Az xinetd használata

Bemutatjuk, hogyan érdemes hozzákezdeni az xinetd beállításához.

A biztonságosabb xinetd hozzáférés-vezérlést, továbbfejlesztett naplózást és erőforráskezelést kínál a számunkra. A RedHat 7 és a Mandrake 7.2 változatok esetében már az xinetd lett a szabványos internetdémon. A cikk az xinetd 2.1.8.8pre3 változata alapján készült. Szeretném felhívni a figyelmet a démon néhány érdekesebb szolgáltatására.
Úgy tűnik, hogy az xinetd eredeti szerzője, *Panagoitis Tsirigotis* (panos@cs.colorado.edu) abbahagyta a projekt fejlesztését. Munkáját *Rob Braun* (bbraun@synack.net) vette át, és jelenleg ő felel a csomag karbantartásáért. Az egyetlen gond, amit a csomag jelenlegi állapotában észrevettem az, hogy néhány fejlécfájllal ki kellett egészítenem, hogy a ***s e l e c t ( )*** megfelelően együttműködjön a régi libc5-öt használó rendszeremmel.

Az xinetd

Az xinetd esetében az inetd-től megszokott sorok helyett zárójelezett, kibővített írásmódú sorokat láthatunk. Ezenkívül a szolgáltatás új naplózó és hozzáférés-vezérlő lehetőségekkel egészült ki. Az inetd lehetővé teszi, hogy TCP-kapcsolatainkat a Vename-féle ***tcp_wrapper*** programmal vezérelhessük, az UDP-kapcsolatok vezérlésére azonban nem teremt lehetőséget. Ezenkívül, miközben az inetd használata mellett szabályozni tudjuk kapcsolataink sebességét (a wait, illetve a nowait kapcsoló után feltüntetett értékkel), nincs lehetőség a példányok számának korlátozására. Ez könnyen szolgáltatásmegtagadás (denial of service) alapú támadásokhoz vezethet.
Az xinetd-t általában az alábbi paranccsal szoktam elindítani, a parancsot az internetszolgáltatások elindításához használt parancsfájlok egyikében helyezem el:

```
/usr/sbin/xinetd -filelog /var/adm/xinetd.log -f
/etc/xinetd.conf
```

A parancs szerint az xinetd tevékenységét a `/var/adm/xinetd.log` fájlba naplózza, és a `/etc/xinetd.conf` beállítófájlt használja. Ebben a cikkben túlnyomórészt ezzel a fájllal foglalkozunk majd.

Beállítások a fordításhoz

A fordítás során három, a hozzáférés-vezérlésért felelős kapcsolóra kell odafigyelnünk: libwrap, loadavg (küszöbfigyelés a terheléskiegyensúlyozáshoz) és az IPv6-támogatás. A legtöbb libwrappel dolgozó démonhoz (ilyen például a portmapper és a sendmail is) hasonlóan, a beállítófájlban elhelyezett „with-libwrap” kapcsoló arra vonatkozik, hogy az xinetd-nek támogatnia kell a /etc/hosts.allow és a /etc/hosts.deny fájlok használatát. Ez a kapcsoló pontosan úgy működik, mint az inetd esetében, és támogatja az összes, az inetd által vezérelt démont. Fontos megjegyezni, hogy az xinetd használata mellett nincs szükség többé a tcpd-re, mivel az xinetd a hozzáférések vezérlését is elvégzi. A libwrap-támogatás mégis nagyon hasznos abban az esetben, ha korábban az inetd/tcpd párost használtuk, és nem szeretnénk megváltoztatni a hozzáférési fájlokat.
A második érdekes beállítás a terhelés figyelése, amit a ./configure parancsfájlban elhelyezett with-loadavg kapcsolóval éleszthetünk fel. A sendmail képes arra, hogy túlságosan nagy terhelés esetén lebontsa a kapcsolatokat. Ezzel a kapcsolóval korlátozni tudjuk az egyes szolgáltatások (vagy akár az összes szolgáltatás) kapcsolatait a gép átlagos terheltsége alapján.

Végül az IPv6-támogatást a ./configure fájlban elhelyezett with-inet6 kapcsolóval tudjuk engedélyezni. Ha engedélyezzük, akkor az xinetd támogatni fogja az IPv6-címek és -kapcsolatok használatát. Ennek a hatását természetesen csak akkor fogjuk érezni, ha a rendszermag (és a hálózat) is támogatja az IPv6 protokollt. Az IPv4 támogatása természetesen továbbra is megmaradt.

1. táblázat **Az xinetd által használt irányelvek**

Irányelv	*Leírás*
socket_type	a hálózati csatlakozó típusa
protocol	IP protokoll, általában TCP vagy UDP
wait	igen vagy nem, egyenértékű az inetd wait/nowait kapcsolójával
user	a folyamat futtatásához használt felhasználói azonosító
server	a futtatni kívánt program teljes elérési útja
server_args	a futtatni kívánt alkalmazás kapcsolói, értékekkel együtt
instances	az elindítható példányok legnagyobb száma
start max_load	az a terheltségi szint, ahol le kell állítani a szolgáltatást (nem kötelező)
log_on_success	a sikeres tevékenységek naplózása
log_on_failure	a sikertelen kapcsolatok naplózása
only_from	azok a hálózatok, illetve gépek, ahonnan engedélyezzük a kapcsolódást
no_access	azok a hálózatok, illetve gépek, ahonnan nem engedélyezzük a kapcsolódást
disabled	ezzel a kapcsolóval tudunk a defaults{} részben szolgáltatásokat kikapcsolni
log_type	a napló elérési útja és típusa, FILE vagy SYSLOG
nice	a szolgáltatás elindításakor beállítandó nice szint
id	a szolgáltatásnak a naplókban használt neve

A beállítófájl

Az xinetd beállítófájlt (általában /etc/xinetd.conf) kétféleképpen hozhatjuk létre: kézzel, vagy pedig az inetd.conf fájlból elkészítve. Az első megoldás igen időigényes, ráadásul sok hibalehetőséget hordoz magában. Ha az önműködő létrehozás mellett döntünk, akkor az itox segédprogramot vagy az xconv.pl parancsfájlt használhatjuk. Habár az xconv.pl tökéletesen helyettesíti az itox segédprogramot, az utóbbinak továbbra is jó használt vehetjük. Ne feledkezzünk meg arról, hogy az itox újbóli lefuttatásával felülírjuk a korábban létrehozott fájlt. Az itox és az xconv használatában nincs különbség.
Mi most az itoxot fogjuk megnézni:

```
$ itox < /etc/inetd.conf > xinetd.conf
```

Az újabb segédprogram, az xconv, jobban bánik a megjegyzésekkel és a tcpd használatával, mint az itox. Az itox esetében meg kell

2. táblázat **A naplózó irányelvek értékei**

Érték	*Sikeres/Sikertelen*	*Leírás*
PID	sikeres	a sikeres kapcsolatfelvétel után elindított folyamat azonosítója
HOST	mindkettő	a távoli gép címe
USERID	mindkettő	a távoli felhasználó RFC 1413 azonosítója
EXIT	sikeres	az elindított folyamat sikeres lefutását jelzi
DURATION	sikeres	a kapcsolat időtartama
ATTEMPT	sikertelen	a kapcsolat sikertelenségének az oka
RECORD	sikertelen	kiegészítő adatok a sikertelen kapcsolatról

jelölnünk azt a könyvtárat, amely a démonokat tartalmazza (például /usr/sbin). Az első olyan rész, amit valószínűleg nem akarunk kihagyni a parancsfájlból, a defaults, amely az xinetd szolgáltatás alapértelmezett beállításait tartalmazza:

```
defaults
{
        instances       = 25
        log_type        = FILE /var/adm/servicelog
        log_on_success= PID HOST EXIT
        flags   = NORETRY
        log_on_failure= HOST RECORD ATTEMPT
        only_from       = 129.22.0.0
        no_access       = 129.22.210.61
        disabled        = nntp uucp tftp bootps who
                          shell login exec
        disabled        += finger
}
```

Rögtön láthatjuk, hogy az xinetd beállító értékek írásmódja a következőképpen néz ki: <irányelv> <műveletjel> <érték>. Az *1. táblázatban* azokat a irányelveket foglaltuk össze, amelyeket az xinetd értelmezni tud. A flags, type, env és passenv irányelvekkel most nem foglalkozunk. A későbbiek folyamán azonban részletesebben is kitérünk majd az only_from és a no_access irányelvekre, illetve a naplózási lehetőségekre is.

Kétféle műveletjelet használhatunk: = és +=. Ha az = jelet használjuk, akkor a bal oldalon álló irányelv felveszi a jobb oldalon megadott értéket. A += segítségével új értékeket adhatunk hozzá egy korábban meghatározott irányelvhez. Ha nem használjuk, akkor az értékeket felülírja. A szolgáltatásokat az alábbi formában kell megadni:

```
szolgáltatás_neve
{
        irányelv = érték
        irányelv += érték

}
```

A szolgáltatást a /etc/services fájlban is fel kell tüntetni, hogy a csatlakozó (socket) és a protokoll beállítása megfelelő legyen.

Néhány szó a hozzáférés-vezérlésről

Nézzük meg egy kicsit közelebbről, hogyan működik a hozzáférés-vezérlés az xinetd esetében. Először is, az xinetd a kapcsolatokat figyeli, nem pedig a csomagokat. Ennek következtében képes megakadályozni egy SYN vagy egy connect() kísérletet, ha az egy olyan gépről érkezik, amelynek nincs engedélye az adott szolgáltatás használatára, viszont semmit sem tud tenni a FIN-pásztázások ellen (ennek során olyan TCP-csomagok érkeznek a kapuhoz, amelyekben be van állítva a FIN kapcsoló). Nem támaszkodhatunk tehát az xinetd szolgáltatásra, ha a kapuk pásztázását szeretnénk megakadályozni. Egy találékony betörő ezen adatok felhasználásával hozzáférés-vezérlési listákat gyűjthet össze. Másodszor, az xinetd (legalábbis a 2.1.8.8pre3 változat) névkeresést hajt végre, amikor egy rendszer kapcsolódni próbál. Korábban ez a keresés a program indításakor zajlott le, de ez mostanra már megváltozott.

A hozzáférés-vezérlés használata meglehetősen egyszerű. Az első irányelv az only_from, amely azokat a hálózatokat vagy gépeket sorolja fel, ahonnan engedélyezzük a kapcsolatot. Az így felállított szabályokat a no_access irányelv segítségével szűkíthetjük. Az irányelvek után hálózati számokat (például 10.0.0.0 vagy 10), hálózatneveket (például *.my.com vagy .my.com), IP-címeket és gépneveket egyaránt megadhatunk. Ha minden kapcsolatot engedélyezni szeretnénk, akkor használjuk a 0.0.0.0 címet.

A szolgáltatás beállításai

Nézzünk meg néhány alapvető alkalmazást. Az első szolgáltatás, amit megvizsgálunk, az echo, amely az inetd és az xinetd esetében egyaránt belső szolgáltatás.

```
service echo
{
        socket_type     = stream
        protocol        = tcp
        wait    = no
        user    = root
        type    = INTERNAL
        id      = echo-stream
}
```

Az echo szolgáltatás rendszergazdai jogokkal fut és tcp protokollt használ. Az id irányelv után megadott echo-stream azonosító fog megjelenni a naplófájlokban. Az only_from és a no_access irányelvek hiánya azt jelzi, hogy a szolgáltatás korlátlanul hozzáférhető. Most nézzünk meg egy másik gyakran használt szolgáltatást, a daytime-ot:

```
service daytime
{
      socket_type       = stream
      protocol          = tcp
      wait              = no
      user              = nobody
      server            = /usr/sbin/in.date
      instances         = 1
      nice              = 10
      only_from         = 0.0.0.0
}
```

A szolgáltatáshoz továbbra is bárki kapcsolódhat, de a szakasz megadja, hogy a kérelmeket kiszolgáló program „nobody” felhasználó jogaival fusson. Ez a példa nem sokkal bővebb az előzőnél. Most egy másik szolgáltatást, az ssh1-et fogjuk megnézni, amely azt hivatott megakadályozni, hogy az sshd kifogyjon az erőforrásokból.

```
service ssh1
{
      socket_type       = stream
      protocol          = tcp
      instances         = 10
      nice              = 10
      wait              = no
      user              = root
      server            = /usr/local/sbin/sshd1
      server_args       = -i
      log_on_failure    += USERID
```

```
        only_from       = 192.168.0.0
        no_access       = 192.168.54.0
        no_access       += 192.168.33.0
}
```

Most a korábban látottakra építettünk. Emlékezzünk vissza arra, hogy az sshd-t a -i kapcsolóval kell elindítani abban az esetben, ha azt az inetd-hez vagy az xinetd-hez hasonló kiszolgálóról indítjuk, éppen ezért ezt a kapcsolót feltüntetjük a server_args irányelv mellett (ha a server irányelv mellett adnánk meg kapcsolókat, akkor nem tudna rendesen működni). Egyszerre legfeljebb tíz ember használhatja a szolgáltatást, ami nem okoz nehézséget annál a kiszolgálónál, ahonnan a példát vettük. A szokásos adatok mellett az RFC 1413-nak megfelelően naplózzuk azokat a felhasználókat is, akik nem tudtak kapcsolódni a szolgáltatáshoz. Végül két hálózatot sorolunk fel, ahonnan nem lehet elérni a szolgáltatást.

Az xinetd és a naplózás

A logging irányelv sokféle kapcsolót megért, és így különféle adatokat kaphatunk a kiszolgálóról (lásd 2. táblázat).

A szolgáltatások naplózására szolgáló sorok általában úgy néznek ki, ahogy azt az alábbi példákban is láthatjuk. Az olyan szolgáltatások esetében, amelyek sikeresen kapcsolódtak a gépünkhöz, általában a folyamat azonosítóját, a gép nevét és a kilépés időpontját szoktuk rögzíteni:

```
log_on_success = PID HOST EXIT
```

Ez elegendő adattal lát el bennünket azokban az esetekben, amikor a kiszolgáló természetes működésében jelentkező hibák okát keressük. A sikertelen kapcsolódások esetében ezzel szemben más jellegű adatokra lesz szükségünk:

```
log_on_failure = HOST RECORD ATTEMPT
```

Feljegyezzük a gép nevét, a kapcsolat sikertelenségének az okát és némi kiegészítő adatot a kapcsolódó gépről (ez néha tartalmazza például a kapcsolat felépítésével kísérletező felhasználó azonosítóját). Ezeket az adatokat célszerű nyilvántartani a naplóban, hogy jó rálátást nyerhessünk a kiszolgáló működésére.

Ha visszatekintünk az alapértelmezett beállításokat tartalmazó részhez, akkor láthatjuk, hogy a naplózás a /var/adm/servicelog fájlba történik. Minden esemény naplózását az xinetdére bíztuk. A legtöbb bejegyzés valahogy így fest majd:

```
00/9/13@16:05:07: START: pop3 pid=25679
from=192.168.152.133
00/9/13@16:05:09: EXIT: pop3 status=0 pid=25679
00/10/3@19:28:18: USERID: telnet OTHER :www
```

Ezen adatok birtokában már nekivághatunk az xinetd működésében felmerülő hibák felkutatásának. A napló tartalmából könnyen kideríthetjük azt is, hogy próbálkoztak-e olyan címekről kapcsolatot felépíteni a gépünkkel, amelyeket kizártunk az adott szolgáltatás használatából. Egyszerűen keressünk rá a naplóban a „FAIL” (Sikertelen) szóra, és máris megjelennek a megfelelő bejegyzések:

```
00/10/4@17:04:58: FAIL: telnet address
from=216.237.57.154
00/10/8@22:25:09: FAIL: pop2 address
from=202.112.14.184
```

A biztonsági kérdésekről még nagyon sokat lehetne beszélni, elég most annyi, hogy a naplóban talált IP-címeket nem szabad teljesen megbízható adatoknak tekinteni, ugyanis azokat viszonylag könnyű hamisítani.

Az xinetd.log fájl ezzel szemben az xinetd-vel kapcsolatos adatokat tárolja. A fájl tartalma nagy segítségével jelenthet, amikor a kapcsolatok hibáinak okát próbáljuk felderíteni.

```
00/10/25@21:10:48 xinetd[50]: ERROR: service
echo-stream, accept:
Connection reset by peer
```

Az xinetd átállítása

Az xinetd.conf fájlt akár az xinetd futása közben is átszerkeszthetjük. Ha szeretnénk életbeléptetni az új beállításokat, akkor egy SIGUSR1 jelet kell küldenünk a futó xinetd folyamatnak:

```
# ps -ax | grep xinetd
50  ?  S     5:47 /usr/sbin/xinetd -filelog
/var/adm/xinetd.log  -f/etc/xinetd.conf
# kill -SIGUSR1 50
```

Ha biztosak szeretnénk lenni abban, hogy a szolgáltatás az új beállításokkal újra lett indítva, akkor nézzünk bele a naplófájlba. Mielőtt kijelentkeznénk, mindenképpen célszerű elvégezni ezt az ellenőrzést; győződjünk meg arról is, hogy újra be tudunk-e jelentkezni. Megjegyzendő, hogy a -HUP kapcsoló hatására nem újraolvassa a folyamat a beállításokat, hanem beszünteti működését. Ezzel a kis trükkel szándékozták megállítani azon betörőket, akik nem ismerik a program leírását.

Mikor használjuk az xinetd-t

Én magam szinte minden szolgáltatásomhoz xinetd-t használok; az egyetlen olyan szolgáltatás, amelynél az xinetd használata a teljesítmény rovására megy, az az Apache. Túlságosan sok folyamatnak kell elindulnia túlságosan rövid idő alatt. A DNS-szolgáltatásokat szintén nem célszerű betölteni az xinetd-be, mivel ebben az esetben is komoly teljesítménycsökkenést tapasztalhatunk.

A sendmailt azonban az xinetd-n keresztül futtatom, mivel így egészen pontosan meghatározhatom, hogy ki kapcsolódhat a kiszolgálóhoz és ki nem. Nálam a sendmail az alábbi módon van beállítva:

```
service smtp
{
        socket_type     = stream
        protocol        = tcp
        wait    = no
        user    = root
        server = /usr/sbin/sendmail
        server_args     = -bs
        instances       = 20
        nice    = 10
        only_from       += 0.0.0.0
        no_access       += 129.22.122.84
                           204.0.224.254 }
```

Az xinetd használata olyan kis mértékben csökkenti csak a teljesítményt, hogy az még egy nagyobb forgalmú levélkiszolgálón is elhanyagolható.

Összefoglalás

Remélem, ez a cikk segít az Olvasónak abban, hogy az xinetd szolgáltatást saját igényeihez igazíthassa. Amint láthattuk, az xinetd lényegesen több lehetőséget tartalmaz, mint az inetd. A Solar Designer webhelyéről (➲ http://www.openwall.com) letölthetünk egy kiegészítést az xinetd egy régebbi (2.2.1-es) változatához, amely lehetővé teszi, hogy a példányokat IP-címenként szabályozhassuk. Ezzel elkerülhetjük az egyszerűbb táblafeldolgozó támadásokat, ne feledkezzünk meg azonban arról, hogy az IP-cím megváltoztatásával ez egyszerűen kijátszható. Nem tudom, hogy ez a kiegészítés bekerült-e az xinetd későbbi változataiba.

José Nazario
doktori tanulmányainak befejezéséhez közeledő biokémikus. Melléktevékenységként foglalkozik a különböző Linux- és Unix-változatokkal. Szabadidejében szeret horgászni és fényképezni.