Cgroups, chrony, sshd

План заняття

- Cgroups
- chrony
- sshd
- Моніторинг ОС, налагодження та журналювання
- Керування користувачами

Контрольні групи

Cgroups

Контрольні групи (або cgroups, як їх зазвичай називають) — це функція, яка надається ядром Linux для керування, обмеження та аудиту груп процесів. Порівняно з іншими підходами, такими як команда nice або /etc/security/limits.conf, контрольні групи більш гнучкі, оскільки вони можуть працювати з (під)наборами процесів (можливо, з різними користувачами системи).

До груп керування можна отримати доступ за допомогою різних інструментів:

- використання директив у файлах модулів systemd для визначення обмежень для служб і фрагментів
- шляхом прямого доступу до файлової системи cgroup
- за допомогою таких інструментів, як cgcreate, cgexec і cgclassify (частина пакету libcgroupAUR)
- використання «демона механізму правил» для автоматичного переміщення певних користувачів/груп/команд до груп (/etc/cgrules.conf та /usr/lib/systemd/system/cgconfig.service) (частина пакету libcgroupAUR)
- а також за допомогою іншого програмного забезпечення, наприклад віртуалізації Linux Containers (LXC).
- cgmanager застарів і не підтримується, оскільки він не працює з версіями systemd 232 і вище.

With systemd: встановлення

systemd є кращим і найпростішим методом виклику та налаштування контрольних груп, оскільки він є частиною встановлення за замовчуванням.

Переконайтеся, що у вас встановлено один із цих пакетів для автоматизованої обробки контрольних груп:

- systemd - для керування ресурсами служби systemd.
- libcgroupAUR - набір автономних інструментів (cgcreate, cgclassify, збереження через cgconfig.conf).

With systemd: Ієрархія

Поточну ієрархію контрольних груп можна побачити за допомогою команди systemctl status або systemd-cgls.

```
$ systemctl status
```

```
● myarchlinux
    State: running
     Jobs: 0 queued
   Failed: 0 units
    Since: Wed 2019-12-04 22:16:28 UTC; 1 day 4h ago
   CGroup: /
           ├─user.slice
           │ └─user-1000.slice
           │   ├─user@1000.service
           │   │ ├─gnome-shell-wayland.service
           │   │ │ ├─ 1129 /usr/bin/gnome-shell
           │   │ ├─gnome-terminal-server.service
           │   │ │ ├─33519 /usr/lib/gnome-terminal-server
           │   │ │ ├─37298 fish
           │   │ │ └─39239 systemctl status
           │   │ ├─init.scope
           │   │ │ ├─1066 /usr/lib/systemd/systemd --user
           │   │ │ └─1067 (sd-pam)
           │   └─session-2.scope
           │     ├─1053 gdm-session-worker [pam/gdm-password]
           │     ├─1078 /usr/bin/gnome-keyring-daemon --daemonize --login
           │     ├─1082 /usr/lib/gdm-wayland-session /usr/bin/gnome-session
           │     ├─1086 /usr/lib/gnome-session-binary
           │     └─3514 /usr/bin/ssh-agent -D -a /run/user/1000/keyring/.ssh
           ├─init.scope
           │ └─1 /sbin/init
           └─system.slice
             ├─systemd-udevd.service
             │ └─285 /usr/lib/systemd/systemd-udevd
             ├─systemd-journald.service
             │ └─272 /usr/lib/systemd/systemd-journald
             ├─NetworkManager.service
             │ └─656 /usr/bin/NetworkManager --no-daemon
             ├─gdm.service
             │ └─668 /usr/bin/gdm
             └─systemd-logind.service
               └─654 /usr/lib/systemd/systemd-logind
```

With systemd: знайти контрольну групу процесу

Ім’я контрольної групи процесу можна знайти в /proc/PID/cgroup.

Наприклад, контрольна група оболонки:

```
$ cat /proc/self/cgroup
0::/user.slice/user-1000.slice/session-3.scope
```

With systemd: використання ресурсів контрольної групи

Команду systemd-cgtop можна використовувати, щоб побачити використання ресурсів:

```
$ systemd-cgtop

Control Group                          Tasks   %CPU   Memory  Input/s Output/s
user.slice                               540  152,8     3.3G        -        -
user.slice/user-1000.slice               540  152,8     3.3G        -        -
user.slice/u…000.slice/session-1.scope   425  149,5     3.1G        -        -
system.slice                              37      -   215.6M        -        -
```

With systemd: власні контрольні групи

Файли *systemd.slice* можна використовувати для визначення власної конфігурації контрольної групи. Їх потрібно розмістити в каталозі systemd, наприклад /etc/systemd/system/. Параметри керування ресурсами, які можна призначити, задокументовані в systemd.resource-control.

/etc/systemd/system/my.slice

```
[Slice]
CPUQuota=30%
```

Не забудьте запустити systemctl daemon-reload, щоб отримати будь-які нові або змінені файли .slice.

With systemd: як послуга

Файл сервісного блоку

Ресурси можна вказати безпосередньо у визначенні служби або як файли

```
[Service]
MemoryMax=1G # Limit service to 1 gigabyte
```

With systemd: як послуга

systemd-run можна використовувати для запуску команди в певному фрагменті.

```
$ systemd-run --slice=my.slice command
```

Параметр --uid=username можна використати для створення команди як конкретного

```
$ systemd-run --uid=username --slice=my.slice command
```

Параметр --shell можна використовувати для створення командної оболонки всередині фрагмента.

With systemd: як непривілейований користувач

Непривілейовані користувачі можуть розділити надані їм ресурси на нові контрольні групи, якщо виконуються деякі умови. Cgroups v2 має бути використано, щоб користувач, який не є root, міг керувати ресурсами контрольної групи.

Перехід на cgroups v2

Linux підтримує контрольні групи як v1, так і v2. Однак за замовчуванням systemd монтує cgroup v1.

Ви можете виконати таку команду:

```
grep cgroup /proc/filesystems
```

Якщо ваша система підтримує cgroup v2, ви побачите:

```
nodev cgroup
nodev cgroup2
```

У системі з лише cgroup v1 ви побачите лише:

```
nodev cgroup
```

Перехід на cgroups v2 потребує одного з наступних параметрів ядра під час завантаження:

- systemd.unified_cgroup_hierarchy=1 - Systemd змонтує /sys/fs/cgroup як cgroup v2
- cgroup_no_v1="all" - Ядро вимкне всі контролери cgroup v1

With systemd: як непривілейований користувач

Крім того, ви можете змонтувати cgroups v2 вручну:

```
# mount -t cgroup2 none /sys/fs/cgroup
```

Перевірте, чи контрольні групи версії 2 змонтовано:

```
$ ls /sys/fs/cgroup
```

```
cgroup.controllers      cgroup.subtree_control  init.scope/      system.slice/
cgroup.max.depth        cgroup.threads          io.cost.model    user.slice/
cgroup.max.descendants  cpu.pressure            io.cost.qos
cgroup.procs            cpuset.cpus.effective   io.pressure
cgroup.stat             cpuset.mems.effective   memory.pressure
```

Якщо ви бачите щось подібне, ви все ще використовуєте v1 cgroups:

```
$ ls /sys/fs/cgroup
```

```
blkio/    cpu,cpuacct/  freezer/  net_cls@          perf_event/  systemd/
cpu@      cpuset/       hugetlb/  net_cls,net_prio/ pids/        unified/
cpuacct@  devices/      memory/   net_prio@         rdma/
```

With systemd: як непривілейований користувач

Типи контролерів

Controller	Can be controlled by user	Options
cpu	Requires delegation	CPUAccounting, CPUWeight, CPUQuota, AllowedCPUs, AllowedMemoryNodes
io	Requires delegation	IOWeight, IOReadBandwidthMax, IOWriteBandwidthMax, IODeviceLatencyTargetSec
memory	Yes	MemoryLow, MemoryHigh, MemoryMax, MemorySwapMax
pids	Yes	TasksMax
rdma	No	?
eBPF	No	IPAddressDeny, DeviceAllow, DevicePolicy

Примітка: технічно eBPF не є контролером, але параметри systemd реалізовані за допомогою нього, і лише користувач root може встановлювати їх.

With systemd: як непривілейований користувач

Делегування користувача

Щоб користувач міг контролювати ресурси процесора та вводу-виводу, ресурси потрібно делегувати. Це можна зробити шляхом створення перевантаження блоку.

Наприклад, якщо ваш ідентифікатор користувача 1000:

```
# systemctl edit user@1000.service
```

```
[Service]
Delegate=yes
```

Перезавантажте та переконайтеся, що фрагмент вашого сеансу користувача містить процесор і контролер вводу-виводу:

```
$ cat /sys/fs/cgroup/user.slice/user-1000.slice/cgroup.controllers
```

```
cpuset cpu io memory pids
```

With systemd: як непривілейований користувач

Визначені користувачем фрагменти

Файли фрагментів користувача можна розмістити в ~/.config/systemd/user/.

Щоб запустити команду під певним фрагментом:

```
$ systemd-run --user --slice=my.slice command
```

Ви також можете запустити свою оболонку входу всередині фрагмента:

```
$ systemd-run --user --slice=my.slice --shell
```

With libcgroup: спеціальні групи

Одна з переваг контрольних груп полягає в тому, що ви можете створювати "ad-hoc" групи на льоту. Ви навіть можете надати привілеї для створення спеціальних груп звичайним користувачам. groupname — ім'я контрольної групи:

```
# cgcreate -a user -t user -g memory,cpu:groupname
```

Тепер ваш користувач може записувати всі параметри в групі groupname:

```
$ ls -l /sys/fs/cgroup/memory/groupname

total 0
-rwxrwxr-x 1 user root 0 Sep 25 00:39 cgroup.event_control
-rwxrwxr-x 1 user root 0 Sep 25 00:39 cgroup.procs
-rwxrwxr-x 1 user root 0 Sep 25 00:39 cpu.rt_period_us
-rwxrwxr-x 1 user root 0 Sep 25 00:39 cpu.rt_runtime_us
-rwxrwxr-x 1 user root 0 Sep 25 00:39 cpu.shares
-rwxrwxr-x 1 user root 0 Sep 25 00:39 notify_on_release
-rwxrwxr-x 1 user root 0 Sep 25 00:39 tasks
```

With libcgroup: спеціальні групи

Cgroups є ієрархічними, тому ви можете створити скільки завгодно підгруп. Якщо звичайний користувач хоче запустити оболонку bash у новій підгрупі під назвою foo:

```
$ cgcreate -g memory,cpu:groupname/foo
$ cgexec    -g memory,cpu:groupname/foo bash
```

Щоб переконатися (лише для застарілих (v1) контрольних груп):

```
$ cat /proc/self/cgroup
```

```
11:memory:/groupname/foo
6:cpu:/groupname/foo
```

With libcgroup: спеціальні групи

Для цієї групи створено новий підкаталог. Щоб обмежити використання пам’яті всіма процесами в цій групі до 10 МБ, виконайте наступне:

```
$ echo 10000000 > /sys/fs/cgroup/memory/groupname/foo/memory.limit_in_bytes
```

Зауважте, що обмеження пам’яті стосується лише використання оперативної пам’яті – щойно завдання досягнуть цього обмеження, вони почнуть мінятися. Але це не вплине істотно на продуктивність інших процесів.

Подібним чином ви можете змінити пріоритет ЦП ("загальні ресурси") цієї групи. За замовчуванням усі групи мають 1024 спільні ресурси. Група зі 100 спільних ресурсів отримає ~10% часу процесора:

```
$ echo 100 > /sys/fs/cgroup/cpu/groupname/foo/cpu.shares
```

With libcgroup: постійна конфігурація групи

Примітка: використовуючи Systemd > = 205 для керування контрольними групами, ви можете повністю ігнорувати цей файл.

Якщо ви хочете, щоб ваші контрольні групи створювалися під час завантаження, ви можете замість цього визначити їх у /etc/cgconfig.conf. Наприклад, «назва групи» має дозвіл для $USER і користувачів групи $GROUP керувати обмеженнями та додавати завдання. Визначення групи підгрупи "groupname/foo" виглядатимуть так:

/etc/cgconfig.conf

```
group groupname {
  perm {
# who can manage limits
    admin {
      uid = $USER;
      gid = $GROUP;
    }
# who can add tasks to this group
    task {
      uid = $USER;
      gid = $GROUP;
    }
  }
# create this group in cpu and memory controllers
  cpu { }
  memory { }
}

group groupname/foo {
  cpu {
    cpu.shares = 100;
  }
  memory {
    memory.limit_in_bytes = 10000000;
  }
}
```

Chrony

Як встановити та використовувати Chrony в Linux

Chrony — це гнучка реалізація протоколу мережевого часу (NTP). Він використовується для синхронізації системного годинника з різних серверів NTP, еталонних годинників або за допомогою введення вручну.

Він також може використовуватися сервером NTPv4 для надання служби часу іншим серверам у тій же мережі. Він призначений для бездоганної роботи за різних умов, таких як переривчасте підключення до мережі, сильно завантажені мережі, зміна температури, яка може вплинути на годинник звичайних комп'ютерів.

Chrony поставляється з двома програмами:

- chronyc – інтерфейс командного рядка для chrony
- chronyd – демон, який можна запустити під час завантаження

Встановіть Chrony в Linux

У деяких системах chrony може бути встановлено за замовчуванням. Якщо пакет відсутній, його можна легко встановити. за допомогою інструмента керування пакунками за замовчуванням у відповідних дистрибутивах Linux за допомогою такої команди.

```
# yum -y install chrony    [On CentOS/RHEL]
# apt install chrony       [On Debian/Ubuntu]
# dnf -y install chrony    [On Fedora 22+]
```

Щоб перевірити статус chronyd, використовуйте наступну команду.

```
# systemctl status chronyd      [On SystemD]
# /etc/init.d/chronyd status    [On Init]
```

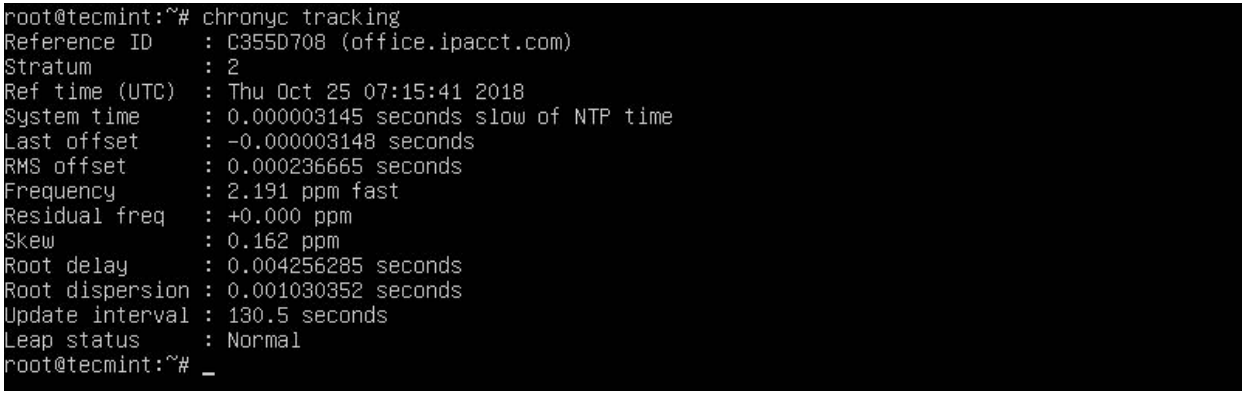
Якщо ви хочете ввімкнути демон chrony під час завантаження, ви можете скористатися наступною командою.

```
# systemctl enable chronyd       [On SystemD]
# chkconfig --add chronyd        [On Init]
```

Перевірте синхронізацію Chrony в Linux

Щоб перевірити, чи синхронізовано chrony, ми скористаємося програмою командного рядка chronyc, яка має опцію відстеження, яка надасть відповідну інформацію.

```
# chronyc tracking
```



Перевірте синхронізацію Chrony в Linux

Перелічені файли містять наступну інформацію:

- *Ідентифікатор посилання* – ідентифікатор посилання та ім’я, з яким зараз синхронізовано комп’ютер.
- *Stratum* – кількість переходів до комп’ютера з підключеним опорним годинником.
- *Ref time* – це час UTC, коли було зроблено останнє вимірювання з еталонного джерела.
- *Системний час* – затримка системного годинника від синхронізованого сервера.
- *Останнє зміщення* – передбачуване зміщення останнього оновлення годинника.
- *RMS offset* – довгострокове середнє значення зміщення.
- *Частота* – це частота, з якою системний годинник збивався б, якби Chronyd не виправляв його. Він надається в ppm (частки на мільйон).
- *Residual freq* – залишкова частота вказує на різницю між вимірюваннями від еталонного джерела та частотою, що використовується в даний момент.
- *Skew* – оцінена межа помилки частоти.
- *Коренева затримка – з*агальна кількість затримок мережевого шляху до комп’ютера рівня, з якого комп’ютер синхронізується.
- *Статус стрибка* – це статус стрибка, який може мати одне з наступних значень – нормальний, вставити другий, видалити другий або не синхронізований.

Перевірте синхронізацію Chrony в Linux

Щоб перевірити інформацію про джерела chrony, ви можете виконати таку команду.

```
# chronyc sources
```

```
root@tecmint:/home/user# chronyc sources
210 Number of sources = 8
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^- chilipepper.canonical.com     2   8   377   216    -29us[  -18us] +/-   48ms
^- pugot.canonical.com           2   7   377    87  -2030us[-2030us] +/-   43ms
^- alphyn.canonical.com          2   8   377   155  -3809us[-3809us] +/-  109ms
^- golem.canonical.com           2   8   377   544   -295us[ -256us] +/-   66ms
^- bigfoot.ieti.eu               2   8   377    89  +2543us[+2543us] +/-   66ms
^- home.mnet.bg                  3   8   377   154   +111us[ +111us] +/-  119ms
^- tryler.ludost.net             2   8   377   155   +213us[ +213us] +/-   52ms
^* office.ipacct.com             1   8   377   157    +67us[  +79us] +/- 3673us
```

Налаштуйте Chrony в Linux

Файл конфігурації chrony знаходиться за адресою /etc/chrony.conf або /etc/chrony/chrony.conf, а зразок файлу конфігурації може виглядати приблизно так:

```
server 0.rhel.pool.ntp.org iburst
server 1.rhel.pool.ntp.org iburst
server 2.rhel.pool.ntp.org iburst
server 3.rhel.pool.ntp.org iburst

stratumweight 0
driftfile /var/lib/chrony/drift
makestep 10 3
logdir /var/log/chrony
```

Налаштуйте Chrony в Linux

Наведена вище конфігурація надає таку інформацію:

- *server* – ця директива використовується для опису NTP-сервера для синхронізації.
- *stratumweight* – скільки відстані слід додати на страту до джерела синхронізації. Значення за замовчуванням 0,0001.
- *driftfile* – розташування та назва файлу, що містить дані дрейфу.
- *Makestep* – ця директива змушує chrony поступово виправляти будь-який часовий зсув шляхом прискорення або сповільнення годинника за потреби.
- *logdir* – шлях до файлу журналу chrony.

Якщо ви хочете негайно перевести системний годинник, ігноруючи будь-які поточні налаштування, ви можете скористатися такою командою:

```
# systemctl stop chrony          [On SystemD]
# /etc/init.d/chronyd stop       [On Init]
```

sshd

Як встановити сервер SSH на Linux

sshd — серверний процес OpenSSH. Він прослуховує вхідні підключення за допомогою протоколу SSH і діє як сервер для протоколу. Він обробляє автентифікацію користувача, шифрування, термінальні підключення, передачу файлів і тунелювання.

Сервер SSH зазвичай постачається як пакунок, який можна легко встановити в більшості дистрибутивів Linux. Однак він не завжди встановлюється за замовчуванням. Ви можете спробувати ssh localhost, щоб перевірити, чи він працює; якщо він відповідає щось на кшталт «Підключення відмовлено», це означає, що він не працює.

У похідних від Debian дистрибутивах команда встановлення SSH-сервера зазвичай така:

```
aptitude install openssh-server
```

Як встановити сервер SSH на Linux

У похідних дистрибутивах Red Hat команда зазвичай буде такою:

```
yum install openssh-server
```

Ці команди потрібно запускати від імені користувача root.

Якщо сервер не запускається автоматично, спробуйте скористатися командою запуску служби sshd або просто перезавантажте комп'ютер.

Запуск і ролі різних процесів sshd

Процес sshd запускається під час завантаження системи. Програма зазвичай знаходиться в /usr/sbin/sshd. Він працює як root. Початковий процес діє як головний сервер, який прослуховує вхідні з'єднання. Зазвичай це процес із найнижчим ідентифікатором процесу або процес, який виконується найдовше. Це також батьківський процес для всіх інших процесів sshd. Наступну команду можна використовувати для відображення дерева процесів у Linux, і легко побачити,

```
ps axjf
```

Запуск і ролі різних процесів sshd

Наприклад, у наведеному нижче виводі легко побачити, що процес 2183 є головним сервером.

```
  PPID   PID  PGID   SID TTY      TPGID STAT   UID   TIME COMMAND
  ...
     1  2183  2183  2183 ?           -1 Ss       0   8:51 /usr/sbin/sshd -D
  2183 12496 12496 12496 ?           -1 Ss       0   0:00  \_ sshd: cessu [priv]
 12496 12567 12496 12496 ?           -1 S    15125  24:07  |   \_ sshd: cessu
  2183 12568 12568 12568 ?           -1 Ss       0   0:00  \_ sshd: cessu [priv]
 12568 12636 12568 12568 ?           -1 S    15125   0:00  |   \_ sshd: cessu@pts/2
 12636 12637 12637 12637 pts/2    12637 Ss+  15125   0:00  |       \_ -zsh
  ...
```

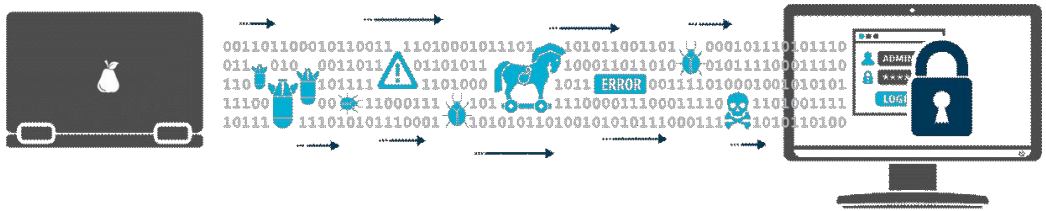
Інші процеси sshd є дочірніми процесами, які обслуговують одне з'єднання. Для кожного нового сеансу SSH створюється новий процес.

Запуск і ролі різних процесів sshd

Якщо сервер SSH оновлюється або перезапускається, зазвичай перезапускається лише головний сервер. Сервер розроблено таким чином, що серверні процеси, які обслуговують існуючі підключення, продовжують працювати. Це мінімізує збої в роботі користувачів, коли, наприклад, змінюється конфігурація сервера. Найпростішим способом перезавантаження SSH-сервера є використання служби sshd restart. Однак слід бути обережним під час віддаленого оновлення конфігурацій, оскільки помилки можуть перешкодити повторному підключенню до сервера (див. нижче).

Також можна вбити окремі процеси, завершивши серверний процес для певного користувача, терміналу або команди. Це можна зробити, наприклад, за допомогою команди kill -9 <processid>.

Також можна запустити кілька головних процесів sshd в одній системі. Це дуже незвично, але ми бачили корпоративних клієнтів з більш ніж десятьма серверами, що працюють одночасно з різними конфігураціями. Кожному серверу потрібно буде слухати інший порт і зазвичай матиме інший файл конфігурації.



Файл конфігурації

Сервер SSH має файл конфігурації, як правило, /etc/sshd/sshd_config. Конфігураційний файл визначає параметри шифрування, параметри автентифікації, розташування файлів, ведення журналу та різні інші параметри. Для детального опису див. документацію sshd_config.

Логування

Сервер SSH використовує підсистему syslog для журналювання. Є багато способів налаштувати syslog і декілька серверів syslog. Багато підприємств також збирають дані syslog у свою централізовану систему SIEM (Security Incident and Event Management).

У більшості систем syslog за замовчуванням налаштовано для реєстрації повідомлень, пов’язаних із SSH, у файлах у /var/log/. У системах, похідних від Debian, типовим файлом журналу є /var/log/auth.log. У системах, похідних від Red Hat, типовим файлом журналу є /var/log/secure.

Як системний журнал, так і рівень журналювання можна налаштувати у файлі конфігурації сервера. Наполегливо рекомендуємо встановити рівень журналювання на VERBOSE, щоб відбитки пальців для доступу до ключа SSH належним чином реєструвалися. Найновіші версії OpenSSH можуть реєструвати їх автоматично, але багато дистрибутивів Linux все ще мають версії, які не реєструють відбитки пальців без цього параметра. Перегляньте розділ Керування ключами SSH, щоб дізнатися, чому це важливо.

Налагодження підключення SSH

Іноді здається, що вхід на сервер SSH просто не працює, і може бути важко зрозуміти, у чому проблема. В основному є три інструменти, які допомагають діагностувати проблеми підключення та автентифікації:

Налагодження підключення SSH: клієнт SSH -v параметр

Перший підхід полягає в додаванні параметра -v під час виклику клієнта в командному рядку. Наприклад:

```
ssh -v [user@]host
```

Це надрукує докладний вихід налагодження, який зазвичай може визначити, у чому полягає проблема. Що потрібно перевірити:

- Чи успішно встановлюється з'єднання TCP із сервером? Якщо ні, це може бути проблема з DNS або маршрутизацією або сервер не працює. Якщо вихідні дані включають З'єднання встановлено, це означає, що з'єднання було успішним.
- Перевірте ім'я користувача, під яким він намагається автентифікуватися. Знайдіть рядок, що містить Authenticating to <hostname> як «<username>».

Налагодження підключення SSH: клієнт SSH -v параметр

Що потрібно перевірити:

- Переконайтеся, що він успішно узгоджує шифрування. Якщо ви бачите рядок із отриманим SSH2_MSG_SERVICE_ACCEPT, узгодження шифрування відбулося успішно. Якщо ні, то сервер або клієнт потрібно переналаштувати. Застарілий ключ хосту на клієнті також може спричинити це (використовуйте ssh-keygen -R <hostname> на клієнті, щоб видалити старий ключ хоста, якщо необхідно; див. ssh-keygen).
- Подивіться на методи автентифікації, які сервер готовий прийняти. Знайдіть рядки, що містять автентифікації, які можна продовжити: <список методів>. Якщо метод, який ви намагаєтесь використати, не включено, вам потрібно змінити конфігурацію сервера та перезапустити сервер. Це досить поширена причина проблем, якщо використовується щось інше, окрім пароля чи автентифікації відкритого ключа.
- Якщо ви бачите рядок Автентифікація успішна, це не проблема автентифікації. Якщо після цього вхід не вдається, це може бути проблемою з оболонкою входу користувача або, наприклад, .bashrc.
- Досить часто трапляється збій пересилання X11. За замовчуванням він вимкнений на сервері OpenSSH. Вам потрібно відредагувати файл sshd_config на сервері, щоб увімкнути рядок X11Forwarding yes. Часто його не можна вмикати на серверах корпоративних додатків, але в університетах, домашніх середовищах і на серверах розробки він зазвичай потрібен. Знову не забудьте перезапустити сервер.

Налагодження підключення SSH: файли журналу

Перегляд файлів журналу часто може виявити причину проблеми. Повідомлення, надіслані клієнту, навмисно створені для того, щоб мало розповісти про користувача, який увійшов у систему. Це з міркувань безпеки. Наприклад, ми не хочемо, щоб зловмисник міг перевірити, які облікові записи користувачів існують у цільовій системі. Більше інформації про, наприклад, помилки автентифікації часто можна знайти у файлі журналу.

Налагодження підключення SSH: запустіть сервер у режимі налагодження

Системний адміністратор може вручну запустити сервер із параметром -d, щоб отримати додатковий докладний вивід із сервера. Часто це останній засіб під час діагностики проблем з підключенням. Зазвичай причину помилок автентифікації досить чітко видно в його виводі.

Можливо, буде бажано запустити новий сервер на порту, відмінному від звичайного сервера, щоб не перешкоджати новим підключенням до сервера (особливо, якщо він віддалений!). У цьому випадку сервер буде запущено (як root) за допомогою щось на кшталт sshd -d -p 2222, а потім клієнт підключиться за допомогою ssh -p 2222 [user@]host.

Параметри командного рядка

Рідко доводиться вручну надавати параметри команд для SSH-сервера. Як правило, це робили б лише люди, які перепаковують SSH або створюють нові дистрибутиви Linux або нові вбудовані платформи (наприклад, пристрої IoT).

У OpenSSH доступні такі параметри:

- **-4** Використовуйте лише адреси IPv4. Це може використовуватися в середовищах, де DNS надає адреси IPv6, але маршрутизація для них не працює.
- **-6** Використовуйте лише адреси IPv6. Це можна використовувати для тестування, щоб переконатися, що підключення IPv6 працює.
- **-C connection_spec** Використовується для перевірки окремих блоків відповідності у файлі конфігурації в поєднанні з опцією -T. Connection_spec — це розділений комами список пар <keyword>=<value>, де <keyword> може бути одним із таких: користувач, хост, laddr, lport, addr. Кілька параметрів -C дозволені та комбіновані.
- **-c файл_сертифікату_хоста** Вказує шлях до файлу, що містить сертифікат хоста для хоста. Сертифікат має власний формат OpenSSH.

Параметри командного рядка

- **-D** Не від'єднуйтесь і не ставайте демоном. Це часто використовується, коли sshd запускається за допомогою systemd. Це дозволяє легше контролювати процес у таких середовищах. Без цієї опції сервер SSH розгалужується та від’єднується від терміналу, перетворюючи себе на фоновий процес демона. Останній донедавна був традиційним способом запуску SSH-сервера. Багато вбудованих систем все ще використовують останній.
- **-d** Вмикає режим налагодження. Сервер не розгалужується і завершить роботу після обробки одного з’єднання. Це можна використовувати для діагностики автентифікації користувача та інших проблем і зазвичай надає більше інформації про проблему, ніж налаштовано клієнту.
- **-E файл_журналу** Додає журнали до вказаного файлу замість надсилання їх до *syslog.*
- **-e** Записати журнали налагодження до стандартної помилки. Це можна використовувати для налагодження.
- **-f файл_конфігурації** Вказує шлях до файлу конфігурації сервера. За замовчуванням використовується */etc/ssh/sshd_config.*
- **-g login_grace** Вказує, як швидко користувачі мають автентифікуватися після відкриття з’єднання із сервером SSH. За замовчуванням 120 секунд, але це можна змінити у файлі конфігурації сервера. Час очікування запобігає постійному резервуванню ресурсів на сервері шляхом відкриття неавтентифікованого підключення до нього.

Параметри командного рядка

- **-h host_key_file** Вказує файл, з якого читається ключ хоста. За замовчуванням використовуються файли */etc/ssh/ssh_host_<algorithm>_key.* Для кожного алгоритму можна вказати лише один ключ хоста.
- **-i** Це буде використано, якщо сервер запускатиметься через inetd. Однак сьогодні цим ніхто не займається.
- **-k timeout** Цей параметр застарів. Він використовувався з SSH версії 1. Його використання наполегливо не рекомендується.
- **Параметр -o** Перевизначає будь-який параметр конфігурації, указаний у файлі конфігурації. Це може бути корисним для тестування та запуску кількох серверів на різних портах.
- **-p порт** Вказує порт, який прослуховує сервер. Типовим значенням є 22. Порт також можна вказати у файлі конфігурації сервера.
- **-q** Нічого не надсилає до системного журналу. Це не рекомендується; єдиним реальним використанням цієї опції для зловмисників було б приховати логіни за допомогою бекдору. Такої опції справді не повинно бути.
- **-T** Читає файл конфігурації сервера, перевіряє його синтаксис і завершує роботу. Це корисно для перевірки того, що конфігураційний файл правильний перед перезапуском сервера. Перевірка файлу конфігурації особливо важлива, якщо конфігурація оновлюється віддалено. Фактично, у таких випадках найкраще спочатку перевірити нову конфігурацію, запустивши другий сервер на новому порту, і перезапустити основний сервер лише після успішного входу за допомогою тестового сервера. Це можна поєднати з опцією -C для перевірки окремих блоків відповідності у файлі конфігурації.

Параметри командного рядка

- **-t** Перевіряє дійсність файлу конфігурації та ключів, на які посилаються. Перегляньте -T для отримання порад щодо додаткового тестування перед віддаленим перезапуском сервера.
- **-u len** Цей незрозумілий параметр має лише одну корисну мету: вказівка -u0 призводить до того, що IP-адреси з крапками зберігаються у файлі utmp (який містить інформацію про входи на сервер). Це вимикає пошук DNS сервером SSH, якщо цього не вимагає механізм автентифікації або шаблони from= на авторизованих ключах. Інакше він вказував би розмір структури utmp на хості, випадки, коли його потрібно вказувати вручну, дуже рідкісні.

Питання та відповіді