-------------------------------+
| d          | t        | ip           | host                | os_guessed                | port | protocol | service | state | fullversion                   |
+------------+----------+--------------+---------------------+---------------------------+------+----------+---------+-------+-------------------------------+
| 2003-12-14 | 10:00:37 | 192.168.10.44 | ophelia.hasnains.com | LinuxKernel 2.4.0 - 2.5.20 |   22 | tcp      | ssh     | open  | OpenSSH 3.5p1 (protocol 1. 99) |
| 2003-12-14 | 10:00:37 | 192.168.10.44 | ophelia.hasnains.com | LinuxKernel 2.4.0 - 2.5.20 |  111 | tcp      | rpcbind | open  | 2 (rpc #100000)               |
| 2003-12-14 | 10:00:37 | 192.168.10.44 | ophelia.hasnains.com | LinuxKernel 2.4.0 - 2.5.20 | 3306 | tcp      | mysql   | open  | MySQL4.0.16-standard-log      |
| 2003-12-14 | 10:00:37 | 192.168.10.44 | ophelia.hasnains.com | LinuxKernel 2.4.0 - 2.5.20 | 6000 | tcp      | X11     | open  | (access denied)               |
+------------+----------+--------------+---------------------+---------------------------+------+----------+---------+-------+-------------------------------+
```

Egy másik gyakran használt kimutatás azt mutatja meg, hogy egy célgép bizonyos kapuja nyitva volt-e valamikor egy korábbi időpontban: „Nyitva volt az SSH a `192.168.10.44` címen két héttel ezelőtt?” Feltéve, hogy az ***nmapsql*** telepítve volt már és a ***crontab*** rendszeres időközönként le is futtatta, a válasz a következő lekérdezéssel kideríthető:

```
mysql> select d, t,target_ip, port,protocol,
    -> service, state, fullversion from portstat
    -> where port = 22 and protocol = “tcp”
    -> and state = “open”
    -> d = date_sub( curdate(), interval 14 day)
    -> order by d, runid, target_ip ;
```

Természetesen egy adott napon több `nmapsql` futtatásunk is lehet, emiatt szerepel az order by záradék. Ha az eredményhalmaz előállításához olyan más gyártótól származó eszközt használunk, mint a ***PHP*** vagy a ***Perl***, a runlist táblából kell a pontos időkerethez tartozó futásazonosítót megszereznünk és ezt felhasználva kell a kiválasztott célra vonatkozó információkat lekérdeznünk.
Egy másik hasznos lekérdezéssel egy adott hálózat azon célpontjait szűrhetjük ki, amelyek bizonyos kapuja nyitott állapotban van: „A `192.168.10/24` alhálózat hány gépén van nyitva a 80/TCP kapu?” A keresett eredményt a következő lekérdezéssel kaphatjuk meg:

```
mysql> select runid, d, t, target_ip, port,
    -> protocol, state from portstats
    -> where port = 80 and protocol = ´tcp´
    -> and state = ´open´
    -> and target_ip like ´192.168.10.%´;
```

A szöveges megfeleltetés nem igazán alkalmas az alhálózat azonosítására, de a példa jól szemlélteti a megoldás lényegét.

Különböző adatbázisok használata

Számos esetben – például amikor egy szakértő több hálózaton dolgozik felváltva – kívánatos, hogy meg tudjuk változtatni az adatbázis nevét (ami esetleg lehet az ügyfelünk neve), így a különböző helyek adatai nem keveredhetnek össze egymással. A cikk írásának idején ezt úgy tehetjük meg, hogy módosítjuk az nmaplog által az adatbázis-tulajdonságok megadására szolgáló ***~/nmapsql.rc*** fájl ***db=nmaplog*** bejegyzést.
Az adatbázis menet közben való megváltoztatásához az ***~/nmapsql.rc*** fájlban lévő nmaplog helyére írjuk be a megfelelő nevet, és győződjünk meg arról, hogy a fájlban megadott felhasználó rendelkezik a megfelelő jogosultságokkal az így beállított adatbázishoz. Ezután töltsük be az adatbázis-sémát az új adatbázisba. Ha az új adatbázis neve newnmap, akkor az alábbi egysoros paranccsal tölthetjük át a sémát:

```
$ mysql newnmap < nmaplog.sql
```

Mégsem javaslom azonban a különböző adatbázisok használatát, sokkal egyszerűbb az adatokat egy fájlba kimásolni és ezután az nmaplog adatbázisba egy üres sémát betölteni. A következő parancssorokkal hajthatjuk végre ezt a műveletet:

```
$ mysqldump nmaplog > newnmap.sql
$ mysql nmaplog < nmaplog.sql
```

Az adatbázis-jogosultságok beállításától függően megeshet, hogy a parancsok fenti futtatásához létre kell hoznunk egy ***MySQL***-felhasználót és jelszót.

Alkalmazási módok

Ha csak ritkán és néhány gépen alkalmazzuk az ***nmapsql***-t, nehezebben ismerhetőek fel a program hasznos tulajdonságai. A program nagyobb környezetekben, több alhálózat és több tucatnyi vizsgálati célpont esetén mutatja meg az igazi tudását. A legegyszerűbb alkalmazási eset természetesen az, amikor az ***nmapsql*** és a ***MySQL*** kiszolgáló ugyanazon a gépen helyezkedik el, ami lehet például egy hálózati szakértő hordozható gépe, amellyel hálózatról hálózatra jár. Mivel a legtöbb hálózat tűzfalas védelem alatt áll és RFC 1918-as címzést használ, egy hálózatok közt kóborló hordozható gépen lévő targets táblában nagy a valószínűsége, hogy egyes IP-címek többször is előfordulnak. Ebben az esetben tehát át kell töltenünk az adatainkat és minden új környezetnél friss adatbázissal kell dolgoznunk.
Az ***nmapsql***-nek létezik más alkalmazási módja is: közepes méretű hálózatokban több pásztázó működhet a különböző alhálózatokban és a naplózásra használhatja ugyanazt a ***MySQL*** kiszolgálót, vagy nagy hálózatokban, ahol több egygépes rendszer (a ***MySQL*** és az ***nmapsql*** ugynazon a gépen fut) helyileg végzi a pásztázást és naplózást. Az ilyen rendszerekben kicsi az RFC 1918-as címek ketőződésének a valószínűsége. Igaz viszont, hogy a helyben történő pásztázás és naplózás valamint a központi kiszolgálón történő adatgyűjtés közti időkülönbség miatt nem rendelkezünk

Az nmaplog által használt táblák

Az *nmaplog* adatbázisban jelenleg nyolc tábla található, amelyek közül a fontosabbak az alábbiak:

- `TARGETS` – a célgépről tartalmaz információkat, többek közt a célazonosítót, IP-címet, a gép nevét és az *Nmap* által feltételezett operációs rendszert.
- `SCANNERS` – az *nmapsql*-t futtató gépről tartalmaz adatokat. A `portstat` táblához a `scan_id` mezővel kapcsolódik.
- `RUNLIST` – a felhasználó azonosítóját, az *Nmap* indításának dátumát és időpontját és a futtató gép információit tartalmazza. A felhasználó neve és azonosítója az /etc/passwd fájlból származik. A `scanner_id` mező a `scanners.scan_id` mezőhöz kapcsolódik.
- `PORTSTAT` – a kapupásztázás eredményeit tartalmazza. rögzítésre kerül az összes nmaplog jelzett kapu az állapotával (`open/close/filtered`, nyitott/zárt/szűrt) együtt.
- `HOSTSSTAT` – a célgép alapvető adatait, vagyis az nmapsql egyes futásainak időpontjaiban vizsgált összes, és az ebből nyitott állapotban lévő kapuk számát tartalmazza.

azonnali adatokkal. Mindkét helyzetre igaz, hogy a pásztázó-azonosító (scanner ID) jól használható az adatok szétválasztására.

Fejlesztési irányok

Gyakorló biztonsági szakértők – és el kell ismernem, néhány fekete kalapos számítógépes kalóz is – elismerik az nmapsql képességeit, amely nagyfokú elvárások kielégítésére is alkalmassá teszik. A projekt jelenlegi fejlesztési céljai közt szerepel, hogy a felhasználóknak lehetőségük legyen az *nmapsql* beállításait közvetlenül az *Nmap* felületén kersztül elvégezni, valamint egy olyan PHP-ben írt jelentéskészítő előtag létrehozása, amely megkíméli a végfelhasználót attól, hogy a lekérdezéseket a MySQL-be kelljen kézzel begépelnie. Ezek a kiegészítések pillanatnyilag fejlesztés alatt állnak. A távolabbi tervek közt szerepel a *Nessus* sebezhetőségvizsgálati eredményének ugyanabba az adatbázisba történő befoglalása, s ezzel egységes kezelőpult létrehozása a kapupásztázás és sebezhetőségi értékelés eredményei számára. A cél felé vezető lépésként az *nmapsql* honlapján találhatunk egy egyszerű formai elemzőt, amely alkalmas a *Nessus* ügyféllel létrehozott fájlok betöltésére.

Linux Journal 2004. október, 126. szám

Hasnain Atique (hatique@hasnains.com) a napos Szingapúrban él feleségével és három éves kislányával. Amikor lányával épp nem a Harry Pottert nézi, akkor próbál a hálózati gyűrűk ura lenni, ami néha sikerül is neki.