Якимчук Сергій

Налаштування та адміністрування офісних Linux-серверів

2019
http://yakim.org.ua

Зміст

Частина І.
Знайомство з системою. Базові офісні сервіси.

Вступ

Linux, можливо, є найбільш значним досягненням в області вільного програмного забезпечення. Він перетворився в операційну систему для бізнесу, освіти та індивідуального програмування. Linux перестав бути системою для фанатиків-програмістів, які годинами сидять перед своїми моніторами (хоча таких і досі не мало).

На сьогодні родина операційних систем Linux займає передову позицію в області серверів, та серверних додатків. Сьогодні важко придумати завдання, яке було б неможливо вирішити за допомогою вільного програмного забезпечення. В деяких областях — таких як поштові сервери, веб-сервери використання відкритого програмного забезпечення де-факто стало стандартом.

Крім цього Linux є більш захищеною системою, ніж Windows. Linux має довгу історію використання ретельно опрацьованої багатокористувацької архітектури. Обмеження за замовчуванням — властивість модульної архітектури Linux. Навіть сервіси, наприклад, веб-сервери, зазвичай запускаються від користувача з обмеженими повноваженнями. Так в Ubuntu Linux веб-сервер Apache працює з правами користувача www-data. Навіть якщо зловмисник на такому сервері якимсь чином отримає повний контроль над веб-сервером Apache, він зможе керувати лише файлами, що належать користувачеві “www-data”, тобто веб-сторінками. Крім того, користувачі, що асоційовані з такими сервісами, як Apache, MySQL та ін., зазвичай встановлюються з обліковими записами, що не мають доступу до командного рядка. Тому навіть якщо зловмисник зможе отримати права такого облікового запису, він не зможе використати її для виконання довільних команд на сервері Linux, оскільки цей аккаунт не може викликати команди.

В зв'язку з тим, що доля Linux на серверних платформах постійно збільшується, а також у зв'язку з часто нічим не виправданою дорожнечею комерційного програмного забезпечення, потреба у співробітниках, які б могли забезпечити встановлення та підтримку Linux-платформ, також постійно збільшується.

Ця книга присвячена налаштуванню офісних серверів на основі Ubuntu Linux. Призначена вона, перш за все, для адміністраторів, що тільки починають знайомитись з цією чудовою системою, Однак, сподіваюсь, що досвідченим користувачам та сисадмінам вона теж може стати в пригоді.

Знайомство з системою Linux

Linux — загальна назва Unix-подібних операційних систем на основі ядра того ж імені, бібліотек та системних програм, що розроблені в мужфх проекту GNU, а також іншого програмного забезпечення.

Linux працює на великій кількості архітектур процесора, таких як Intel x86, x86-64, PowerPC, ARM, Alpha AXP, Sun SPARC, Motorola 68000, Hitachi SuperH, IBM S/390, MIPS, HP PA-RISC, AXIS CRIS, Renesas M32R, Atmel AVR32, Renesas H8/300, NEC V850, Tensilica Xtensa та багатьох інших.

На відміну від більшості інших операційних систем, Linux не має єдиної “офіційної” комплектації. Замість цього Linux розповсюджується у вигляді великої кількост дистрибутивів, в яких ядро Linux сполучається з утилітами GNU та іншими прикладними програмами, що робить його повноцінним операційним середовищем.

Найбільш відомими дистрибутивами Linux являються Arch Linux, CentOS, Debian, Fedora, Gentoo, Mandriva, Mint, openSUSE, Red Hat, Slackware, Ubuntu.

Дистрибутиви на основі Linux мають широке застосування в різних областях: від вбудованих систем до суперкомп'ютерів. Вони надійно утримують передові позиції на ринку серверів, як правило в складі комплексу серверного програмного забезпечення.

Також зростає використання Linux як десктопного середовища для дому та офісу. Дистрибутиви Linux користуються популярністю у різних державних структур. Федеральний уряд Бразилії добре відомий своєю підтримкою Linux, а російські вояки розробляють свій власний дистрибутив Linux. Уряд індійського штату Керала випустив наказ про переведення всіх шкіл на використання Linux. Для забезпечення технологічної незалежності Китай використовує лише Linux на своїх процесорах Loongson. Деякі регіони Іспанії розробили власні Linux-дистрибутиви, що використовуються в освіті та державних структурах, наприклад такі як gnuLinEx в Естремадурі та Guadalinex в Андалусії. Португалія також використовує свій власний дистрибутив Caixa Mágica, що був розроблений для нетбука Magalhães та державної програми електронної освіти. Франція та Німеччина роблять кроки в напрямку збільшення використання Linux. Крім цього і у нас в Україні є приклади повного переведення інфраструктури на Linux. Такими прикладами можуть служити Приватбанк, УкрПромБуд та інші.

Файлова система

Операційна система Linux підтримує велику кількість файлових систем, на сьогодні найбільш широко використовуються: **ext2, ext3, ext4, ReiserFS та xfs**. Також сучасні ОС Linux сумісні з файловими системами, що використовуються в ОС Windows, такими як **NTFS та FAT32**, але використання цих файлових систем в Linux небажане, бо ці системи розроблялись під іншу операційну систему і в зв'язку з цим підтримка Windows-розділів в ядрі Linux реалізована за допомогою сторонніх утиліт/драйверів, що накладає деякі обмеження (наприклад, ОС Linux не має можливості розмежовувати права доступу до файлів на розділах NTFS).

Структура файлової системи

В операційній системі Windows при відкритті теки “Мій комп'ютер” користувач звик до

наступної картини: зазвичай один або більше твердих дисків (частіш за все логічних) що називаються починаючи з латинської літери C. Кожен з дисків є кореневою текою. Так, наприклад, якщо в системі є три диска, то буде три кореневих теки (скоріш за все C, D та E), кожен з яких містить вкладені теки та файли. Іншими словами в системі буде існувати три дерева. Оскільки час від часу треба користуватися компакт-дисками та USB-накопичувачами, то періодично будуть “виростати” ще кілька дерев.

В дистрибутивах Linux все дещо інакше. Файлова система цілісна і має лише одну кореневу теку, яка позначається косою рискою — слеш (/). Тут треба звернути увагу на відмінність від Windows. В ній при формуванні повної адреси використовується обернена коса риска “\”. В Linux при формуванні повної адреси завжди використовується “/”.

Файлова структура Linux має певну структуру тек.

В ній все впорядковано та “лежить” на своєму місці. Кожна тека має власне призначення, яке регламентується документом під назвою FHS (Filesystem Hierarchy Standart — стандарт структури файлової системи).

Ось короткий опис призначення основних тек згідно даного стандарту.

Тека	**Опис**
/	Коренева тека, що містить всю файлову ієрархію
/bin/	Основні утиліти, які необхідні як в однокористувацькому режимі, так і при звичайній роботі всім користувачам (наприклад: cat, ls, cp)
/boot/	Завантажувальні файли (в тому числі файли завантажувача, ядро, initrd, System.map). Часто виноситься на окремий розділ.
/dev/	Основні файли пристроїв (наприклад, /dev/null, /dev/zero)
/etc/	Загальносистемні конфігураційні файли (назва походить від *et cetera*).
/etc/opt/	Файли конфігурації для /opt/.
/etc/X11/	Файли конфігурації X Window System версії 11.
/home/	Містить домашні теки користувачів, які, в свою чергу, містять персональні налаштування та дані користувача. Часто виноситься на окремий розділ.

/lib/	Основні бібліотеки, що необхідні для роботи програм з /bin/ и /sbin/
/media/	Точки монтування для змінних носіїв, таких як CD-ROM, DVD-ROM (вперше описано в FHS-2.3)
/mnt/	Містить тимчасово змонтовані файлові системи
/opt/	Додаткове програмне забезпечення
/proc/	Віртуальна файлова система, що надає стан ядра операційної системи та запущених процесів у вигляді файлів
/root/	Домашня тека користувача root
/sbin/	Основні системні програми для адміністрування та налаштування системи, наприклад: init, iptables, ifconfig
/srv/	Дані, що специфічні для оточення системи
/tmp/	Тимчасові файли
/usr/	Вторинна ієрархія для даних користувача; містить більшість програмних додатків користувача та утиліт, що використовуються в багатокористувацькому режимі. Може бути змонтована через мережу в режимі “лише для читання” та бути загальною для декількох комп'ютерів
/usr/bin/	Додаткові програми для всіх користувачів, які не є необхідними в однокористувацькому режимі.
/usr/include/	Стандартні файли заголовків.
/usr/lib/	Бібліотеки для програм, що знаходяться в /usr/bin/ та /usr/sbin/.
/usr/sbin/	Додаткові системні програми (такі як демони різних мережевих сервісів).
/usr/share/	Архітектурно-незалежні загальні дані.
/usr/src/	Вихідні коди (наприклад, тут містяться вихідні коди ядра).

/usr/X11R6/	X Window System, версії 11, реліз 6.
/usr/local/	*Третинна ієрархія* для даних, що специфічні для даного хосту. Зазвичай містить такі підтеки, як bin/, lib/, share/
/var/	Файли, що часто змінюються, наприклад файли реєстрації (log-файли), тимчасові поштові файли, файли спулерів.
/var/lib/	Постійні дані, що змінюються програмами в процесі роботи (наприклад, бази даних, метадані пакетного менеджера та інш.).
/var/lock/	Лок-файли, що вказують на зайнятість деякого ресурсу.
/var/log/	Різні файли реєстрації (log-файли).
/var/mail/	Поштові скриньки користувачів
/var/run/	Інформація про запущені програми (в основному, про демони).
/var/spool/	Завдання, що чекають на обробку (наприклад, черги друку, непрочитані або не відправлені листи).
/var/spool/mail/	Місце для поштових скриньок користувачів (застаріле).
/var/www/	Файли веб-сайтів (наприклад, ієрархія файлів віртуальних хостів).
/var/tmp/	Тимчасові файли, які повинні бути збережені між перезавантаженнями.

Користувачі та групи в Linux

Операційна система Linux є багатокористувацької системою в якій може одночасно існувати досить велика кількість користувачів.

В Linux робота з користувачами містить набір наступних маніпуляцій: додавання, видалення або модифікація налаштувань користувача або групи.

Крім облікових записів, які використовують люди для роботи з системою, в Linux передбачені облікові записи для *системних користувачів*: з точки зору системи це такі ж користувачі, однак ці облікові записи використовуються виключно для роботи деяких програм-сервісів. Наприклад, стандартний системний користувач mail використовується програмами доставки пошти.

Коли в системі створюється новий користувач, він додається щонайменше в одну групу. В системі при створенні нового облікового запису створюється спеціальна група, назва якої співпадає з іменем нового користувача і користувач включається в цю групу. В подальшому адміністратор може додати користувача і в інші групи.

Механізм груп може використовуватись для організовування спільного доступу кількох користувачів до певних ресурсів. Наприклад на сервері компанії для кожного проекту може бути створена окрема група, в яку будуть входити облікові записи співробітників, що працюють над цим проектом. При цьому файли, що відносяться до проекту, можуть належати цій групі та бути доступними для її членів. В системі також існує певна кількість груп (наприклад bin), які використовуються для керування доступом системних програм до різних ресурсів. Як правило членами таких груп є системні користувачі, а користвачі-люди до таких груп не входять.

Також за допомогою груп можуть бути надані права, що є необхідними для виконання певних задач. Наприклад для додавання, видалення або налаштування принтера в системі користувач повинен входити в групу lpadmin.

Керування користувачами та групами

Створення користувача

 # adduser username

Створення групи

 # addgroup groupname

Додавання користувача в групу

1 спосіб

 # addgroup username groupname

2 спосіб

usermod -aG groupname username

Видалення користувача з групи
 # delgroup username groupname

Видалення користувача
 # deluser username

Права доступу до файлів та тек

Оскільки система Linux з самого початку розроблялась як багатокористувацька система, в ній передбачений такий механізм, як права доступу до файлів та тек. Він дозволяє розмежовувати повноваження користувачів, які працюють в системі. Зокрема, права доступу дозволяють окремим користувачам мати «особисті» файли та теки.

Правильне налаштування прав доступу дозволяє підвищити надійність системи, захищаючи від змін або видалення важливі системні файли. А оскільки зовнішні пристрої з точки зору Linux також являються об'єктами файлової системи, механізм прав доступу можна застосовувати й для керування доступом до пристроїв.

У будь-якого файла у системі є власник — один з користувачів. Однак кожен файл одночасно належить ще й до деякої групи користувачів системи. Кожен користувач може входити в будь-яку кількість груп, і в кожну групу може входити довільна кількість користувачів з числа тих, які визначені в системі.

Права доступу визначаються по відношенню до трьох типів дій: читання, запис та виконання. Ці права можуть бути надані трьом класам користувачів: власнику файла (користувачу), групі, якій належить файл, а також всім іншим користувачам, що не входять в цю групу. Право на читання дає користувачу змогу читати вміст файла або, якщо такий доступ дозволений до теки, продивлятися вміст теки (використовуючи команду **ls**). Право на запис дає можливість користувачу записувати або змінювати файл, а право на запис для теки — можливість створювати або видаляти файли в цій теці. Врешті, право на виконання дозволяє користувачу запускати файл як програму або сценарій командної оболонки (зрозуміло, що ця дія має сенс лише в тому випадку, коли файл є програмою або сценарієм). Володіння правами на виконання для теки дозволяє перейти (командою **cd**) в цю теку.

Візуальне значення прав доступу записується двома способами — в символьному вигляді, або у вигляді вісімкового числа. У символьному вигляді це r, w, x або їх комбінація.

У вісімковому відображенні кожному виду доступу відповідає число

x відповідає 1, **w** — 2, а **r** — 4

Комбінація символьних значень в вісімковому вигляді буде виглядати як сума відповідних значень.

Символьний вигляд	**Вісімковий вигляд**
r	4
w	2
x	1

Так rwx — тобто повний доступ у вісімковому вигляді буде 1 + 2 + 4 = 7

rw- — доступ на читання та запис — 2 + 4 = 6

а r-x — доступ на читання и виконання — 1 + 4 = 5

Повний запис прав доступу до файлу складається з трьох частин: право власника, право групи та право всіх інших та записується відповідно

rwxr-xr-- — в символьному вигляді
або
754 — в вісімковому вигляді

Перегляд встановлених прав доступу:
$ ls -l

Зміна прав доступу до файла

1 спосіб — явне вказання всіх прав

chmod 640 ~/test.txt

2 спосіб — зміна певного права доступу

chmod g+w ~/test.txt

Аналогічним чином можна змінювати і власника файлу

 # chown newuser:newgroup ~/test.txt

Атрибути файлів і тек за замовчуванням. umask.

umask — функція середовища POSIX, що змінює права доступу, які надаються новим файлам та текам за замовчуванням. umask зазвичай встановлюється у вісімковій системі числення.

Фактично, umask вказує, які біти треба скинути в правах на файл, що встановлюються — кожен встановлений біт umask забороняє встановлення відповідного біту прав.

Для себе слід запам'ятати — якщо ви створюєте файл, то значення маски віднімається від значення 666, а якщо створюєте теку, то значення маски віднімається от значення 777.

Продивитися поточне значення umask

$ umask

За замовчуванням значення umask — 22. Це значить, що всі файли створюються з правами 666 — 22 = 644, тобто rw-r--r--, або, простіше кажучи, у володаря файла є права на читання та запис, але не на виконання, а у інших (як у групи, так і у всіх інших користувачів) права лише на читання.

Зміна umask

$ umask 002

Тепер при створенні нового файлу він з'явиться з правами 666 - 2 = 664, тобто rw-rw-r--

Біти атрибутів SUID та SGID

В Unix-подібних системах програма запускається з правами користувача, який викликав вказаний додаток. Це забезпечує додаткову безпеку, так як процес з правами користувача не зможе отримати доступ на запис до важливих системних файлів, наприклад */etc/passwd*, який належить суперкористувачу root. Якщо на виконуваний файл встановлений біт suid, то при виконанні ця програма автоматично змінює «ефективний userID» на ідентифікатор того користувача, який є власником цього файла. Тобто, в незалежності від того — хто запускає цю програму, вона при виконанні має права власника цього файла.

Встановлення біту suid на файли, що не є виконувальними, не має сенсу. Також немає рації встановлювати цей біт на теку. В останніх версіях ядра Linux додане ще одне обмеження — з міркувань безпеки suid-біт може застосовуватись лише до бінарних файлів, але аж ні як щодо скриптів.

Коли груповий s-бит встановлюється для теки, то кожен файл, що буде створено в цій теці буде віднесено до тієї групи, до якої відноситься і сама тека (але не користувач, що створює цей файл). Використання цього біта має сенс лише відносно теки.

Встановлення SGID-біта

 # chmod g+s ~/testdir

Зняття SGID-біта

 # chmod g-s ~/testdir

Додаткові атрибути файлів

Крім права на читання/запис/виконання файлові системи ext2, ext3 та ext4, які використовуються в Linux підтримують додаткові атрибути файлів.

Необхідно зразу відзначити, що в більшості випадків вам не прийдеться з ними працювати, але знати про те, що вони існують потрібно. Коротко опишемо деякі з них.

A - не оновлювати час доступу до об'єкту. Теоретично встановлення цього атрибуту має підвищити продуктивність файлової системи і, відповідно, системи в цілому.

a - вказує, що в файл можна додавати інформацію, але не можна видаляти.

d - вказує на те, що не треба робити резервні копії файла. Файл буде проігноровано командою **dump**.

i - вказує на те, що файл не можна видаляти та модифікувати.

s - вказує, що при видаленні файла місце де був розміщений файл буде перезаписано нулями.

u - вказує на те, що при видаленні файла його треба кудись зберегти.

Повний список атрибутів можна побачити прочитавши

$man chattr

Існують і інші атрибути. Про них ви можете прочитати в довідці. Також треба звернути увагу на те, що не всі атрибути працюють на всіх файлових системах.

Змінювати додаткові атрибути можна командою **chattr, a** для того, щоб їх продивитися існує команда **lsattr**.

Монтування дисків. Mount та fstab.

Розглянемо процедуру монтування файлових систем. Монтування може здійснюватися автоматично, або вручну.

mount — утиліта командного рядка в UNIX-подібних системах. Застосовується для монтування файлових систем.

Ручне монтування локальних ресурсів. Команда mount.

Варіант команди монтування, що зустрічається найчастіше, буде виглядати так:

mount -t тип_файлової_системи -o опції_монтування що_монтуємо куди_монтуємо

Наприклад, монтування ntfs-розділу на нашому ж жорсткому диску в теку /mnt/win буде виглядати так:

mount -t ntfs-3g -o defaults /dev/sda5 /mnt/win

де

ntfs-3g — тип файлової системи

defaults — всі опції монтування за замовчанням

/dev/sda5 — пристрій, який монтується

/mnt/win — тека, в яку буде проведено монтування

Якщо при монтуванні виникли проблеми з кирилицею на ntfs-розділі, в опціях монтування можна явно вказати кодову сторінку:

mount -t ntfs-3g -o locale=uk_UA.utf8 /dev/sda5 /mnt/win

Найчастіше утиліта mount всі потрібні параметри (в тому числі і тип файлової системи) може визначити сама. У такому випадку команда монтування може виглядати так:

mount /dev/sdb1 /mnt/new_disk

За допомогою mount можна монтувати не тільки розділи на фізичних пристроях, а й просто файли, наприклад iso-образи дисків. В такому випадку використовується пристрій loop:

mount -t iso9660 -o loop /home/user/ubuntu-10.04.3-server-i386.iso /home/ubuntu/

Автоматичне монтування. Файл fstab.

Автоматичне монтування файлових систем проводиться за допомогою файлу */etc/fstab*. Саме в ньому й описано монтування пристроїв, яке буде здійснено при завантаженні операційної системи.

Якщо ми відкриємо цей файл в Ubuntu, то побачимо приблизно таке:

```
proc            /proc           proc    nodev,noexec,nosuid 0       0
UUID=87c97237-0bf0-4a62-bae2-47a850643996 /               ext4    errors=remount-ro 0       1
UUID=0af767a3-c740-47fe-87ad-fb495bc1444e /home           ext4    defaults        0       2
UUID=45bc8486-ed86-4a86-96db-ea34e4c0b9da none            swap    sw              0       0
```

Кожен запис в цьому файлі містить 6 полів. Його формат простий:

Пристрій — Точка монтування — Тип файлової системи — Опції монтування — вказування необхідності резервного копіювання — Порядок перевірки розділу.

Пристрій може визначатися не тільки через UUID, а й через мітку тома, або просто як файл пристрою, наприклад /dev/sda1.

Точку монтування вибираємо самі по необхідності. Вона не обов'язково повинна знаходитися безпосередньо в кореневому розділі. Наприклад у мене на поштовому сервері окремий розділ примонтувати як /var/mail.

Тип файлової системи — тут все зрозуміло. В який формат відформатували, те і пишемо.

Опції монтування — тут можна дуже багато чого описати. Наприклад: монтувати пристрій тільки для читання, або в режимі читання і запису, кодування імен файлів, чи зберігати час доступу до файлів, логін-пароль для доступу (якщо це мережевий ресурс) і так далі.

Ознака необхідності резервного копіювання — зазвичай не використовується і встановлюється в 0.

Порядок перевірки розділу (0 — не перевіряти, 1 — встановлюється для кореня, 2 — для решти розділів).

Тепер деякі подробиці про опції монтування. Всі опції записуються через кому.

async — всі дії введення/виводу будуть проводитися асинхронно
noatime — не оновлювати час доступу до файлів
defaults — використовувати опції за замовчуванням: rw, suid, dev, exec, auto, nouser і async.
exec — дозволяти виконання бінарних файлів
noexec — не дозволяти виконання бінарних файлів
suid — дозволяти використовувати біти set-user-identifier і set-group-identifier.
nosuid — не дозволяти використовувати біти set-user-identifier і set-group-identifier.
ro — монтувати пристрій в режимі тільки для читання
rw — монтувати пристрій в режимі читання/запису
users — дозволяти всім користувачам монтувати і демонтувати цей пристрій

Крім цього, варто звернути увагу, що при монтуванні різних файлових систем, опції монтування можуть змінюватися. Особливо це відноситься до мережевих файлових систем, таким як sshfs, curlftpfs, smbfs і їм подібним.

Система керування пакунками

Пакет Debian - архівний файл (зібраний утилітою ar), що містить два звичайних архіву *.tar.gz, один з яких включає скомпільовані виконувальні бінарники (та необхідні для їх роботи компоненти — бібліотеки, конфіги, документацію и так далі, другий же — так звані керуючі файли: контрольні суми, опис залежностей, перед- та післяінсталяційні сценарії.

Засоби управління пакетами повинні не тільки забезпечити розгортання архівів і розміщення їх в потрібних місцях файлової системи, але й якимось чином відреагувати на залежності — або просто сповістити про їх порушення, або спробувати задовольнити їх своїми силами.

Поняття залежностей в Debian та його похідних відрізняється від прийнятого в більшості інших дистрибутивів (і взагалі ОС Unix-родини). Зазвичай розрізняють лише:

• обов'язкові (або "жорсткі") залежності, без задоволення яких встановлення та робота програми неможлива (наприклад, залежність від системних бібліотек)

• залежності необов'язкові ("м'які"), без задоволення яких програма зберігає працездатність, але їх задоволення додає функціональності (наприклад, залежність links або mc від сервису консольної миші gpm).

В Debian залежності мають декілька градацій: обов'язкові (depends), настійливо рекомендовані (recommends), рекомендовані помірно наполегливо (suggests), конфліктуючі (conflicts). Перша градація — це звичайні "жорсткі" залежності. З останніми теж все зрозуміло — це, так сказати, антизалежності. Ну а наполегливо рекомендовані та рекомендовані просто — це два різновиду "м'яких" залежностей. Тобто перша категорія ніби то більш потрібна, ніж Друга. Втім, таке суб'єктивна думка мейнтейнера даного пакета — цілком можливо, що у користувача будуть інші потреби. І це ми врахуємо при виборі засобу керування пакетами.

Крім того, специфікою Debian є ще існування так званих перед-залежностей (pre-depends) — при їх порушенні встановлення пакета навіть не може початися. Втім, з точки зору користувача вони не набагато відрізняються від звичайних залежностей типу depends.

Крім залежностей, в системах пакетного менеджменту Debian важливо також поняття пріоритету пакета, що показує ступінь необхідності його для функціювання системи, наприклад: обов'язковий (required), без якого функціювання системи неможливе, основний (base) та важливий (important), також що виявляються практично необхідними, стандартний (standard) — тобто наявний практично в будь повнофункціональної Linux-системи, додатковий (optional) — тут вже ступінь важливості кожен повинен вирішувати для себе сам.

Засоби керування пакунками

Що стосується засобів керування пакунками, в Debian та його клонах є великий вибір:

• команда dpkg, призначена для встановлення, конфігурування та видалення одиничних пакунків, але не має власних засобів вирішення залежностей між ними;

• dselect — front-end (оболонка) для dpkg, що працює в текстовому режимі; забезпечує не тільки встановлення/ видалення програм, але й груповий вибір пакунків за цільовим призначенням, а також розв'язок залежностей між ними;

• механізм apt — універсальний набір інструментів для керування deb-пакунками, включаючи розв'язок залежностей між ними і навіть побудову з вихідних текстів окремих пакунків та тотальне перезібрання встановленої системи з заданими параметрами компіляції;

• aptitude — оболонка для apt, як за інтерфейсом, так і за функціоналом схожа з dselect;

Команда dpkg розв'язком залежностей не займається, лише повідомлює в відповідних випадках про їх порушення. Інструменти з набору apt обмежуються встановленням необхідних залежностей та виводять список настійливо та помірно рекомендованих пакунків. Користувач може сам визначитись, чи потрібно встановлювати йому ці залежності.

Набір apt (Advanced Packaging Tools), як видно з назви — це програмний комплекс, що охоплює всі сторони керування пакунками, аж навіть до їх побудови з вихідних текстів. Він

включає в себе майже десяток команд, з яких нині нас зацікавлять лише дві — apt-cache, засіб роботи з кешем пакунків та apt-get - інструмент для їх отримання та встановлення.

Конфігураційні файли APT

- /etc/apt/sources.list — список джерел пакунків (репозиторіїв)
- /etc/apt/apt.conf — основний файл конфігурації APT
- /etc/apt/preferences — файл уподобань, керує тим, яка версія пакунка буде встановлена у випадку наявності в репозиторії одразу декількох версій

Apt-get

#apt-get update — оновлює список доступних пакунків

Якщо ви не зробите цього час від часу, ваш локальний список доступних пакунків може застаріти. Виконуйте цю команду час від часу перед тим як запускати команду dist-upgrade або намагатися знайти новий пакунок. Списки пакунків великі і під час оновлення списків з інтернету будуть завантажені дані розміром в декілька мегабайт.

#apt-get install пакунок — встановлює вказаний пакунок

За допомогою цієї команди ви можете отримати останню версію конкретного пакунка і встановити його разом з іншими пакунками, від яких залежить його робота. Якщо запрошуваний пакунок вже встановлений, він буде оновлений до останньої доступної версії.

#apt-get upgrade — оновлює всі програми в системі

З часом більшість пакунків програмного забезпечення, які є на вашому комп'ютері, застарівають, так як з'являються нові версії, в яких додаються нові функції та виправляються помилки. Ви можете кожен раз вручну виконувати команду apt-get install foo, але це не дуже зручно, тому apt надає простий спосіб зразу оновити всю систему. Для цього наберіть команду apt-get upgrade і apt перевірить наявність нової версії для кожного пакета у вашій системі, а потім її завантажить та встановить. ця команда ніколи не буде встановлювати нові пакунки, вона призначена тільки для оновлення пакунків, які вже встановлені:

#apt-get remove пакунок — видаляє вказаний пакунок

Якщо ви раніше встановили програму і вирішили, що вона вам більше не потрібна, ви можете видалити її за допомогою цієї команди. Оскільки деякі пакунки програм можуть залежати від інших, видалення однієї програми може привести до припинення роботи інших програм. Тому коли ви запускаєте програму apt-get remove, спочатку буде перевірено, чи є інші програми, робота яких залежить від програми, що видаляється і вам буде запропоновано їх також видалити. Це тільки один приклад того, наскільки якісно були розроблені інструментальні засоби управління пакетами, наявні в Ubuntu і завдяки яким ваш комп'ютер не припинить працювати через непрацююче або не повністю встановлене програмне забезпечення. Звичайно, є ймовірність порушити роботу Ubuntu, але, в цілому, ви повинні видалити програми, які пропоновані до видалення. Команда remove також має параметр -- purge, при використанні якого видаляється не тільки сама програма, а й пов'язані з нею конфігураційні файли.

#apt-get clean — коли ви просите утиліту apt встановити пакунок програмного забезпечення, вона завантажує пакунок і перш, ніж виконати інсталяцію, зберігає його в кеші на жорсткому диску. Якщо ви потім видаліть пакунок, але пізніше передумаєте і захочете

знову його встановити його, утиліті apt не потрібно буде його шукати в інтернеті, оскільки пакунок знаходиться в локальному кеші. Це добре для економії обсягу завантажуваних даних, але через деякий час кеш може займати значний простір на жорсткому диску, так що добре періодично видаляти з нього старі пакунки. Запуск команди apt-get clean повністю очистить кеш пакунків, вивільнивши дорогоцінний дисковий простір. Ця команда абсолютно безпечна, оскільки найбільш погане, що може трапитися, це те, що утиліті apt, можливо, прийдеться завантажити пакунок знову, якщо ви його видалили, а потім знову встановлюєте.

Apt-cache

apt-cache — маніпулює кешем доступних пакунків, зазвичай використовується для пошуку пакунка та отримання інформації про нього

$apt-cache search keyword — шукає в описах доступних пакунків ключове слово. Використовується для пошуку необхідного програмного забезпечення.

$apt-cache show pkgname — виводить повну інформацію про пакунок pkgname, який нас зацікавив.

Робота з процесами

Процес — це сукупність програмного кода та даних, які завантажені в пам'ять ЕОМ. На перший погляд процес — це запущена програма (додаток) або команда. Але це не зовсім так. Деякі додатки можуть створювати декілька процесів одночасно.

Код процесу не обов'язково має виконуватися в поточний момент часу, так як процес може знаходитися в сплячому стані. В цьому випадку виконання коду такого процесу призупинено. Існує лише 3 стани, в яких може знаходитися процес:

Працюючий процес — в даний момент код процесу виконується.

Сплячий процес — в даний момент код процесу не виконується, чекає на яку небудь подію (натискання клавіші на клавіатурі, надходження даних з мережі и т.д.)

Процес-зомбі — сам процес вже не існує, його код та дані вивантажені з оперативної пам'яті, але запис в таблиці процесів залишається через ті чи інші причини.

Кожному процесу в системі призначаються числові ідентифікатори (особисті номери) в діапазоні від 1 до 65535 (**PID – Process Identifier – ідентифікатор процесу**) та ідентифікатори батьківського процесу (**PPID – Parent Process Identifier — ідентифікатор батьківського процесу**). PID є ім'ям процесу, за яким ми можемо адресувати процес в операційній системі при використанні різних засобів перегляду і керування процесами. PPID визначає родинні відносини між процесами, які в значній мірі визначають його властивості та можливості. Інші параметри, які необхідні для роботи програми, називають "оточення процесу". Одним із таких параметрів є **керуючий термінал** — ім'я термінального пристрою, на яке процес виводить інформацію і з якого інформацію отримує. Керуючий термінал мають далеко не всі процеси. Процеси, що не прив'язані до якогось конкретного терміналу називаються "демонами" (daemons). Такі процеси, будучи запущеними користувачем, не завершують свою роботу після закінчення сеансу, а продовжують працювати, так як вони не пов'язані ніяк з поточним сеансом і не можуть бути автоматично завершені. Як правило, за допомогою демонів реалізуються серверні служби, так наприклад сервер друку реалізований процесом-демоном cupsd, а сервер журналювання — syslogd.

Для перегляду списку процесів в Linux існує команда **ps**. Формат команди наступний:

ps *[PID] [options]* — перегляд списку процесів. Без параметрів ps показує всі процеси, які були запущені протягом поточної сесії, за винятком демонів. Options може приймати одне з наступних значень або їх комбінації:

-a или -e — показати всі процеси

-f — повний лістинг

-w — показати повні строки опису процесів.

Процеси в ОС Linux володіють тими ж правами, якими володіє користувач, від імені якого був запущений процес.

Насправді операційна система сприймає користувача, що в ній працює, як набір запущених від його імені процесів. Адже і сам сеанс користувача відкривається в командній оболонці (або оболонці X) від імені користувача. Тому коли ми кажемо "права доступу користувача до файлу" то маємо на увазі "права доступу процесів, запущених від імені користувача до файлу".

Для визначення імені користувача, що запустив процес, операційна система використовує **реальні ідентифікатори користувача та групи**, які призначаються процесу. Але ці ідентифікатори не є вирішальними при визначенні прав доступу. Для цього в кожного процесу існує інша група ідентифікаторів — **ефективні**.

Як правило, реальні та ефективні ідентифікатори процесів однакові, але є і виключення. Наприклад, для роботи утиліти passwd необхідно використовувати ідентифікатор суперкористувача, так як тільки суперкористувач має права на запис у файли паролів. У цьому випадку ефективні ідентифікатори процесу будуть відрізнятися від реальних. Виникає резонне питання — як це було реалізовано?

У кожного файлу є набір спеціальних прав доступу - біти SUID і SGID. Ці біти дозволяють при запуску програми привласнити їй ефективні ідентифікатори власника та групи-власника відповідно і виконувати процес з правами доступу іншого користувача. Так як файл passwd належить користувачу root і у нього встановлений біт SUID, то при запуску процес passwd буде мати права користувача root.

Батьком всіх процесів в системі є процес init. Його PID завжди 1, PPID — 0. Всю таблицю процесів можна уявити собі у вигляді дерева, в якому коренем буде процес init. Цей процес хоч і не є частиною ядра, але виконує в системі дуже важливу роль — визначає поточний рівень ініціалізації системи і стежить щоб були запущені програми, що дозволяють користувачеві спілкуватися з комп'ютером (mingetty, X або інші).

Процеси, імена яких укладені в квадратні дужки, наприклад "[keventd]" — це процеси ядра. Ці процеси керують роботою системи, а точніше такими її частинами, як менеджер пам'яті, планувальник часу процесора, менеджери зовнішніх пристроїв і так далі.

Решта процеси є користувацькими, запущеними або з командного рядка, або під час ініціалізації системи.

Життя кожного процесу представлена наступними фазами:

Створення процесу — на цьому етапі створюється повна копія того процесу, який створює новий. Наприклад, ви запустили з інтерпретатора на виконання команду ls. Командний інтерпретатор створює свою повну копію.

Завантаження коду процесу та підготовка до запуску — копія, створена на першому етапі замінюється кодом завдання, яке необхідно виконати і створюється її оточення — встановлюються необхідні змінні і т.п.

Виконання процесу

Стан зомбі — на цьому етапі виконання процесу закінчилося, його код вивантажується з пам'яті, оточення знищується, але запис у таблиці процесів ще залишається.

Вмирання процесу — після всіх завершальних стадій видаляється запис з таблиці процесів — процес завершив свою роботу.

Для керування пріоритетом виконання процесу служить команда nice

nice — UNIX-утиліта, що запускає програму зі зміненим пріоритетом для планувальника завдань. Якщо не вказано жодного аргументу, команда nice виводить поточний успадкований пріоритет для планувальника завдань. В іншому випадку nice запускає вказану команду зі зміненим пріоритетом. Якщо зсув не вказано, то пріоритет команди збільшується на 10. Привілейований користувач (root) може вказати негативний зсув. Команда nice може зміщати пріоритет в діапазоні від -20 (найвищий пріоритет) до 19 (нижчий пріоритет) від поточного.

Командою, що найбільш часто використовується для керування процесами можна по праву вважати команду kill:

kill -SIGNAL pid — надсилає сигнал процесу з ідентифікатором pid. Якщо сигнал не вказаний, команда надсилає процесу сигнал SIGTERM. Ось приклад її використання:

kill -SIGKILL 1352

Не менш популярною командою ніж kill є killall:

killall -s SIGNAL процес — надсилає сигнал всім процесам з іменем процес. Якщо сигнал не вказаний — надсилає процесу сигнал SIGTERM

killall -9 konsole

Завантаження системи. Рівні запуску.

Термін рівень виконання означає режим функціювання операційної системи комп'ютера, в якій реалізована ініціалізація в стилі ОС Unix System V. Традиційно існують сім рівнів виконання, пронумерованих від 0 до 6.

Для запуску сервісу при старті Linux необхідно переконатися, що існує необхідний сервіс в теці /etc/init.d/. Приклад скрипту для запуску самописного сервісу можна подивитися в /etc/init.d/skeleton. Після цього можна додати сервіс при завантаженні створивши сімлінк в потрібному рівні запуску /etc/rcX.d/, де X може мінятися від 0…6.

Формат іменування скриптів
S[порядковий номер][ім'я] - ім'я скрипту для запуску сервісу
K[порядковий номер][ім'я] — ім'я скрипту для зупинки сервісу
Порядковий номер може змінюватись в межах 1…99, чим вище порядковий номер, тим пізніше запуститься сервіс.

Після запуску всіх сервісів для вказаного рівня ініціалізації (runlevel) виконується скрипт /etc/rc.local, який містить користувацькі сценарії запуску.

Рівні ініціалізації в Debian (Ubuntu):
0 — зупинка
1 — однокористувацький режим
2-5 — багатокористувацький режим
6 - перезавантаження
За замовчуванням використовується рівень 2.

Upstart

Проект upstart створено з метою замінити традиційний спосіб завантаження користувацьких процесів в Linux, а також спосіб керування ними.

Як і в класичному варіанті — upstart має головний процес, який відповідає за завантаження та управління демонами. Називається він також як і в класичному варіанті — init.

В upstart ввели таке поняття як "завдання" (jobs). Якщо під час запуску процес init читає конфігурацію з файлу /etc/inittab, то процес upstart init читає конфігурацію з файлів теки /etc/init/. Ці файли і є файлами-завданнями (jobs). Кожен файл в теці /etc/init/ відповідає за запуск окремого демона або сервісу і повинен закінчуватися на .conf. Також допускається, створювати підтеки, в яких можуть знаходиться файли-завдання.

Синтаксис файлів-завдань нескладний. У файлах-завданнях обов'язково міститься інформація коли завдання повинно стартувати (визначення start on), коли зупинятися (stop on) і що запускати (ключові слова exec або script).

Отже, коли ж завдання спрацює? Ось тут і є перша основна відмінність яке вирішує проблему послідовного запуску скриптів. upstart — це підсистема яка реагує на події. Тому

після ключових слів start on або stop on, обов'язково вказується подія (event), при настанні якої починає виконуватися запуск або зупинка завдання.

Якщо розглядати події найбільш часто вживані в файлах завдань, то це startup, runlevel, stopped і started. startup — це найперше подія, яка розпізнається підсистемою upstart на самому ранньому етапі завантаження операційної системи. Подія runlevel (із зазначенням цільового рівня), як випливає з назви генерується при зміні рівня запуску. Подія stopped генерується після зупинки вказаного завдання, а started після завершення старту завдання. Переглянувши файли-завдання в теці /etc/init/ можна знайти й інші події.

Таким чином якщо необхідно створити завдання яке буде запускати тестовий демон test на 2-му і 3-му рівні запуску і зупиняти на інших рівнях, достатньо створити файл test.conf в якому написати:

start on runlevel [23]
stop on runlevel [!23]
exec test

Для ручного запуску чи зупинки завдань в upstart передбачені команди start і stop, розташовані, як правило, у теці /sbin. Щоб в Ubuntu зупинити демон cron достатньо ввести команду:

sudo stop cron.
Щоб запустити cron вводимо:
sudo start cron

З інших команд керування можна відзначити команду status, яка показує в якому стані (зупинки або роботи) знаходиться завдання (а відповідно і демон).

Також слід окремо згадати команду initctl. Ця команда як випливає з її назви (скорочене від init control), дозволяє управляти демоном upstart init. Команда initctl теж уміє зупиняти (initctl stop) і запускати (initctl start) завдання, а також перевіряти їх статуси (initctl status). Якщо потрібно подивитися статус всіх завдань є команда initctl list.

Upstart переглядає і виконує тільки файли-завдання розташовані в теці /etc/init/, тому для того, щоб виконувалися скрипти з тек /etc/rcX.d/, необхідно "сказати" про це upstart. Для цього в теці /etc/init/ знаходиться файл-завдання rc.conf. Саме завдяки цьому файлу і відбувається перегляд і запуск скриптів для стандартного init з теки /etc/rcX.d/

Виконуючи цей файл-завдання, **upstart** запускає скрипт **/etc/init.d/rc** з передачею йому рівня запуску ($RUNLEVEL).

А далі вже по класичній схемі — скрипт **rc** починає виконувати скрипти з відповідної теки **/etc/rcX.d/**.

Керування сценаріями запуску SysV

Для керування запуском різних сервісів існують різні утиліти. Кожен може використовувати те, що йому зручніше.

update-rc.d

update-rc.d входить в стандартну поставку Debian і заснованих на ньому системах (в тому числі і Ubuntu). Ця утиліта призначена для створення і видалення посилань на скрипти автозапуску. Можливості її максимально прості — дійсно тільки встановити або видалити посилання на скрипт в теці відповідного рівня завантаження. Наприклад:

#update-rc.d -f ssh remove

Видаляємо демон ssh з автозавантаження повністю.

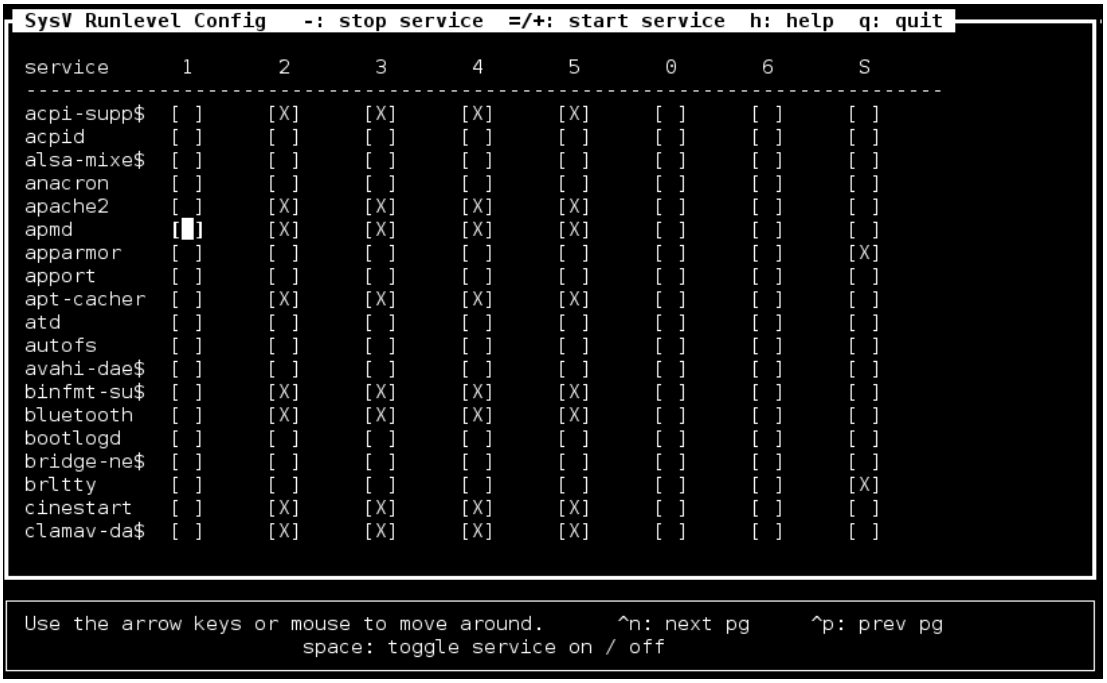
#update-rc.d foobar start 30 2 3 4 5 . stop 70 0 1 6 .
Запускаємо скрипт foobar на 2, 3, 4 і 5 рівнях з порядковим номером 30 і зупиняємо на рівнях 0, 1 і 6 з порядковим номером 70

Недолік цієї утиліти в тому, що переглянути списком що саме і на якому рівні запускається не вийде. Не призначена вона для цього.

Sysv-rc-conf

Sysv-rc-conf за замовчуванням не встановлено в системі. Для установки слід виконати:
#apt install sysv-rc-conf

Утиліта під час запуску без параметрів має консольний інтерфейс, в якому видно, які



скрипти і на якому рівні будуть запускатися. Свій скрипт, що лежить в /etc/init.d/ додається в автозавантаження звичайним позначенням потрібних рівнів запуску.

Так само можна її використовувати з параметрами командного рядка.

Показати рівні запуску всіх скриптів
#sysv-rc-conf --list

Показати рівні запуску конкретного сервісу (Postfix)
#sysv-rc-conf --list postfix

Відключити ssh на рівнях 3 і 5
 # sysv-rc-conf --level 35 ssh off

Додати сервіс eTSrv на рівні завантаження 2 і 3)
#sysv-rc-conf --level 23 eTSrv on

Якщо опустити параметр --level, то за замовчуванням будуть використовуватися рівні 2, 3, 4 і 5.

service

service — це спеціальна утиліта для запуску скриптів System V init. Запуск цих скриптів проводиться в заздалегідь визначеному середовищі з видаленням більшості змінних оточення та встановленою робочою текою скрипту в "/"

Команда service має прості параметри:

Запустити всі init-скрипти в алфавітному порядку з параметром status, тобто подивитися поточний статус всіх сервісів
#service --status-all

Зупинити сервіс ssh
#service ssh stop

Запустити сервіс ssh
#service ssh start

Перезапустити сервіс ssh
#service ssh restart

SystemD

SystemD — системний менеджер, демон ініціалізації інших демонів в Linux, що прийшов на заміну SysV. Його особливістю є інтенсивне розпаралелювання запуску служб в процесі завантаження системи, що дозволяє істотно прискорити запуск операційної системи. Назва походить від прийнятого в Unix додавання суфікса «d» до демонів.

Systemd розвивають Леннарт Поттерінг, Кей Сіверс та інші розробники. Опублікований як вільне програмне забезпечення під ліцензією GNU Lesser General Public License версії 2.1

Механізм systemd має зворотну сумісність з SysV і з Upstart.

Systemd оперує спеціально оформленими файлами конфігурації - юнітами (unit). Кожен юніт відповідає за окремо взяту службу, точку монтування, пристрій, що підключається, файл підкачки, віртуальну машину і т.п. Існують спеціальні типи юнітів, які не несуть функціонального навантаження, але дозволяють задіяти додаткові можливості systemd. До них відносяться юніти типу target, slice, automount і ін. Systemd підтримує такі типи юнітів:

- .target - дозволяє групувати юніти, втілюючи концепцію рівнів запуску (runlevel)
- .service - відповідає за запуск сервісів (служб), також підтримує виклик інтерпретаторів для виконання скриптів
- .mount - відповідає за монтування файлових систем
- .automount - дозволяє відкласти монтування файлових систем до фактичного звернення до точки монтування

- .swap - відповідає за підключення файлу або пристрою підкачки
- .timer - дозволяє запускати юніти за розкладом
- .socket - надає службам підтримку механізму сокет-активації
- .slice - відповідає за створення контейнера cgroups
- .device - дозволяє реагувати на підключення пристроїв
- .path - керує ієрархією файлової системи

У порівнянні з SysV, systemd дає переваги в наступному:

- Контроль стану служби, реакція на зміни стану.
- Сокет-активні і шина-активні служби, які іноді приводять до кращого розпаралелювання взаємозалежних служб.
- cgroups використовується для відстеження службових процесів, замість ідентифікаторів процесів (PID). Це означає, що демони не будуть втрачені навіть після розпаралелювання в інші процеси.

Основні команди systemd

Список запущених юнітів:
systemctl
Юніти, запуск яких завершився невдачею:
systemctl --failed
Список доступних юнітів:
systemctl list-unit-files
Запуск юніта:
systemctl start <unit-name>
Зупинка юнита:
systemctl stop <unit-name>
Перезавантаження юніта:
systemctl restart <unit-name>
Перезавантаження налаштувань юніта:
systemctl reload <unit-name>
Переглянути статус юніта:
systemctl status <unit-name>
Перевірити чи дозволений запуск юніта при старті системи:
systemctl is-enabled <unit-name>
Дозволити запуск юніта при старті системи:
systemctl enable <unit-name>
Заборонити запуск юніта при старті системи
systemctl disable <unit-name>
Перезавантаження systemd з пошуком змінених або нових юнітів:
systemctl daemon-reload

Керування живленням

Перезавантажити комп'ютер:
systemctl reboot
Вимкнути комп'ютер:
systemctl poweroff
Сплячий режим:
systemctl suspend

Режим очікування:
systemctl hibernate
Гібридний сон (suspend-to-both)
systemctl hybrid-sleep

Журнал (ведення, читання логів)

Для регулювання розміру файлу логів, потрібно відредагувати */etc/systemd/journald.conf*
SystemMaxUse=100M
За замовчуванням розмір файлу логів обмежений в 10% від розміру файлової системи де він розташований (*/var/log/journal*)
Читання всіх логів:
journalctl
Логи з моменту запуску системи:
journalctl -b
Якщо був крах системи, можна ввести параметр -1 і подивитися логи з попереднього запуску системи (-2 з двох попередніх і т.д):
journalctl -b -1
Виведення останнього запису:
journalctl -f
Всі повідомлення конкретної утиліти, наприклад **systemd**:
journalctl /usr/lib/systemd/systemd
Всі повідомлення конкретного процесу:
journalctl _PID=1
Всі повідомлення конкретного юніта:
journalctl -u netcfg

Команда sudo. Підвищення прав користувачів.

sudo (англ. superuser do, дослівно «виконати від імені суперкористувача») — програма для системного адміністрування UNIX-систем, що дозволяє делегувати ті чи інші привілейовані ресурси користувачам з веденням протоколу роботи. Основна ідея — дати користувачам як щонайменше прав, при цьому достатньо для рішення поставлених задач. Програма існує для більшості UNIX та UNIX-подібних операційних систем.

Команда sudo надає можливість користувачам виконувати команди від імені суперкористувача root, або інших користувачів.

В більшості випадків правильне налаштування sudo робить непотрібною роботу від імені суперкористувача. Всі дії можна виконати з-під облікового запису звичайного користувача, хоча стиль роботи може бути незручним тля тих, хто звик працювати під root..

Програма *sudo* конфігурується в файлі */etc/sudoer*s. Редагувати його рекомендується тільки за допомогою спеціального редактору *visudo*. Базовий синтаксис (розділювач в рядку, знак ","):

user hosts = (runas) commands

- **users** один або більше користувачів або група (наприклад %wheel) для розширення прав доступу
- **hosts** список хостів (або ALL) Якщо ви не системний адміністратор групи з декількох робочих станцій Лінукс, то не вам це не потрібно. А якщо потрібно, то доведеться скопіювати файл за допомогою команди rsync на інші хости, або використовувати що-небудь на зразок Network Information Service (NIS), щоб забезпечити кому треба доступ до цього файлу.
- **runas** список користувачів (або ALL) від чийого імені можуть виконуватися команди. Заключається в ()!
- **commands** список команд (або ALL), які можна запустити від імені root або від імені інших користувачів

Крім того існують псевдоніми *User_Alias, Host_Alias, Runas_Alias* и *Cmnd_Alias*

Ось невеликий приклад файлу */etc/sudoers :*

 # Псевдоніми користувача, список користувачів мають деякі права доступу
User_Alias ADMINS = user1, user2, admin

 # Псевдоніми команд, список повних шляхів до команд
Cmnd_Alias PW = /usr/bin/passwd [A-z], !/usr/bin/passwd root*
 # Not root pwd!

 # Актуальні права доступу
root,ADMINS ALL = (ALL) NOPASSWD: ALL
 # ADMINS може все робити без паролю

%admin ALL=(ALL) ALL
#Користувачі системної групи admin мають повний доступ до підвищення прав, але з паролем

user ALL=(ALL) PW

Система якийсь час пам'ятає введений пароль (за замовчуванням 15 хвилин). Тому при подальших виконаннях sudo введення пароля може не знадобитися. Для гарантованого припинення сесії sudo наберіть в терміналі:

$ sudo -K

Час запам'ятовування пароля, звичайно, можна змінити. Для цього в файл */etc/sudoers треба* додати рядок:
Defaults:ALL timestamp_timeout=25 #время в минутах

Крім того, часто зустрічаються помилки, пов'язані із застосуванням цієї команди. При виконанні

$sudo cat sources.list > /etc/apt/sources.list

буде видана помилка прав доступу (так як з правами root виконується тільки процес cat, а перенаправлення виконує shell з правами звичайного користувача), хоча таке можна зробити, використавши конвеєр:

$ cat sources.list | sudo tee /etc/apt/sources.list

Налаштування мережі

Спосіб 1. Для старших дистрибутивів.

Для налаштування мережі в консолі у всіх Linux системах, включаючи Ubuntu, існує спеціальна команда ifconfig. Якщо в терміналі просто написати цю команду то консоль видасть всі мережеві інтерфейси, які запущені на цьому комп'ютері. Це буде виглядати приблизно так:

```
eth0      Link encap:Ethernet  HWaddr 00:26:2d:a4:d8:c2
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:16

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:423 errors:0 dropped:0 overruns:0 frame:0
          TX packets:423 errors:0 dropped:0 overruns:0 carrier:0
```

collisions:0 txqueuelen:0
RX bytes:30194 (30.1 KB) TX bytes:30194 (30.1 KB)

wlan0 Link encap:Ethernet HWaddr 20:7c:8f:01:11:97
inet addr:192.168.7.55 Bcast:192.168.7.255 Mask:255.255.255.0
inet6 addr: fe80::227c:8fff:fe01:1197/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:881137 errors:0 dropped:0 overruns:0 frame:0
TX packets:520917 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:1243341370 (1.2 GB) TX bytes:51051482 (51.0 MB)

Перший стовпчик це імена інтерфейсів, в другій налаштування відповідних інтерфейсів. Команда *ifconfig eth0* виведе налаштування лише інтерфейсу eth0.

Для вимкнення інтерфейсу eth0 служать команди:
#ifconfig eth0 down
або
#ifdown eth0

Для ввімкнення
#ifconfig eth0 up
або
#ifup eth0

Також можна змінювати IP-адресу мережевого пристрою
ifconfig eth0 inet 192.168.1.1
або його MAC-адресу
#ifconfig eth0 hw ether 00:12:34:56:78:90

Однак налаштування, зроблені таким чином, збережуться лише до перезавантаження комп'ютера.

Для того, щоб цього не трапилось потрібно прописати необхідні налаштування в конфігураційному файлі. В Debian та Debian-подібних дистрибутивах це буде */etc/network/interfaces*

Даний файл виглядає приблизно наступним чином:

auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.7.20
hwaddress ether 00:12:34:56:78:90
netmask 255.255.255.0
network 192.168.7.0
broadcast 192.168.7.255
gateway 192.168.7.1

auto eth1
iface eth1 inet dhcp

Після зміни цього конфігураційного файлу перезавантажте мережеву службу командою:
sudo service networking restart

Для зупинки або запуску мережевої служби користуйтеся командою:
sudo service networking stop
sudo service networking start

Спосіб 2. Для нових дистрибутивів.

В сучасних дистрибутивах команда ifconfig замінена на команду ip. За її допомогою можна не лише налаштовувати IP адреси мережевих інтерфейсів, але й керувати маршрутизацією, ARP-таблицями і таке інше.

Інакше кажучи, на заміну утилітам ifconig, route, arp. прийшла єдина утиліта ip яка має стандартизований набір параметрів і більш логічний опис.

Синтаксис утиліти ip

ip [опції] об'єкт команда [параметри]

Опції

-v - тільки виведення інформації про утиліту і її версію
-s - містить виведення статистичної інформаціи
-o - виводити кожен запис з нового рядка
-f - позволяет указать протокол, с которым нужно работать, если протокол не указан, то берется на основе параметров команды. Опция f должна принимать одно из значений: bridge, dnet, inet, inet6, ipx или link. По умолчанию используется inet, link - означает отсутствие протокола.
Все остальные опции - ярлыки опции -f:
-4 - ярлык для -f inet
-6 - -f inet6
-B - -f bridge
-0 - -f link

об'єкти

- address - мережева адреса на пристрої
- link- фізичний мережевий пристрій
- monitor - моніторинг стану пристроїв
- neigh - ARP
- route - керування маршрутизацією
- rule - правила маршрутизації
- tunnel - налаштування тунелювання

команди
add, change, del або delete, flush, get, list чи show, monitor, replace, restore, save, set, та update.

Якщо команда не задана, за умовчанням використовується show (показати).

параметри
Параметри залежать від об'єкта і зазначеної команди. Розглянемо основні:

- dev ім'я_пристрою - мережевий пристрій
- up - включити
- down - вимкнути
- llarrr - MAC-адреса
- via - через роутеру
- default - маршрут за замовчуванням
- blackhole - маршрут "чорна діра" - відкидати пакети і не посилати ICMP повідомлення про недоступність
- prohibit - маршрут "заборони" - відкидати пакети і повертати ICMP повідомлення про заборону доступу

Зверніть увагу - утиліта ip дозволяє скорочувати назви назви об'єктів і команд, наприклад всі ці команди рівнозначні:
ip address show
ip address sh
ip address
ip addr
ip a

Приклади використання

ip link
Утиліта ip в Linux з параметром link дозволяє керувати станом мережевих інтерфейсів і переглядати інформацію про них.

- ip link show - показати стан всіх мережевих інтерфейсів
- ip link show eth0 - показати стан eth0
- ip link list up - відобразити статус всіх увімкнутих інтерфейсів
- ip link set eth1 up - увімкнути eth1
- ip link set eth1 down - вимкнути eth1

ip neigbour
Об'єкт neigbour використовується для керування ARP таблицями.

- ip neigh show - показати всі записи ARP
- ip neigh show dev eth0 - подивитися всі ARP записи для eth0
- ip neigh flush - видалити всі ARP записи
- ip ne fl dev eth0 - видалити всі ARP записи для eth0
- ip nei add 1.1.1.13 lladdr AA:BB:CC:DD:EE:FF dev eth0 - додати ARP запис для певної IP адреси.
- ip n del 1.1.1.13 dev eth0 - видалити всі записи для вказаної адреси

ip address
Треба зауважити, що вторинні ip адреси не використовуються як вихідні адреси для відправки пакетів.

- ip address show - показати все ip адреси та їх інтерфейси
- ip a l permanent - показати тільки статичні ip адреси
- ip a l dynamic - показати тільки динамічні ip адреси
- ip addr add 1.1.1.13/24 dev eth0 - встановити ip адресу для інтерфейсу eth0
- ip addr del 1.1.1.13/24 dev eth0 - видалити ip адресу інтерфейсу eth0
- ip add flush dev eth0 - видалити всі ip адреси інтерфейсу eth0

ip route

Утиліта ip в Linux дозволяє не тільки встановлювати ip адреси, а й налаштовувати маршрути. За замовчуванням в Linux використовується таблиця маршрутизації 254.

- ip route show - показати всі маршрути в таблиці маршрутизації
- ip route show table 255 - показати всі маршрути з таблиці 255
- ip route get 10.10.20.0/24 - показати маршрут до цієї мережі
- ip route get 10.10.20.0/24 from 192.168.12.9 - показати маршрут до цієї мережі від зазначеного інтерфейса.
- ip route add 10.10.20.0/24 via 192.168.50.100 - створити маршрут
- ip route delete 10.10.20.0/24 - видалити маршрут.
- ip route del 10.10.20.0/24 via 192.168.50.100 - видалити маршрут.
- ip route add default via 192.168.50.100 - створити маршрут за замовчуванням.
- ip route add 10.10.20.0/24 dev eth0 - створити маршрут до вказаної мережі.
- ip route add table nnn 10.10.20.0/24 dev eth0 - створити маршрут в спеціальній таблиці маршрутизації.

Налаштування мережі в файлі конфігурації

Розробники перестали використовувати класичний /etc/init.d/networking і / і /etc/network/interfaces для конфігурації мережі і переключилися на дещо під назвою netplan. Тепер файли конфігурації мережевих інтерфейсів знаходяться в теці /etc/netplan і мають формат yaml.
Приклад файлу конфігурації:

```
network:
  ethernets:
    enp0s3:
      addresses: []
      dhcp4: true
       enp0s8:
      addresses: [192.168.56.10/24]
#       gateway4: 192.168.1.1
#       nameservers:
#          addresses: [8.8.8.8,8.8.4.4]
```

```
        dhcp4: no
    version: 2
```

Тут описаний приклад конфігурування двох інтерфейсів — статичні налаштування для enp3s8 та автоматичні налаштування через DHCP для інтерфейса enp0s3.

Треба бути дуже уважним при редагуванні цього файлу — в даному випадку для формату yaml не допускається вирівнювання табуляцією — виключно пробілами. Крім того кожен логічний блок має бути однаково вирівнянним.

Для застосування налаштувань використовується команда

 # netplan apply

Але, так як налаштування ускладнились, розробники дали можливість застосовувати налаштування тимчасово. Для цього треба виконати команду
 # netplan try

При виконанні цієї команди з’являється текст
Do you want to keep these settings?
Press ENTER before the timeout to accept the new configuration

І йде зворотній рахунок від 120 секунд.
Якщо ніяких дій проведено не було, то будуть повернені попередні налаштування.

Налаштування SSH

SSH (англ. Secure SHell — «безпечна оболонка») — мережевий протокол сеансового рівня, який дозволяє проводити віддалене керування операційною системою та тунелювання TCP-з'єднань (наприклад, для передачі файлів). Подібний за функціональністю з протоколами Telnet і rlogin, але, на відміну від них, шифрує весь трафік, включаючи і паролі, що передаються. SSH допускає вибір різних алгоритмів шифрування. SSH-клієнти і SSH-сервери доступні для більшості мережевих операційних систем.

SSH дозволяє безпечно передавати в незахищеному середовищі практично будь-який інший мережевий протокол. Таким чином, можна не тільки віддалено працювати на комп'ютері через командну оболонку, але й передавати по шифрованому каналу звуковий потік або відео. Також SSH може використовувати стиснення даних, що передаються, для подальшого їх шифрування, що зручно, наприклад, для віддаленого запуску клієнтів X Window System.

Ssh захищає від:

- IP spoofing'y [ip-підміни], коли віддалений комп'ютер висилає свої пакети симулюючи нібито вони прийшли з іншого комп'ютера, з якого дозволений доступ. Ssh захищає від підміни навіть в локальній мережі, коли хтось наприклад, вирішив підмінити ваш роутер назовні собою.
- IP source routing, коли комп'ютер може симулювати що IP-пакети приходять від іншого, дозволеного комп'ютера.
- DNS spoofing, коли атакуючий фальсифікує записи name server'a
- Прослуховування нешифрованих паролів і інших даних проміжними комп'ютерами.
- Маніпуляцій над вашими даними людьми керуючими проміжними комп'ютерами.
- Атак заснованих на прослуховуванні X authentication data і підробки з'єднання до X11 серверу.

Іншими словами, ssh ніколи не довіряє мережі; будь-який "противник" який має певний доступ до мережі може лише насильно розірвати встановлене ssh з'єднання, але ніяк не розшифрувати його.

Однак ssh ніяк не захищає від атак на перебір пароля (брутфорс-атак), метою яких служить отримання доступу до системи.

Так само цей протокол не може захистити від слабких паролів, встановлених адміністратором системи.

Встановлення та налаштування сервера ssh

Виконаємо встановлення ssh-сервера:

$sudo apt install ssh

Відкриємо на редагування файл конфігурації ssh-сервера — */etc/ssh/sshd_conf*

В файлі конфігурації встановимо заборону віддаленого входу в систему з правами root. Для цього змінну *PermitRootLogin* встановимо в *no*.

Для більшої безпеки явно визначимо список користувачів, яким дозволяється віддалене підключення до сервера. Для цього додамо в файл конфігурації строку:

AllowUsers user

user — ім'я нашого користувача

Для застосування змін необхідно перезапустити ssh-сервер:

$sudo service ssh restart

При віддаленому адмініструванні серверів, дуже зручно використовувати авторизацію по парі ключів, що б не вводити пароль при кожному з'єднанні з віддаленою машиною. Для з'єднання з віддаленим хостом без введення пароля, можна використовувати так званий Public key (публічний ключ). Потрібно додати запис про публічний ключі в файл ~/.ssh/authorized_keys на віддаленому SSH сервері.

1. Enter file in which to save the key (/root/.ssh/id_rsa): — погоджуємося на значення за замовчуванням.

   $ssh-keygen -t rsa

   Відповіжаємо на питання:

2. Enter passphrase (empty for no passphrase): — залишаємо значення порожнім. Паролем користуватися не будемо.
3. Enter same passphrase again: — знову залишаємо значення порожнім.

Скрипт створив 2 ключа: приватний і публічний.
Your identification has been saved in /home/user/.ssh/id_rsa. — секретний приватний ключ для декодування.

Your public key has been saved in /home/user/id_rsa.pub. — публічний ключ для кодування.

Зараз потрібно скопіювати на сервер наш публічний ключ:
$ssh-copy-id -i ~/.ssh/id_rsa.pub user@server

Все. Тепер спробуємо залогінитися:
$*ssh user@server*

Як ми бачимо, тепер при вході на сервер нас не питають про пароль.

Встановлення та налаштування DenyHosts

Для того, щоб захистити нашу систему від підбирання пароля по ssh встановимо програму DenyHosts. Нажаль зараз ця програма в репозиторії не входить, тому встановлюємо її наступним чином:

Якщо програма є в репозиторіях, то встановлюємо її командою:
 # apt install denyhosts

В іншому випадку завантажуємо архів з вихідними кодами
$ cd /tmp/
$ wget http://downloads.sourceforge.net/project/denyhost/denyhost-2.8/denyhosts-2.8.tar.gz

Розархівуємо його
$ tar xzf denyhosts.tar.gz*

Змінимо теку
*$ cd DenyHosts**

Та, нарешті, встановимо програму
$ sudo python setup.py install
Тепер треба скопіювати скрипт автозавантаження в потрібне місце
sudo cp /usr/local/bin/daemon-control-dist /etc/init.d/denyhosts

Також в файлі конфігурації /etc/init.d/denyhosts змінимо значення основного файлу
DENYHOSTS_BIN = "/usr/local/bin/denyhosts.py"

Для того, щоб сервіс при завантаженні системи стртував автоматично, потрібно його включити на 2, 3, 4 та 5 рівнях за допомогою, наприклад, *sysv-rc-conf*

Тепер налаштуємо DenyHosts. Для цього відкриємо на редагування файл */etc/denyhosts.conf* та змінимо значення наступних змінних:

PURGE_DENY	Змінну PURGE_DENY залишимо порожньою, щоб хости, які нас пробували ламати, ніколи не видалялися з чорного списку. При необхідності їх завжди можна додати в білий список, який має більший пріоритет.
BLOCK_SERVICE	Цю змінну має сенс виставити в «all», тому що якщо з цього хоста почалися атаки на ssh, то можуть початися і будь-які інші.
DENY_TRESHHOLD_INVALID	Змінна показує, скільки разів можна спробувати підібрати пароль до неіснуючого користувача. Рекомендується значення виставити в «5».
DENY_TRESHHOLD_VALID	Змінна показує, скільки разів можна спробувати підібрати пароль до існуючого користувача. Рекомендується значення виставити в «3».
DENY_TRESHHOLD_ROOT	Змінна показує, скільки разів можна спробувати підібрати пароль до користувача root. Рекомендується значення виставити в «1». Хоча, в разі заборони з'єднання для root в налаштуваннях ssh-сервера, ця змінна особливого значення не має.
SYNC_SERVER	Розкоментуємо рядок зі змінною SYNC_SERVER, для обміну списком поганих хостів з центральним сервером програми.
SYNC_INTERVAL	Показує проміжок часу для синхронізації. Встановимо значення цієї змінної в «1h».
SYNC_UPLOAD	Змінна показує, чи можна завантажувати на центральний сервер список хостів, які попалися нам. Рекомендується встановити в «yes», що б інші користувачі могли від них превентивно захиститися.
SYNC_DOWNLOAD	Змінна показує, чи можна завантажувати з сервера список поганих хостів. За замовчуванням має значення «yes». Рекомендується так і залишити.

Інші параметри рекомендується не змінювати.
Після закінчення налаштування необхідно перезапустити програму.
$sudo service denyhosts restart

Встановлення та налаштування проксі-сервера Squid

У багатьох організаціях виникає необхідність забезпечити кожного співробітника загальним доступом в Інтернет через один інтернет канал. Для виконання цього завдання використовується інтернет-шлюз. При настройці програмного інтернет-шлюзу (проксі-сервера), крім можливості зробити загальний Інтернет для всіх користувачів, клієнт також отримує повнофункціональний контроль інтернет трафіку і надійний бар'єр проти мережевих погроз. Загальний доступ в Інтернет через сервер дозволяє контролювати витрати на Інтернет, а також забезпечувати безпеку мережі.

Установка інтернет-шлюзу для організації дозволить здійснити наступний контроль:

Обмеження доступу в Інтернет

- обмеження Інтернету по швидкості для кожного користувача;
- обмеження доступу в Інтернет для завантаження файлів (наприклад: mp3, avi, jpg, exe і т.д.)

Обмеження сайтів

У багатьох організаціях чималу частину свого робочого часу співробітники витрачають на перегляд сайтів, які абсолютно не відносяться до їх професійної діяльності (розважальні портали, сайти знайомств, пошук роботи, онлайн-ігри і т.д.).

За допомогою інтернет-шлюзу можливо ввести такі обмеження:

- обмеження сайтів з пошуку роботи;
- обмеження сайтів для перегляду відео роликів;
- обмеження сайтів, які створені для спілкування;
- обмеження сайтів для завантаження аудіо та відео файлів;
- обмеження сайтів для online ігор

Антивірусна перевірка інтернет-трафіку

Більшість вірусів потрапляють в комп'ютери користувачів через загальний доступ в Інтернет. При зараженні комп'ютера, найчастіше вірус видаляє або підміняє системні файли, що призводить до непрацездатності операційної системи, а нерідко і до втрати важливих документів.

При установці Інтернет-шлюзу весь загальний інтернет-трафік перевіряється вбудованим антивірусом в режимі реального часу. Використання інтернет-шлюзу для забезпечення загального доступу в Інтернет на підприємстві забезпечує додатковий антивірусний захист комп'ютерів.

Squid — програмний пакет, який реалізує функцію кешуючого проксі-сервера для протоколів HTTP, FTP і HTTPS. Розроблений співтовариством як програма з відкритим вихідним кодом (поширюється відповідно до ліцензії GNU GPL).

Використовується в UNIX системах і в операційних системах сімейства Windows NT. Має можливість взаємодії з Active Directory шляхом аутентифікації через LDAP, що дозволяє використовувати розмежування доступу до інтернет ресурсів користувачів, які мають облікові записи на Windows Server, також дозволяє організувати «нарізку» інтернет трафіку для різних користувачів.

Встановлення та первинне налаштування проксі-сервера Squid

Установка здійснюється виконанням команди:
#apt install squid

Після закінчення встановлення для його налаштування відкриваємо на редагування файл конфігурації */etc/squid/squid.conf*

Знаходимо опис acl і додаємо туди опис нашої мережі, яку і буде обслуговувати наш проксі-сервер:

acl mynetwork src 10.10.10.0/24

Де 10.10.10.0/24 — це наша внутрішня мережа з маскою 24

Тепер знаходимо рядок *http_access deny all* і перед ним дописуємо:
http_access allow mynetwork

Зверніть увагу! Всі правила пропуску трафіку в Squid виконуються по черзі, і якщо з'єднання вже потрапило під якесь дозволяюче або заборонне правило, то виконується зазначена дія і далі ланцюжок правил не перевіряється. Саме тому з метою безпеки слід останнім правилом ставити *http_access deny all.*

Перезапускаємо наш проксі-сервер командою:
#service squid restart

Після того, як Squid перезапуститься, можна в налаштуваннях браузера клієнтської машини встановити його параметри і перевіряти, як він працює.

Фільтрація завантажень файлів за розширенням

Проксі-сервер Squid дозволяє забороняти завантаження різних типів файлів, виходячи з їх розширення. Це дозволяє знизити завантаження каналу, заборонивши різний мультимедіа-контент.

Для того, щоб зробити відповідні налаштування, у файл конфігурації в блок опису acl впишемо рядок:

acl multimedia urlpath_regex -i \.avi$ \.mp3$

Тепер перед рядком *http_access allow mynetwork* додамо:
http_access deny multimedia

И перезавантажимо Squid.

Можемо перевіряти, що всі файли із зазначеними нами розширеннями не завантажуються, а користувачеві в браузері виводиться повідомлення про заборону доступу.

Авторизація на проксі-сервері Squid за логіном та паролем

Для того, щоб дозволити авторизацію користувачів по створеному нами логіну та паролю, у файл конфігурації впишемо наступні рядки:

auth_param basic program /usr/lib/squid/basic_ncsa_auth /etc/squid/passwd
auth_param basic children 5
auth_param basic realm Squid proxy-caching web server
auth_param basic credentialsttl 2 hours

В яких описується:

1. яка саме програма (плагін івторизації) і з якого файлу буде брати логіни і паролі користувачів;
2. скільки примірників програми авторизації може працювати одночасно;
3. текст, який буде показаний користувачеві у вікні авторизації;
4. час, на який кешує пароль на сервері.

Далі в описах acl додаємо рядок:
acl password proxy_auth REQUIRED
який і вказує проксі-серверу запитувати авторизацію.

Для того, щоб дозволити вихід авторизованим користувачам закоментуємо раніше написаний нами рядок *http_access allow mynetwork* і вписуємо новий:
http_access allow password
Природно, цей рядок мусить бути безпосередньо перед *http_access deny all*

Тепер необхідно створити файл з логінами і паролями користувачів. Для цього потрібно поставити пакет *apache2-utils*. Введемо команду:
$sudo apt install apache2-utils

Після закінчення встановлення почнемо створювати пари логінів-паролів.
Перший раз це робиться командою:
$htpasswd -c /etc/squid/passwd ivan

Ключ -c говорить про те, що файл з обліковими записами потрібно створити заново.
Увага! Якщо Ви з ключем -c створите другу або ще більш пізню пару логін/пароль, то всі попередні дані будуть втрачені.
Наступні облікові записи створюються командою:
htpasswd /etc/squid/passwd petro

Після перезапуску сервісу проксі-сервера можна перевіряти, як працює наш Squid з авторизацією.
Видалити облікові записи користувача з файлу можна або самому, знайшовши у файлі з паролями відповідний рядок, або за допомогою команди з ключем -D. Наприклад для видалення створеного нами користувача ivanov, слід ввести команду:
$htpasswd -D /etc/squid/passwd petro

Обмеження ширини каналу для різних груп користувачів.

Для обмеження ширини інтернет-каналу що використовується користувачами, в проксі-сервер Squid використовуються delay_pool'и. Вони бувають трьох класів:

1. Індивідуальні — задається ширина каналу для користувача в байтах за секунду delay_parameters 1 32000/32000
2. Delay pools другого класу. Призначені для невеликих мереж. В параметрах тепер задається шейп на всю мережу перша пара і шейп на робочу станцію, це друга пара цифр delay_parameters 1 8000/8000 4000/4000
3. Delay pools третього класу призначені для великих мереж, що включають підмережі. Тепер параметри пулу описуються трьома парами цифр перша пара описує загальний, для всіх підмереж шейп друга пара — шейп виділений для підмережі третя пара - для хоста delay_parameters 1 32000/32000 16000/16000 4000/4000

Створимо обмеження по швидкості для одного користувача, тобто створимо delay pool 1-го класу.

У блоці acl створимо новий запис:

*acl baduser src 10.10.10.1*00

де 10.10.10.100 адресу співробітника офісу, якому потрібно урізати канал.

Закоментуємо наш рядок авторизації http_access allow password

Далі, розкоментуємо рядок http_access allow mynetwork і вставимо перед ним:

delay_pools 1
delay_class 1 1
delay_access 1 allow baduser
delay_access 1 deny all
delay_parameters 1 2000/2000
http_access allow baduser

Після перезавантаження сервісу Squid можна перевіряти, що швидкість закачування всіх файлів для нашого користувача з адресою 10.10.10.100 впала до, приблизно, 2-х кілобайт за секунду.

Налаштування контентної фільтрації за допомогою DansGuardian

Встановимо DansGuardian.
#apt install dansguardian

Зверніть увагу, що разом з DansGuardian встановлюється і антивірус ClamAV.
Тепер займемося налаштуванням цих програм.
За замовчуванням SQUID налаштований коректно і ми його зараз чіпати не будемо. Займемося виключно DansGuardian.
Відкриємо на редагування файл:

/etc/dansguardian/dansguardian.conf

Для того, щоб потім можна було дивитися логи ходіння в інтернет у файлі конфігурації змінної logfileformat призначимо значення 3.
Далі раскоментуємо змінну loglocation.

Всі налаштування в розділі Network Settings залишаємо без змін, так як вони прописані саме для нашого випадку - коли Squid встановлений на тому ж сервері і його налаштування (зокрема порт, що прослуховується) залишені незмінними.

Налаштування антивірусної перевірки

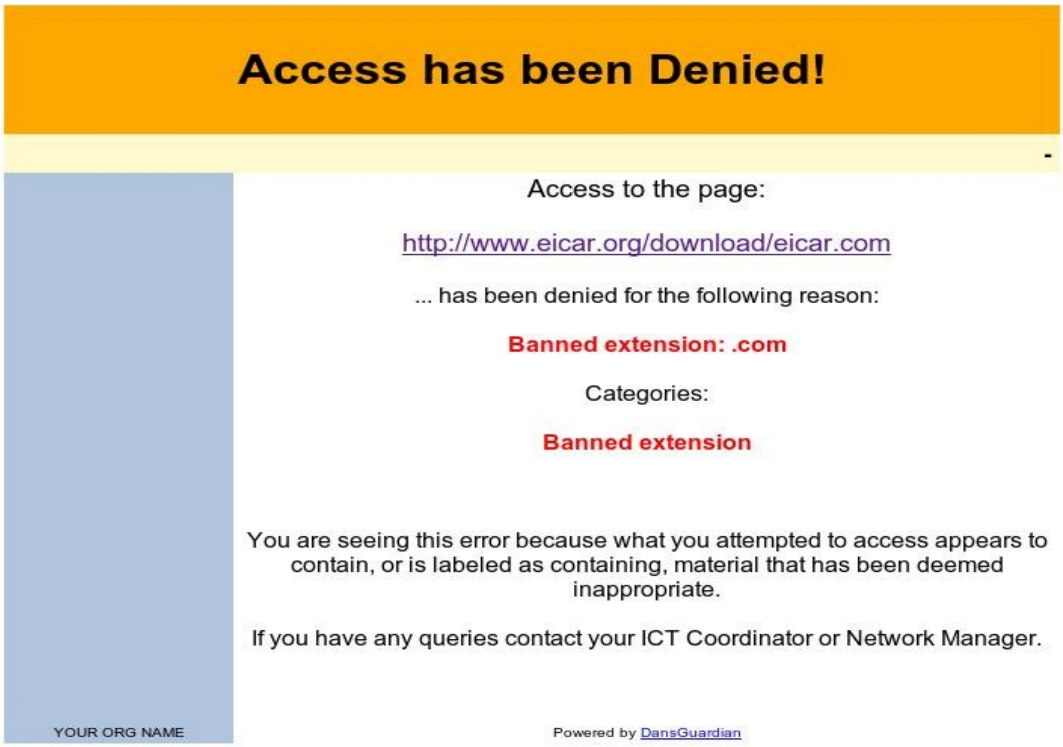
Для того, щоб увімкнути антивірусну перевірку, розкоментуємо рядок:
contentscanner = '/etc/dansguardian/contentscanners/clamav.conf'

Збережемо файл конфігурації.
Зараз дозволимо завантаження zip-архівів.
Для цього закоментуємо в файлі
/etc/dansguardian/lists/bannedextensionlist
рядок:
.zip # Windows compressed file

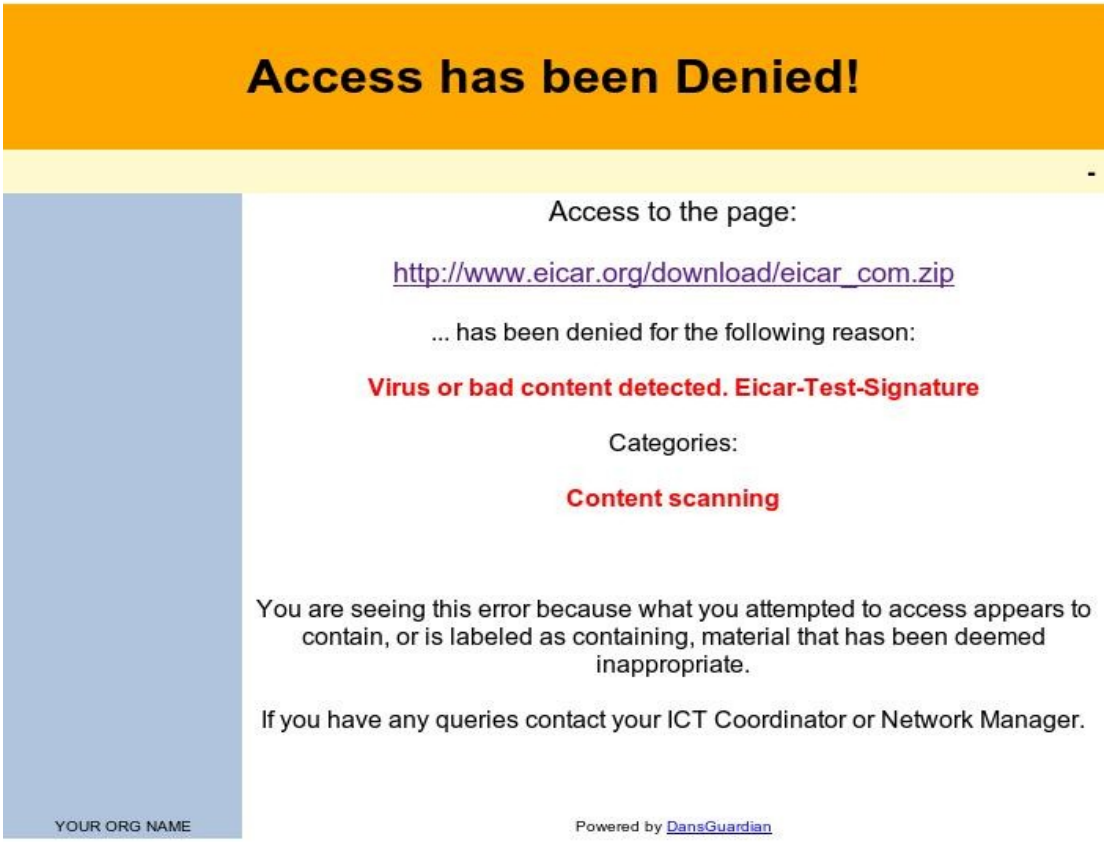
та в файлі
/etc/dansguardian/lists/bannedmimetypelist
рядок:
application/zip

Для того, щоб налаштування подіяли, треба перезапустити DansGuardian.
#service dansguardian stop
#service dansguardian start

Тепер подивимось, як працює перевірка на віруси.
В налаштуваннях браузера пропишемо в якості проксі-сервера IP-адресу нашого сервера та порт 8080.
Спробуємо зайти на http://eicar.org та завантажити звідти eicar.com
Ми відразу ж побачили, що фільтрація спрацювала на розширення файлу. Відповідно антивірус цей файл перевіряти не буде.



Тепер спробуємо завантажити eicar_com.zip. Це розширення ми дозволили, відповідно блокуватися він не буде.



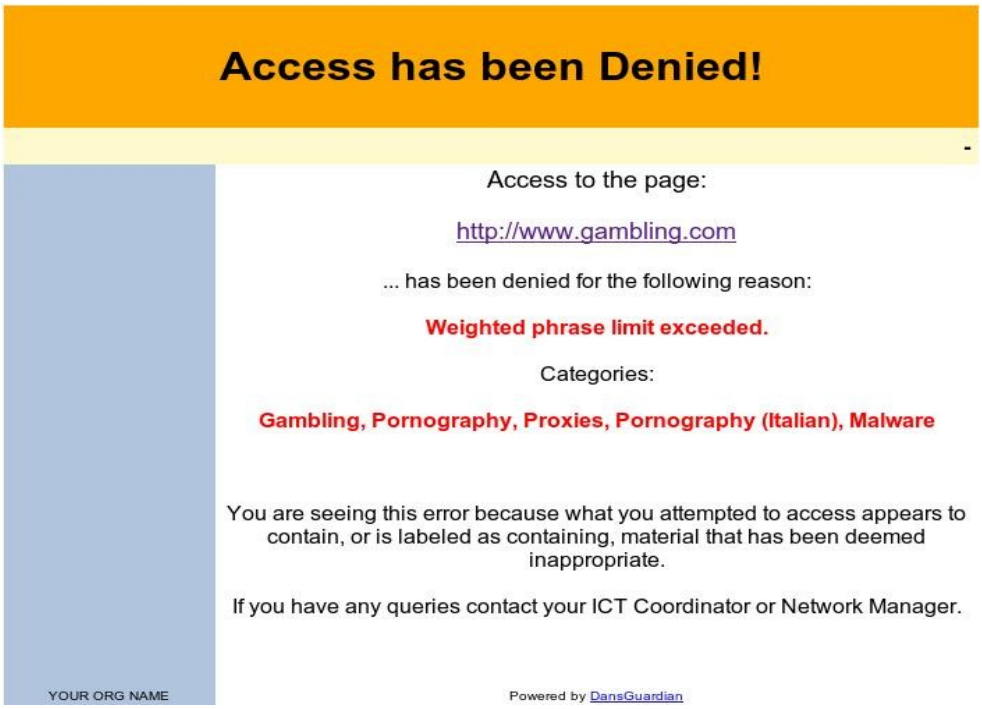
Тут ми вже бачимо, що спрацювала антивірусна перевірка та завантаження файлу заблокована з причини його зараженості.

Налаштування контентної фільтрації

Тепер перевіримо, як працює контентна фільтрація трафіку.
Відкриємо на редагування файл:
/etc/dansguardian/lists/weightedphraselist
і розкоментуємо в ньому рядок:
.Include</etc/dansguardian/lists/phraselists/gambling/weighted>

Знову перезавантажимо DansGuardian і спробуємо з браузера відкрити сайт http://www.gambling.com/

В вікні, що з'явилося, видно не тільки те, що сторінка заблокована, а й з якої саме причини це сталося.



Налаштування роботи з групами користувачів

Спробуємо розібратися з групами користувачів.

Для початку включимо визначення IP-адрес користувачів. Для цього відкриємо на редагування файл:
/etc/dansguardian/dansguardian.conf

в ньому розкоментуємо рядок:
authplugin = '/etc/dansguardian/authplugins/ip.conf'
змінимо значення змінної filtergroups
filtergroups = 2
Тепер скопіюємо файл /etc/dansguardian/dansguardianf1.conf в /etc/dansguardian/dansguardianf2.conf

#cp /etc/dansguardian/dansguardianf1.conf /etc/dansguardian/dansguardianf2.conf

В файлі */etc/dansguardian/dansguardianf2.conf* змінимо
weightedphraselist = '/etc/dansguardian/lists/weightedphraselist'
на
weightedphraselist = '/etc/dansguardian/lists/weightedphraselist.boss'

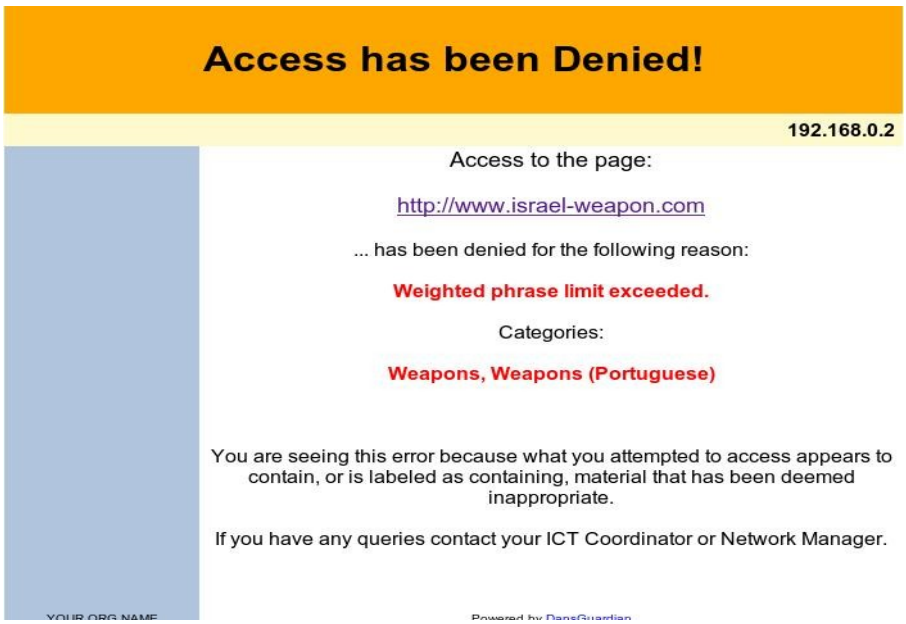
Тепер введемо команду
#cp /etc/dansguardian/lists/weightedphraselist /etc/dansguardian/lists/weightedphraselist.boss

Простежимо, що б у файлі /etc/dansguardian/lists/weightedphraselist рядки
.Include</etc/dansguardian/lists/phraselists/weapons/weighted>
.Include</etc/dansguardian/lists/phraselists/weapons/weighted_portuguese>

були розкоментовані, а в */etc/dansguardian/lists/weightedphraselist.boss* відповідно закоментовані.

Тепер починаємо правити файл /etc/dansguardian/lists/authplugins/ipgroups
У ньому потрібно буде прописати рядок:
<IP-адреса клієнтського комп'ютера>=filter2
та
192.168.0.0/255.255.255.0 = filter1

Перезавантажуємо DansGuardian
Перевіряємо як працюють наші фільтри.
Відкриваємо в браузері посилання http://www.weapons-universe.com/
Сторінка відкрилася нормально.
У файлі /etc/dansguardian/lists/authplugins/ipgroups закоментуємо рядок:
<IP-адреса клієнтського комп'ютера>=filter2 та перезавантажуємо DansGuardian
У браузері чистимо кеш і знову пробуємо відкрити http://www.israel-weapon.com/
Бачимо:



Блокування спрацювало.

Робота з пакетним фільтром в Linux

Iptables — утиліта командного рядка, вона є стандартним інтерфейсом управління роботою міжмережевого екрану для ядер Linux, починаючи з версії 2.4. Для використання утиліти iptables потрібні повноваження суперкористувача.

Попередниками iptables були проекти ipchains (застосовувалася для адміністрування фаєрвола ядра Linux версії 2.2) і ipfwadm (для ядер Linux версій 2.0).

Iptables зберігає ідеологію, що бере початок від ipfwadm: функціонування фаєрволу визначається набором правил, кожне з яких складається з критерію та дії, що застосовується до пакетів, які підпадають під цей критерій. В ipchains з'явилася концепція ланцюжків — незалежних списків правил. Були введені окремі ланцюжки для фільтрації вхідних (INPUT), вихідних (OUTPUT) і транзитних (FORWARD) пакетів. У продовженні цієї ідеї, в iptables з'явилися таблиці — незалежні групи ланцюжків. Кожна таблиця вирішувала свою задачу — ланцюжки таблиці filter відповідали за фільтрацію, ланцюжки таблиці nat — за перетворення мережевих адрес, до завдань таблиці mangle належали інші модифікації заголовків пакетів (наприклад, зміна TTL або TOS). Крім того, була злегка змінена логіка роботи ланцюжків: в ipchains всі вхідні пакети, включаючи транзитні, проходили ланцюжок INPUT. У iptables через INPUT проходять тільки пакети, адресовані самому хосту.

Такий поділ функціональності дозволив iptables при обробці окремих пакетів використовувати інформацію про з'єднання в цілому. У цьому iptables значно перевершує ipchains.

iptables може відстежувати стан з'єднання і перенаправляти, змінювати або фільтрувати пакети, ґрунтуючись не лише на даних з їх заголовків (джерело, одержувач) або вмісті пакетів, але і на підставі даних про з'єднання. Можна сказати, що iptables аналізує не тільки дані, що передаються, а й контекст їх передачі і тому може приймати обґрунтовані рішення про долю кожного конкретного пакета.

Архітектура

Ключовими поняттями iptables є:

- **Правило** — складається з критерію, дії і лічильника. Якщо пакет відповідає критерію, до нього застосовується дія, і він враховується лічильником. Критерію може і не бути — тоді неявно передбачається критерій «всі пакети». Вказувати дію теж не обов'язково - за відсутності дії правило буде працювати тільки як лічильник.
    - **Критерій** — логічний вираз, що аналізує властивості пакета або з'єднання і визначає, чи підпадає даний конкретний пакет під дію поточного правила.
    - **Дія** — опис дії, яку потрібно виконати з пакетом або з'єднанням в тому випадку, якщо вони підпадають під дію цього правила.
    - **Лічильник** — компонент правила, що забезпечує облік кількості пакетів, які потрапили під критерій даного правила. Також лічильник враховує сумарний обсяг таких пакетів в байтах.
- **Ланцюжок** — упорядкована послідовність правил. Ланцюжки можна розділити на *користувацькі* і *базові*.
    - **Базовий ланцюжок** — ланцюжок, що створюється за замовчуванням при ініціалізації таблиці. Кожен пакет, в залежності від того, призначений він самому хосту, згенерований ним або є транзитними, повинен пройти покладений йому набір базових ланцюжків різних таблиць. Крім того, базовий ланцюжок відрізняється від користувацького наявністю «дії за умовчанням» (default policy). Ця дія застосовується до тих пакетів, які не були оброблені

іншими правилами цього ланцюжка і викликаних з неї ланцюжків. Імена базових ланцюжків завжди записуються у верхньому регістрі (PREROUTING, INPUT, FORWARD, OUTPUT, POSTROUTING).

- **Користувацький ланцюжок** — ланцюжок, створений користувачем. Може використовуватися тільки в межах своєї таблиці. Рекомендується не використовувати для таких ланцюжків імена у верхньому регістрі, щоб уникнути плутанини з базовими ланцюжками і вбудованими діями.

- **Таблиця** — сукупність базових і користувацьких ланцюжків, об'єднаних загальним функціональним призначенням. Імена таблиць (як і модулів критеріїв) записуються в нижньому регістрі, так як в принципі не можуть конфліктувати з іменами користувацьких ланцюжків. При виклику команди iptables таблиця вказується в форматі -t ім'я_таблиці. При відсутності явної вказівки, використовується таблиця filter.

В IPTables використовується три види таблиць:

1. Mangle — зазвичай цей ланцюжок використовується для внесення змін в заголовок пакета, наприклад для зміни бітів TOS і пр.
2. Nat — цей ланцюжок використовується для трансляції мережевих адрес (Destination Network Address Translation). Будь-якого роду фільтрація в цьому ланцюжку може проводитися тільки у виняткових випадках.
3. Filter — тут відбувається фільтрація трафіку. Слід пам'ятати, що всі вхідні пакети, адресовані нам, проходять через цей ланцюжок, незалежно від того з якого інтерфейсу вони надійшли.

Коли пакет приходить на наш брандмауер, то він спершу потрапляє на мережевий пристрій, перехоплюється відповідним драйвером і далі передається в ядро. Далі пакет проходить ряд таблиць і потім передається або локальному додатку, або переправляється на іншу машину. Порядок проходження пакета наводиться нижче.

Таблиця 1. Порядок руху транзитних пакетів

Крок	**Таблиця**	**Ланцюжок**	**Примітка**
1	—	—	Кабель (тобто Інтернет)
2	—	—	Мережевий інтерфейс (наприклад, eth0)
3	Mangle	PREROUTING	Зазвичай цей ланцюжок використовується для внесення змін в заголовок пакета, наприклад для зміни бітів TOS та ін.
4	Nat	PREROUTING	Цей ланцюжок використовується для трансляції мережевих адрес (Destination Network Address Translation). Source Network Address Translation виконується пізніше, в іншому ланцюжку. Будь-якого роду фільтрація в цьому ланцюжку може проводитися тільки у виняткових випадках.
5	—	—	Прийняття рішення про подальшу маршрутизацію, тобто в цій точці вирішується куди піде пакет - локальному додатку чи на інший вузол мережі.

6	Filter	FORWARD	В ланцюжок FORWARD потрапляють тільки ті пакети, які йдуть на інший хост. Вся фільтрація транзитного трафіку повинна виконуватися тут. Не забувайте, що через цей ланцюжок проходить трафік в обох напрямках, обов'язково враховуйте цю обставину при написанні правил фільтрації.
7	Mangle	FORWARD	Далі пакет потрапляє в ланцюжок FORWARD таблиці mangle, який мусить використовуватися тільки у виняткових випадках, коли необхідно внести деякі зміни в заголовок пакета між двома точками прийняття рішення про маршрутизації.
8	—	—	Прийняття рішення про подальшу маршрутизацію, тобто в цій точці, наприклад, вирішується на який інтерфейс піде пакет.
9	Nat	POSTROUTING	Цей ланцюжок призначена в першу чергу для Source Network Address Translation. Не використовуйте його для фільтрації без особливої на те необхідності. Тут же виконується і маскування (Masquerading).
10	Mangle	POSTROUTING	Цей ланцюжок призначений для внесення змін в заголовок пакета вже після того як прийнято остаточне рішення про маршрутизацію.
11	—	—	Вихідний мережевий інтерфейс (наприклад, eth1).
12	—	—	Кабель (нехай буде LAN).

Як ви можете бачити, пакет проходить кілька етапів, перш ніж він буде переданий далі. На кожному з них пакет може бути зупинений, будь то ланцюжок iptables або що небудь ще, але нас головним чином цікавить iptables. Зауважте, що немає жодних ланцюжків, специфічних для окремих інтерфейсів або чогось подібного. Ланцюжок FORWARD проходять ВСІ пакети, які рухаються через наш брандмауер/роутер. Не використовуйте ланцюжок INPUT для фільтрації транзитних пакетів, вони туди просто не потрапляють. Через цей ланцюжок рухаються тільки ті пакети, які призначені даного хосту.

А тепер розглянемо порядок руху пакета, призначеного локальному процесу/додатку.

Таблиця 2. Для локального додатку

Крок	**Таблиця**	**Ланцюжок**	**Примітка**
1	—	—	Кабель (тобто Інтернет)
2	—	—	Вхідний мережевий інтерфейс (наприклад, eth0)
3	Mangle	PREROUTING	Зазвичай використовується для внесення змін в заголовок пакета, наприклад для установки бітів TOS і пр.
4	Nat	PREROUTING	Перетворення адрес (Destination Network Address Translation). Фільтрація пакетів тут допускається лише у виняткових випадках.

5	—	—	Прийняття рішення про маршрутизацію.
6	Mangle	INPUT	Пакет потрапляє в ланцюжок INPUT таблиці mangle. Тут вносяться зміни в заголовок пакета перед тим як він буде переданий локальному додатку.
7	Filter	INPUT	Тут відбувається фільтрація вхідного трафіку. Пам'ятайте, що всі вхідні пакети, адресовані нам, проходять через цей ланцюжок, незалежно від того з якого інтерфейсу вони надійшли.
8	—	—	Локальний процес / додаток

Важливо пам'ятати, що на цей раз пакети йдуть через ланцюжок INPUT, а не через FORWARD. І на закінчення ми розглянемо порядок руху пакетів, створених локальними процесами.

Таблиця 3. Від локальних процесів

Крок	**Таблиця**	**Ланцюжок**	**Примітка**
1	—	—	Локальний процес
2	Mangle	OUTPUT	Тут відбувається внесення змін в заголовок пакета. Фільтрація, що виконується в цьому ланцюжку, може мати негативні наслідки.
3	Nat	OUTPUT	Цей ланцюжок використовується для трансляції мережевих адрес (NAT) в пакетах, що виходять від локальних процесів брандмауера.
4	Filter	OUTPUT	Тут фільтрується вихідний трафік.
5	—	—	Прийняття рішення про маршрутизацію. Тут вирішується куди піде пакет далі.
6	Nat	POSTROUTING	Тут виконується Source Network Address Translation (SNAT). Не слід в цьому ланцюжку виконувати фільтрацію пакетів, щоб уникнути небажаних побічних ефектів. Однак і тут можна зупиняти пакети, застосовуючи політику по-замовчуванню DROP.
7	Mangle	POSTROUTING	Ланцюжок POSTROUTING таблиці mangle в основному використовується для правил, які повинні вносити зміни в заголовок пакета перед тим, як він покине брандмауер, але вже після прийняття рішення про маршрутизацію. У цей ланцюжок потрапляють всі пакети, як транзитні, так і створені локальними процесами брандмауера.
8	—	—	Мережевий інтерфейс (наприклад, eth0)
9	—	—	Кабель (тобто Інтернет)

Правила IPTables

Розглянемо як записуються правила IPTables.

Кожне правило — це рядок, що містить в собі критерії, що визначають, чи підпадає пакет під задане правило, і дія, яку необхідно виконати в разі виконання критерію. В загальному вигляді правила записуються приблизно так:

iptables [-t table] command [match] [target]

Ніде не стверджується, що опис дії (target) повинен стояти останнім у рядку, проте, така нотація більш зручна для читання. І найчастіше зустрічається саме такий спосіб запису правил.

Якщо в правило не включається специфікатор -t table, то за замовчуванням передбачається використання таблиці filter, якщо ж передбачається використання іншої таблиці, то це потрібно вказати явно. Специфікатор таблиці так само можна вказувати в будь-якому місці рядка правила, однак більш-менш стандартом вважається зазначення таблиці на початку правила.

Далі, безпосередньо за іменем таблиці, має стояти команда. Якщо специфікатора таблиці немає, то команда завжди повинна стояти першою. Команда визначає дію iptables, наприклад: вставити правило, чи додати правило в кінець ланцюжка, чи видалити правило тощо.

Розділ match задає критерії перевірки, за якими визначається підпадає пакет під дію цього правила чи ні. Тут ми можемо вказати найрізноманітніші критерії — IP-адреса джерела пакета чи мережі, IP-адреса місця призначення, порт, протокол, мережевий інтерфейс і т.д. Існує чимало різноманітних критеріїв.

І нарешті target вказує, яке дія повинна бути виконана за умови виконання критеріїв у правилі. Тут можна змусити ядро передати пакет в інший ланцюжок правил, "скинути" пакет і забути про нього, видати на джерело повідомлення про помилку і т.п.

Для перегляду вже існуючих правил введемо команду:
#iptables -L

Ми не вказували, до якої таблиці застосовується дана команда, значить за замовчуванням це буде таблиця Filter.

Так як у нас поки не створено жодних правил, то вивід буде приблизно таким:

Chain INPUT (policy ACCEPT)

target prot opt source destination

Chain FORWARD (policy ACCEPT)

target prot opt source destination

Chain OUTPUT (policy ACCEPT)

target prot opt source destination

Блокування доступу до сервера по портах

Введемо команду, яка буде блокувати доступ до 80 порту нашого сервера по протоколу TCP:

#iptables -A INPUT -p tcp -s 0/0 --dport 80 -j DROP

Якщо тепер подивитися на виведення наших правил, то він вже буде виглядати так:

Chain INPUT (policy ACCEPT)

target prot opt source destination

DROP tcp -- anywhere anywhere tcp dpt:www

Chain FORWARD (policy ACCEPT)

target prot opt source destination

Chain OUTPUT (policy ACCEPT)

target prot opt source destination

І тепер зайти на сервер на 80 порт уже не вийде.

Подивимося як видаляються правила.

Якщо потрібно повністю очистити всю таблицю можна використовувати команду:

#iptables [-t table] -F

Або, якщо потрібно видалити тільки одне правило, то в нашому випадку, можна скористатися командою:

#iptables -D INPUT -p tcp -s 0/0 --dport 80 -j DROP

Тобто та ж сама команда, але з ключем -D замість -A

Якщо ж правил у нас багато, то команда

 # iptables -L --line-numbers

виведе їх вже пронумерованими

t*arget prot opt source destination*

 # iptables -L --line-numbers

Chain INPUT (policy ACCEPT)

num target prot opt source destination

1 DROP tcp -- anywhere anywhere tcp dpt:www

2 DROP tcp -- anywhere anywhere tcp dpt:http-alt

Chain FORWARD (policy ACCEPT)

num target prot opt source destination

Chain OUTPUT (policy ACCEPT)

num target prot opt source destination

І тоді їх можна видаляти за номером:
#iptables -D INPUT 2

Буде видалене правило 2 з таблиці Filter і ланцюжка INPUT

Прокидування портів

Нехай у нашого навчального сервера є два інтерфейси: eth0 — зовнішній з адресою 192.168.0.10 і eth1 — внутрішній з адресою 10.10.10.1.

Наша задача — прокинути порт з зовнішнього інтерфейсу на комп'ютер у внутрішній мережі.

Внутрішня мережа у нас має адресацію 10.10.10.0/24 і в ній є веб-сервер з адресою 10.10.10.10, який слухає з'єднання на порту 8080.

Дозволимо кидок порту на 8080. Для цього створимо відповідне правило в таблиці Filter і ланцюжку FORWARD:

 # iptables -A FORWARD -i eth0 -p tcp --dport 8080 -j ACCEPT

Тепер опишемо саме правило проброса порту:

 # iptables -t nat -A PREROUTING -p tcp -d 192.168.0.10 --dport 8080 -j DNAT --to-destination 10.10.10.10:8080

І, нарешті, дозволимо форвардинг пакетів в ядрі:

echo 1 > /proc/sys/net/ipv4/ip_forward

Якщо ж правило прокидування порту має діяти і у внутрішній мережі, необхідно додати ще одне правило:

 # iptables -A POSTROUTING -t nat -p tcp --dst 10.10.10.10 --dport 8080 -j SNAT --to-source 10.10.10.1

Тепер порт прокинутий і при з'єднанні з зовнішньої мережі на наш сервер на порт 8080 з'єднання піде на внутрішній комп'ютер на той же самий порт.

Однак лишається ще один випадок — коли таке з'єднання ініціює сам сервер. Якщо спробувати створити таке з'єднання (наприклад виконати telnet 192.168.0.10 8080) ми побачимо, що зв'язок відсутній. Для того, що б прокинути порт і для цього випадку ¸ треба додати ще й таке правило:

 # iptables -t nat -A OUTPUT -d 192.168.0.10 -p tcp --dport 8080 -j DNAT —to-destination 10.10.10.10:8080

Побудова NAT

NAT будується за тим же принципом, що і прокидування портів.

По-перше необхідно дозволити форвард пакетів з внутрішньої мережі назовні. Це робиться командою:

 # iptables -A FORWARD -i eth1 -o eth0 -j ACCEPT

Далі будуємо, власне, сам NAT Для цього створимо відповідне правило:

 # iptables -t nat -A POSTROUTING -o eth0 -s 10.10.10.0/24 -j MASQUERADE

Залишилося створити останнє правило — потрібно що б пакети, які відносяться до вже встановлених з'єднань поверталися на комп'ютери у внутрішній мережі. Це робиться такою командою:

 # iptables -A FORWARD -i eth0 -m state --state ESTABLISHED,RELATED -j ACCEPT

Крім цього обов'язково дозволити форвардинг пакетів в ядрі. Ми це зробили при прокидуванні портів. Але якщо в файл /proc/sys/net/ipv4/ip_forward записаний «0», то це значення необхідно змінити на «1», інакше NAT просто не запрацює.

Насправді можливості IPTables значно ширші і для їх використання я б радив почитати документацію. Або man iptables, або http://iptables.ru/

Створення власної конфігурації IPTables

Після того, як ми налаштували IPTables, звичайно хочеться всі налаштування зберегти, що б після перезавантаження комп'ютера не довелося їх робити знову вручну. Для цього є декілька шляхів.

Стандартним шляхом є використання iptables-save і iptables-restore.

Наприклад, після того, як ми відконфігурували iptables, виконаємо:

#iptables-save > /etc/iptables.rules

Тепер пропишемо автоматичне відновлення всіх правил. Для цього в файл */etc/network/interfaces* в блок опису одного з інтерфейсів допишемо команду відновлення правил. Тепер цей блок має виглядати приблизно так:

```
auto eth0
iface eth0 inet static
pre-up iptables-restore < /etc/iptables.rules
address 192.168.0.4
network 192.168.0.0
netmask 255.255.255.0
gateway 192.168.0.1
```

Тепер при перезавантаженні сервера всі правила iptables будуть відновлені автоматично.

На сьогоднішній день це вже застарілий спосіб, бо в сучасних дистрибутивах файл */etc/network/interfaces* не підтримується. Зараз використовується networkd-dispatcher. Для відновлення правил IPTables треба створити файл /etc/networkd-dispatcher/routable.d/50-ifup-hooks і записати в нього:

```
#!/bin/sh
   iptables-restore <  /etc/iptables.conf
exit 0
```

Потім виставити на цей файл ознаку виконання */etc/networkd-dispatcher/routable.d/50-ifup-hooks*

 # chmod +x

І тепер при старті системи всі правила IPTables будуть відновлені.

Розглянемо і інший варіант налаштувань.

Створимо скрипт */etc/init.d/firewall* і заповнимо його наступними даними:

```
#! /bin/sh
set -e
INET_IFACE="eth0"
LAN_IFACE="eth1"
LO_IFACE="lo"
```

```
LAN_IP="192.168.0.1"
LO_IP="127.0.0.1"
LAN_IP_RANGE="192.168.0.0/24"
IPTABLES="/sbin/iptables"
case "$1" in
  start)
        echo -n "Starting firewall... "
   #Завантажуємо необхідні модулі
        /sbin/depmod -a
        /sbin/modprobe ip_conntrack
        /sbin/modprobe ip_tables
        /sbin/modprobe iptable_filter
        /sbin/modprobe iptable_mangle
        /sbin/modprobe iptable_nat
        /sbin/modprobe ipt_LOG
        /sbin/modprobe ipt_limit
        /sbin/modprobe ipt_MASQUERADE
        /sbin/modprobe ip_conntrack_ftp
        /sbin/modprobe ip_nat_ftp
        /sbin/modprobe ipt_REJECT

# В таблиці FILTER у всіх ланцюжках міняємо політику за замовчуванням
        $IPTABLES -P INPUT DROP
        $IPTABLES -P OUTPUT ACCEPT
        $IPTABLES -P FORWARD DROP
# Створюємо нові ланцюжки для обробки трафіку по протоколах
        $IPTABLES -N bad_tcp_packets
        $IPTABLES -N tcp_packets
        $IPTABLES -N udp_packets
        $IPTABLES -N icmp_packets

#Прописуємо в ланцюжку bad_tcp_packets правило яке буде перешкоджати спуфінгу
# від нашого імені. Адже якщо ми отримуємо пакет з встановленими прапорцями SYN
# і ACK (така комбінація прапорців є тільки у відповіді на SYN-пакет) по ще не
# відкритому з'єднанню, це означає, що хтось послав іншому хосту SYN-пакет від
# нашого імені, і відповідь прийшла до нас. Згідно наведеного правила, наш хост дасть
# відповідь RST-пакетом, після отримання якого хост, що атакується закриє
# з'єднання.
        $IPTABLES -A bad_tcp_packets -p tcp --tcp-flags SYN,ACK SYN,ACK -m state --state
NEW -j REJECT --reject-with tcp-reset
        $IPTABLES -A bad_tcp_packets -p tcp ! --syn -m state --state NEW -j DROP

        #Блокуємо TCP-з'єднання без прапорця  SYN
        $IPTABLES -A tcp_packets -p TCP ! --syn -j DROP

        #Відкриваємо порт для ssh-сервера
        $IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 22 -j ACCEPT

#Відкриваємо порти для веб-сервера
      $IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 80 -j ACCEPT
```

```
    $IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 443 -j ACCEPT

  #Дозволяємо пінги
    $IPTABLES -A udp_packets -p UDP -s 0/0 --sport 67 --dport 68 -j ACCEPT
    $IPTABLES -A icmp_packets -p ICMP -s 0/0 --icmp-type 0 -j ACCEPT
    $IPTABLES -A icmp_packets -p ICMP -s 0/0 --icmp-type 8 -j ACCEPT
    $IPTABLES -A icmp_packets -p ICMP -s 0/0 --icmp-type 11 -j ACCEPT

  # Перенаправляємо весь трафік в ланцюжок bad_tcp_packets з ланцюжка INPUT
    $IPTABLES -A INPUT -p tcp -j bad_tcp_packets

  # Дозволяємо весь трафік з внутрішнього і loopback інтерфейсів
    $IPTABLES -A INPUT -p ALL -i $LAN_IFACE -s $LAN_IP_RANGE -j ACCEPT
    $IPTABLES -A INPUT -p ALL -i $LO_IFACE -j ACCEPT
  # Дозволяємо трафік який відноситься до вже встановлених з'єднань
    $IPTABLES -A INPUT -p ALL -m state --state ESTABLISHED,RELATED -j ACCEPT


  # Перенаправляємо весь трафік в ланцюжки, відповідні до його протоколу
    $IPTABLES -A INPUT -p TCP -j tcp_packets
    $IPTABLES -A INPUT -p UDP -j udp_packets
    $IPTABLES -A INPUT -p ICMP -j icmp_packets

  # Перенаправляємо весь трафік в ланцюжок bad_tcp_packets з ланцюжка FORWARD
    $IPTABLES -A FORWARD -p tcp -j bad_tcp_packets

  # Дозволяємо форвардинг всіх пакетів з внутрішньої мережі
    $IPTABLES -A FORWARD -i $LAN_IFACE -j ACCEPT
  # Дозволяємо форвардинг пакетів зі світу, якщо вони відносяться до встановлених
  # з'єднань.
    $IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

  # Перенаправляємо весь трафік в ланцюжок bad_tcp_packets з ланцюжка OUTPUT
    $IPTABLES -A OUTPUT -p tcp -j bad_tcp_packets

  # Будуємо NAT для внутрішньої мережі
    $IPTABLES -A POSTROUTING -t nat -o $INET_IFACE -j MASQUERADE
  # Дозволяємо форвардинг пакетів в ядрі
    echo "1" > /proc/sys/net/ipv4/ip_forward
    echo "done."
    ;;
  stop)
    echo "Stopping firewall..."
 # У всіх ланцюжках правила за замовчуванням встановлюємо в ACCEPT
    $IPTABLES -P INPUT ACCEPT
    $IPTABLES -P FORWARD ACCEPT
    $IPTABLES -P OUTPUT ACCEPT
    $IPTABLES -t nat -P PREROUTING ACCEPT
    $IPTABLES -t nat -P POSTROUTING ACCEPT
    $IPTABLES -t nat -P OUTPUT ACCEPT
```

```
        $IPTABLES -t mangle -P PREROUTING ACCEPT
        $IPTABLES -t mangle -P POSTROUTING ACCEPT
        $IPTABLES -t mangle -P INPUT ACCEPT
        $IPTABLES -t mangle -P OUTPUT ACCEPT
        $IPTABLES -t mangle -P FORWARD ACCEPT
    #Видаляємо всі правила
        $IPTABLES -F
        $IPTABLES -t nat -F
        $IPTABLES -t mangle -F
    # Видаляємо всі нами створені ланцюжки
        $IPTABLES -X
        $IPTABLES -t nat -X
        $IPTABLES -t mangle -X
        ;;
     restart|force-reload)
        /etc/init.d/firewall stop
        /etc/init.d/firewall start
        ;;
     *)
        echo "Usage: $SCRIPTNAME {start|stop|restart|force-reload}" >&2
        exit 1
        ;;
    esac
    exit 0
```

Після чого встановлюємо на цей скрипт права на виконання:
#chmod +x /etc/init.d/firewall

И пропишемо його на запуск на рівнях 2-5
#sysv-rc-conf --level 2345 firewall on

Для того, щоб в скрипті кожен раз не використовувати конструкцію
echo "1" > /proc/sys/net/ipv4/ip_forward

Форвард пакетів в ядрі можна дозволити один раз і назавжди. Для цього потрібно в файлі */etc/sysctl.conf* виставити значення параметра *net.ipv4.ip_forward* в 1.

Для того, щоб зміни вступили в силу без перезавантаження сервера, необхідно виконати команду:

#sysctl -p /etc/sysctl.conf

Fail2ban — захист від перебору паролів

Fail2ban — це інструмент, який відстежує в log-файлах спроби звернення до мережемих сервісів та якщо знаходить невдалі спроби, що повторюються,з одного й того ж хосту, то блокує подальші спроби за допомогою правил iptables чи host.deny.

Встановимо fail2ban:

 # apt install fail2ban

Для своєї роботы fail2ban створює власні ланцюжки в таблиці filter, до яких переспрямовуються на аналіз пакети з ланцюжка INPUT.

Наприклад, у випадку підключеного захисту лише ssh, правила IPTables будуть выглядати так:

```
# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
fail2ban-ssh  tcp  --  anywhere             anywhere            multiport dports ssh

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain fail2ban-ssh (1 references)
target     prot opt source               destination
RETURN     all  --  anywhere             anywhere
```

Налаштування fail2ban

Основні налаштування проводяться в файлі */etc/fail2ban/jail.conf*
Сам файл розділений на декілька секцій.
В секції Default описані загальні налаштування, які стосуються всіх jail-секцій.
Основні параметри в цій секції:
ignoreip — білий список адрес. Може включати ip-адреси, підмережі та доменні імена
bantime — час бану в секундах
maxretry — кількість спроб перед попаданням до чорного списку
chain — в який ланцюжок таблиці filter будуть вноситись правила заборони

Далі йдут так звані «тюремні» (jail) секції. Кожна з них має свою назву та свій набір параметрів. Поглянемо, наприклад, на секцію ssh:

```
[ssh]

enabled  = true
port     = ssh
filter   = sshd
logpath  = /var/log/auth.log
maxretry = 6
```

Тут вказані:
enabled — ознака вмикання фільтру
port — який порт буде блокуватися
filter — ім’я файлу, в якому описані регулярні вирази для пошуку некоректних з’єднань
logpath— ім’я файлу логу, який буде аналізуватися щодо спроб некоректних з’єднань

Зверніть увагу на параметр maxretry. Не дивлячись на те, що він описаний в секції Default, він, як і інші параметри цієї секції, може бути перепризначений для будь-якої jail-

секції.

Файли з фільтрами знаходяться в теці */etc/fail2ban/filter.d ma* обов'язково мають розширення conf.

Наприклад у файлі */etc/fail2ban/filter.d/sshd.conf* знаходиться список регулярних виразів, за якими і визначаються некоректні спроби авторизації та перебору паролів.

PortKnocking

Portknocking — це спосіб зовнішнього відкриття портів на файєрволі, генеруючи спроби з'єднання на кілька заздалегідь визначених закритих портів. Після правильної послідовності спроб з'єднання динамічно змінюються правила файєрвола, що дозволяє хосту, який відправив пакети, з'єднатися через певний порт.

Основною метою Portknocking є захист від сканування. Зловмисник при спробі сканування портів для визначення потенційно небезпечних сервісів швидше за все не потрапить на потрібну послідовність і, відповідно, важливий порт йому залишиться недоступним.

Portknocking може складатися з будь-якої кількості пакетів TCP, UDP або ICMP, що відправляються на певні порти на цільовій машині.

PortKnocking засобами IPTables

Створюємо ланцюжок для перевірки спроб з'єднань на порт, що захищається
 # iptables -N ssh_knock

Якщо за останні 60 секунд було 2 і більше стуків — блокуємо
 # iptables -A ssh_knock -m recent --rcheck --seconds 60 --hitcount 2 -j RETURN

Якщо за останні 10 секунд стук в потрібний порт був — дозволити з'єднання
 # iptables -A ssh_knock -m recent --rcheck --seconds 10 -j ACCEPT

Дозволити пакети по встановленим з'єднанням
 # iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

Всі спроби відкрити нове з'єднання по SSH направляємо на перевірку
 # iptables -A INPUT -m conntrack --ctstate NEW -p tcp --dport 22 -j ssh_knock

Тут ми додаємо правило для реєстрації стуку
 # iptables -A INPUT -m conntrack --ctstate NEW -p tcp --dport 27520 -m recent --set

Знову ж про всяк випадок — при стукоті в сусідні порти закриваємо SSH
 # iptables -A INPUT -m conntrack --ctstate NEW -p tcp -m multiport --dport 27519,27521 -m recent --remove

Та забороняємо всі вхідні з'єднання
 # iptables -P INPUT DROP

У даній конструкції присутній критерій resend

recent — це спеціальний критерій, що дозволяє запам'ятовувати пакети що проходять через нього, а потім використовувати отриману інформацію для прийняття рішень.

Точніше recent запам'ятовує не самі пакети, а їх кількість, час надходження, адресу джерела (в останніх версіях iptables також може запам'ятовувати й адресу призначення), а також, при необхідності, TTL.

Спочатку коротко розглянемо його опції:

--set — запам'ятати адресу джерела/призначення пакета (внести його у внутрішній список). Якщо такий запис вже присутній в списку — оновити час останнього доступу для нього. Зверніть увагу, що критерію recent з опцією --set задовольняють всі пакети. Щоб йому не задовольняв жоден пакет — використовуйте логічну інверсію (! --set).

--rcheck — дозволяє перевірити, чи присутня адреса джерела/призначення пакета у внутрішньому списку. Критерій recent з цією опцією поверне істину, якщо адреса в списку присутня.

--update — працює аналогічно --rcheck, але ще і оновлює час останнього доступу для цього запису.

--remove — працює аналогічно --rcheck, але ще й видаляє знайдений запис зі списку. Якщо запис не знайдено, пакет вважається таким, що не відповідає критерію.

--seconds число — додаткова опція в режимах --rcheck та --update. Пакет вважається відповідним критерію, тільки якщо останній доступ до запису був не пізніше, ніж вказане число секунд тому.

--hitcount число — додаткова опція в режимах --rcheck та --update. Пакет вважається таким, що задовольняє критерію, якщо було не менше вказаного числа звернень до цього запису. Зазвичай використовується разом з --seconds — тоді критерій має зміст «не менше n звернень за останні m секунд».

Зауважимо, що, хоча в офіційній документації для параметрів --seconds та --hitcount вказується можливість заперечення (зазначення перед ними знака оклику), насправді таке заперечення ніяк не обробляється в коді і не змінює сенсу параметрів, так що вказувати його безглуздо. Ця помилка в документації виправлена у версії iptables 1.4.7.

--rttl — додатково до адреси джерела/призначення, заносити в список та перевіряти ще й TTL пакету.

--rsource — ця опція з'явилася в останніх версіях iptables, з того часу, як критерій recent почав підтримувати запам'ятовування адрес не тільки джерела, а й призначення. Дозволяє явно вказати, що в список вноситься саме адреса джерела пакету. Однак з метою забезпечення сумісності цей режим використовується за умовчанням, й тому вказувати цю опцію не обов'язково.

--rdest — ця опція з'явилася в останніх версіях iptables. Дозволяє явно вказати, що в список вноситься саме адреса призначення пакету.

Демон knockd для налаштування PortKnocking

Налаштування Port Knocking можна зробити самому в повністю ручному режимі за допомогою IPTables, однак простіше та зручніше для цього використовувати спеціальний демон knockd.

Встановимо його командою:

 # apt install knockd

Налаштування сервісу робляться в файлі */etc/knockd.conf*
За замовчуванням він виглядає так:

```
[options]
        UseSyslog

[openSSH]
        sequence    = 7000,8000,9000
        seq_timeout = 15
        command     = /sbin/iptables -A INPUT -s %IP% -p tcp --dport 22 -j ACCEPT
        tcpflags    = syn


[closeSSH]
        sequence    = 9000,8000,7000
        seq_timeout = 5
        command     = /sbin/iptables -D INPUT -s %IP% -p tcp --dport 22 -j ACCEPT
        tcpflags    = syn
```

В цьоиу файлі описані дві дії — відкрити порт для ssh та закрити його ж.
Опис полів:
sequence — список портів, на які треба «постукати»
seq_timeout — максимальний час в секундах, який відводиться на «простукування» портів
command— команда, яка буде виконана
tcpflags — прапорці пакетів

Після встановлення також слід дозволити запуск демона knockd. Для цього в файлі */etc/default/knockd* треба прописати рядок:
START_KNOCKD=1

Однак сенсу в цих налаштуваннях поки що немає. В IPTables за замовчуванням всі порти відкриті. Для того, що б не відстежувати послідовність правил фільтрації, в таблиці filter в ланцюжку INPUT задамо правило за замовчуванням, що забороняє всі з'єдання:
 # iptables -P INPUT DROP

Та перезапустивj сервіс:
#service knockd restart

Тепер у нас за замовчуванням всі порти закриті. Для того, щоб з'єднатися з сервером потрібно до нього постукати. Звичайно, це можна зробити за допомогою telnet, але є більш зручний шлях.
Якщо ми намагаємося з'єднатися з Linux-комп'ютера, то встановимо на ньому той же самий knockd, але не будемо запускати сервіс, а скористаємося клієнтською частиною:
$ knock 192.168.0.140 7000 8000 9000

Після чого можна з'єднуватися з нашим сервером:
$ ssh user@ 192.168.7.140

Якщо ми використовуємо Windows-комп'ютер, то можна завантажити knock-утиліту
www.zeroflux.org/proj/knock/files/knock-cygwin.zip

Та після її запуску з командного рядка (параметри такі ж як та в Linux) з'єднатися з сервером за допомогою Putty, як зазвичай.

Після того, як ми «постукали» в syslog повинні з'явитися записи на зразок:

knockd: 192.168.0.5: openSSH: Stage 1
knockd: 192.168.0.5: openSSH: Stage 2
knockd: 192.168.0.5: openSSH: Stage 3
knockd: 192.168.0.5: openSSH: OPEN SESAME
knockd: openSSH: running command: /sbin/iptables -A INPUT -s 192.168.0.5 -p tcp --dport 22 -j ACCEPT

Після закінчення роботи потрібно не забути закрити порт:

$ knock 192.168.0.140 9000 8000 7000

Після цього knockd виконає видалення правила на відкриття 22 порту.

У такого підходу є певні недоліки — не можна просто обірвати сесію з сервером та продовжити займатися іншими справами. В такому випадку порт на сервері буде відкритий або до перезавантаження, або поки ми не згадаємо і не скасуємо дозволяюче правило. Щоб уникнути такої ситуації правило на «стукіт» по портам можна записати в дещо іншому вигляді.

Приведемо файл /etc/knockd.conf до наступного вигляду:

```
[options]
        UseSyslog

[openSSH]
        sequence    = 7000,8000,9000
        seq_timeout = 15
        tcpflags    = syn
        start_command     = /sbin/iptables -A INPUT -s %IP% -p tcp --dport 22 -j ACCEPT
        cmd_timeout = 10
        stop_command     = /sbin/iptables -D INPUT -s %IP% -p tcp --dport 22 -j ACCEPT
```

Зараз у нас дещо змінився опис реакції на «стук» по портах:

start_command — команда, яка буде виконана

cmd_timeout — час затримки в секундах

stop_command — команда, яка буде виконана через *cmd_timeout* секунд після виконання *start_command.*

Та перезапустимо сервіс knockd

#service knockd restart

Якщо ми зараз спробуємо «постукати» та приєднатися до сервера, то через 10 секунд наша сесія обірветься, так як спрацює правило закриття порту, та не факт, що ми встигнемо зробити все, що потрібно.

Для того, що б сесія не обривалася на сервері в IPTables потрібно додати правило, яке дозволятиме вхідний трафік, за умови, що він відноситься до вже встановленого з'єднанню:

 # iptables -A INPUT -i eth0 -m state --state ESTABLISHED,RELATED -j ACCEPT

Ось після цього ми зможемо «постукати» та після цього нормально відкрити сесію й попрацювати скільки потрібно. Однак через 10 секунд нову сесію відкрити ми вже не зможемо.

Поштовий сервер на основі Postfix та Dovecot

Сьогодні кожна компанія використовує електронну пошту як один з основних засобів комунікацій у бізнесі. Необхідною умовою ефективного застосування пошти є наявність спеціальної програми — поштового сервера. Створювати корпоративні ящики за допомогою публічних сервісів в Інтернеті вже давно вважається свого роду дурним тоном. До того ж, такий підхід абсолютно небезпечний і супроводжується труднощами в керуванні.

Використання власного корпоративного поштового сервера має ряд переваг:

- можливість створення необмеженої кількості поштових доменів (для різних компаній, юридичних осіб, напрямків, підрозділів) на одному сервері;
- можливість створення необмеженої кількості поштових скриньок для різних поштових доменів;
- можливість контролю вхідної та вихідної пошти;
- створення поштових скриньок необмежених розмірів (обмеження обумовлюється тільки технічними характеристиками сервера)
- підтримка протоколів передачі з шифруванням пошти;
- можливість централізованого зберігання пошти на сервері;
- антивірусний захист вхідної/вихідної пошти;
- захист від спаму;
- web-інтерфейс доступу до електронної пошти з шифруванням даних, що передаються.

Поштовий сервер або сервер електронної пошти — в системі пересилки електронної пошти так зазвичай називають агент пересилки повідомлень.

В Linux-системах сервери доставки повідомлень і сервери передачі повідомлень розділені і адміністратор зв'язує їх між собою самостійно. Найбільш популярна комбінація: Postfix та Dovecot.

Що таке Postfix

Postfix — це агент передачі пошти (Mail Transfer Agent).

Postfix є вільним програмним забезпеченням.

Postfix створювався як альтернатива Sendmail. Вважається, що Postfix швидше працює, легше в адмініструванні, більш захищений і, що важливо, сумісний з Sendmail.

Спочатку Postfix був розроблений Вейтсом Венемой в той час, коли він працював у Дослідницькому центрі імені Томаса Уотсона компанії IBM. Перші версії програми стали доступні в середині 1999 року.

Postfix вирізняється продуманою модульної архітектурою, яка дозволяє створити дуже надійну і швидку поштову систему. Так, наприклад, привілеї root потрібні тільки для відкриття 25 порту, а демони, які виконують основну роботу, можуть працювати непривілейованим користувачем в ізольованому оточенні, що дуже позитивно позначається на безпеці.

Архітектура Postfix виконана в стилі UNIX — де прості програми виконують мінімальний набір функцій, але виконують їх швидко та надійно. При простої поштової системи непотрібні демони можуть припиняти свою роботу, вивільняючи тим самим пам'ять, а при необхідності знову запускаються master-демоном.

Також варто відзначити більш просту і зрозумілу конфігурацію в порівнянні з Sendmail і меншу ресурсоємність, особливо під час простою поштової системи.

Postfix сумісний з, BSD, HP-UX, IRIX, GNU/Linux, Mac OS X, Solaris, фактично може

бути зібраний на будь Unix-подібної операційної системи, що підтримує POSIX і має компілятор C.

Що таке Dovecot

Dovecot — вільний IMAP- і POP3-сервер, що розробляється в розрахунку на безпеку, гнучкість налаштування та швидкодію. Перший реліз відбувся у 2002 році

Dovecot підтримує формати поштових скриньок mbox та Maildir, а також власні формати dbox та Cydir

Особливості Dovecot, це:

- Висока швидкодія завдяки індексації вмісту ящиків.
- Велика кількість підтримуваних механізмів зберігання аутентифікаційної інформації (включаючи LDAP) і самої аутентифікації (підтримується SSL).
- Власна реалізація SASL. Postfix 2.3 + та Exim 4.64 + можуть аутентифікуватися безпосередньо через Dovecot.
- Розширюваність за допомогою плагінів.
- Можливість модифікації індексів з декількох комп'ютерів - що дозволяє йому працювати з NFS та кластерними файловими системами.
- Підтримує різні види квот
- Підтримка різних ОС: Linux, Solaris, FreeBSD, OpenBSD, NetBSD і Mac OS X
- Простота налаштування.
- Суворе дотримання стандартів — Dovecot один з небагатьох хто проходить тест на відповідність усім стандартам IMAP. До речі, прошу звернути увагу, що Microsoft Exchange цей тест не проходить.

В архітектурі Dovecot велика увага приділяється безпеці.

Автор пропонує € 1000 першому, хто виявить віддалену уразливість в Dovecot, яку можна експлуатувати.

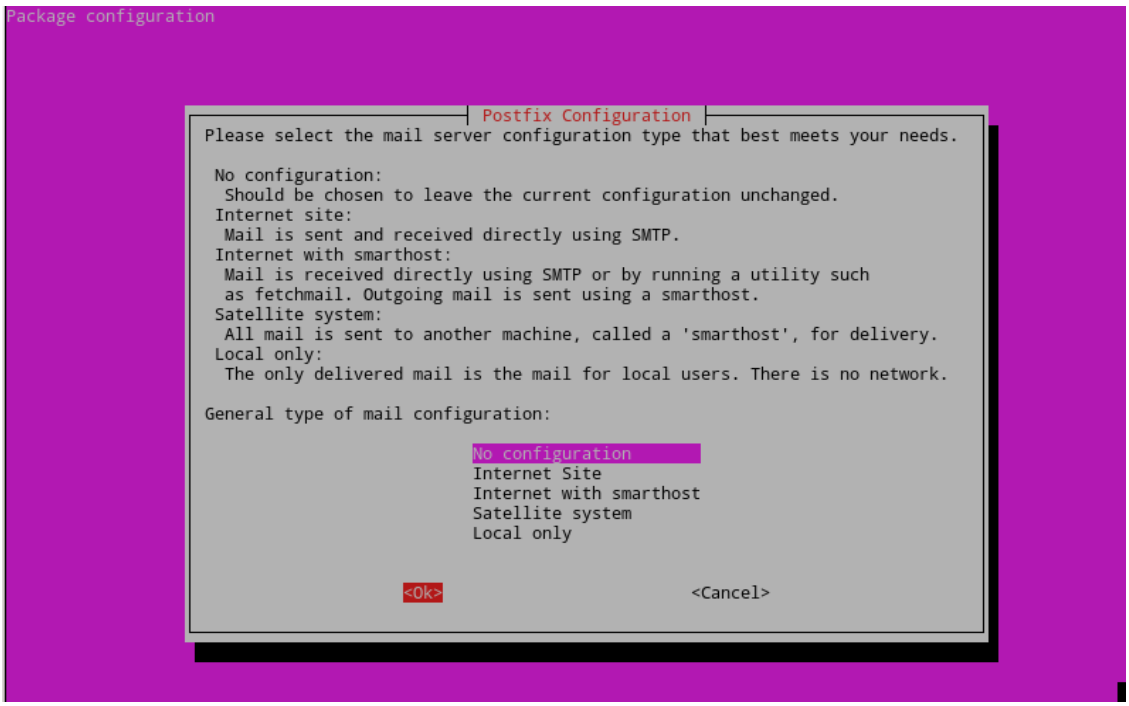
За 3 роки не було знайдено жодної проблеми, яку можна вважати віддаленої уразливістю.

Встановлення поштового сервера

Спочатку встановлюємо Postfix та Dovecot:

$sudo apt install postfix dovecot-core dovecot-imapd

Відмовляємося від запропонованих варіантів налаштувань. Мається на увазі, що все налаштуємо самі без жодного автоматизму з боку розробників.



Постінсталляційний скрипт повідомить, що в такому стані Postfix працювати не може. Створимо відсутній файл конфігурації:
$ sudo touch /etc/postfix/main.cf

На цьому встановлення завершене.

Підготовка до налаштування поштового сервера:

Створимо місце для зберігання пошти на сервері для нашого поштового домену study.local:
#mkdir -p /var/spool/mail/study.local

Створимо групу *virtual* та користувача *virtual*:
#groupadd -g 5000 virtual
#useradd -g virtual -u 5000 virtual

Для них ми призначили uid та gid 5000. Число було обрано довільно, як достатньо велике.

Вкажемо власника та права доступу до теки з поштою:
#chown virtual:virtual /var/spool/mail/study.local
#chmod 770 /var/spool/mail/study.local

Налаштування Postfix

Відкриваємо на редагування файл */etc/postfix/main.cf* та приведемо його до наступного вигляду:

#Так наш сервер буде представлятися при відправці та отриманні пошти
smtpd_banner = $myhostname ESMTP (ubuntu)
biff = no #Вимикаємо використання comsat
#Забороняємо автоматично доповнювати неповне доменне ім'я в адресі листа
append_dot_mydomain = no
queue_directory = /var/spool/postfix #Вказуємо теку черги для Postfix
myhostname = mail.study.local #Вказуємо ім'я нашого хоста
alias_maps =
myorigin = study.local
mydestination = localhost #Вказуємо, для яких доменів будемо приймати пошту

#Вказуємо, для яких віртуальних доменів будемо приймати пошту
virtual_mailbox_domains = study.local
virtual_mailbox_base = /var/spool/mail/ #Початок шляху для зберігання пошти
virtual_alias_maps = hash:/etc/postfix/virtual #Файл з описом поштових аліасів
virtual_mailbox_maps = hash:/etc/postfix/vmailbox #Файл з описом поштових скриньок
virtual_minimum_uid = 100
virtual_uid_maps = static:5000
virtual_gid_maps = static:5000
mynetworks = 127.0.0.0/8 #Вказуємо список довірених підмереж
recipient_delimiter = +
inet_interfaces = all #Приймаємо з'єднання на всіх інтерфейсах
#Описуємо авторизацію через Dovecot
smtpd_sasl_auth_enable = yes
smtpd_sasl_type = dovecot
smtpd_sasl_path = private/auth
smtpd_sasl_security_options = noanonymous
broken_sasl_auth_clients = yes
smtpd_helo_required = yes #Обов'язково при з'єднанні вимагати helo
#Далі налаштовуємо фільтри прийому/відправлення пошти
smtpd_recipient_restrictions = permit_mynetworks,
permit_sasl_authenticated,
check_helo_access hash:/etc/postfix/helo.list,
check_sender_access hash:/etc/postfix/ext_sender,
reject_unauth_destination,

reject_unknown_sender_domain,
reject_unknown_recipient_domain,
reject_non_fqdn_recipient,
reject_non_fqdn_sender,
reject_non_fqdn_hostname,
reject_invalid_hostname,
reject_unknown_hostname

Створимо файл /etc/postfix/helo.list

#touch /etc/postfix/helo.list

Відкриємо його на редагування та запишемо в нього рядок:

study.local 550 Don't use my hostname

Та створимо з нього індексовану мапу:

#postmap /etc/postfix/helo.list

Створимо файл /etc/postfix/ext_sender

#touch /etc/postfix/ext_sender

Відкриємо його на редагування та запишемо в нього рядок:

study.local 550 Do not use my domain in your envelope sender

Та створимо з нього індексовану мапу:

#postmap /etc/postfix/ext_sender

Налаштування Dovecot

Проведемо налаштування Dovecot версії 2.xx.

Тепер в теці /etc/dovecot, на відміну від старих версій програми, ми маємо багато файлів конфігурації. При чому навіть з підтеками.
Звичайно можна всю конфігурацію звести в один файл, але це не дуже правильно, бо суперечить тому, що задумали розробники.
Відкриємо основний файл конфігурації */etc/dovecot/dovecot.conf* та приведемо його до наступного вигляду:

```
# За яким протоколом працюємо
protocols = imap
# Слухаємо з'єднання на всіх інтерфейсах по протоколу IPv4
listen = *
# Робоча тека
base_dir = /var/run/dovecot/
# Ім'я інстансу (для відображення в лозі)
instance_name = dovecot
# Рядок привітання
```

login_greeting = Dovecot ready.
 # Відключати клієнтські з'єднання при виключенні або перезавантаженні майстер-сервісу
shutdown_clients = yes
 # Сокет керуючого сервісу doveadm
doveadm_socket_path = doveadm-server
 # Підключаємо окремі файли конфігурації
!include conf.d/.conf*

Тепер переходимо до теки */etc/dovecot/conf.d*
Відкриємо в ній файл *10-auth.conf* і пропишемо в ньому два рядки:

disable_plaintext_auth = no
auth_mechanisms = plain login

а також в кінці цього файлу закоментуємо рядок
!*include auth-system.conf.ext*

та розкоментуємо
!include auth-passwdfile.conf.ext

Далі відредагуємо файл *10-mail.conf*
mail_location = maildir:/var/spool/mail/study.local/%n
mail_uid = 5000
mail_gid = 5000
mail_privileged_group = virtual
valid_chroot_dirs = /var/spool/mail/

Далі нас буде цікавити файл *10-master.conf*

service imap-login {
inet_listener imap {
#port = 143
}
inet_listener imaps {
#port = 993
#ssl = yes
}
}
service auth {
 # Postfix smtp-auth
unix_listener /var/spool/postfix/private/auth {
mode = 0666
}
 # Auth process is run as this user.
user = postfix
group = postfix
}
І, нарешті, в файлі *10-ssl.conf* треба прописати

ssl = no

Наостаннє, треба видалити файл *15-mailboxes.conf*

Створення поштових скриньок та псевдонімів

Тепер створимо користувача та поштову скриньку для нього:
Логін — user
Пароль — user
Адреса — user@study.local

Створимо необхідні файли в Postfix:
 # touch /etc/postfix/vmailbox
 # touch /etc/postfix/virtual

Пропишемо в Postfix дані про нову поштову скриньку. Для цього в файл */etc/postfix/vmailbox* допишемо рядок:
user*@study.local study.local/user/*

Створимо для прикладу аліас на цю поштову скриньку. Для цього в файл */etc/postfix/virtual* допишемо рядок:
postmaster@study.local user@study.local

Та створимо індексовану мапу з цих файлів:
#postmap /etc/postfix/virtual
#postmap /etc/postfix/vmailbox

Тепер потрібно перезапустити Postfix:
 # service postfix restart

Внесемо дані про нашого користувача в Dovecot.
Якщо подивитися файл *auth-passwdfile.conf.ext* то ми побачимо, що логіни та паролі користувачів мають зберігатися у файлі */etc/dovecot/users* зі схемою шифрування CRYPT
Створимо запис для користувача user з паролем user.
$doveadm pw -s CRYPT -u user -p user

Отримані дані внесемо до файлу */etc/dovecot/users*
user:{CRYPT}CaKFEZXiRl/aE:5000:5000

Поштовий веб-інтерфейс RoundCube

RoundCube Webmail — це клієнт для роботи з електронною поштою з веб-інтерфейсом, написаний на PHP з використанням CSS та XHTML і технології AJAX. RoundCube Webmail встановлюється практично на будь-який сервер з підтримкою PHP та MySQL і надає можливість роботи з поштовими скриньками за протоколами IMAP та SMTP.

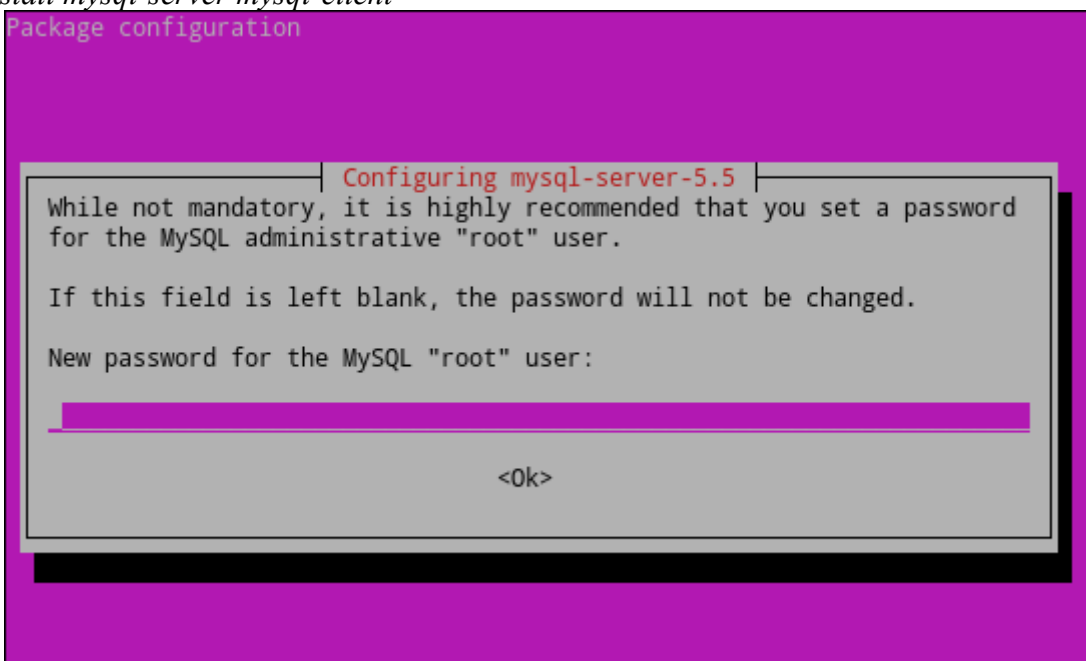
Проект був заснований 18 травня 2005 року. Тоді RoundCube Webmail був скромний клієнт для роботи з електронною поштою. Зараз це потужний поштовий додаток, який майже нічим не поступається звичайним поштовим клієнтам.

RoundCube Webmail доступний за ліцензією GPL та є вільним програмним забезпеченням.

Встановлення roundcube

Для своєї роботи, а точніше для зберігання даних, кешування пошти та внутрішньої адресної книги RoundCube вимагає наявності MySQL-сервера. Так як у нас його в мережі поки-то немає, то встановимо його локально

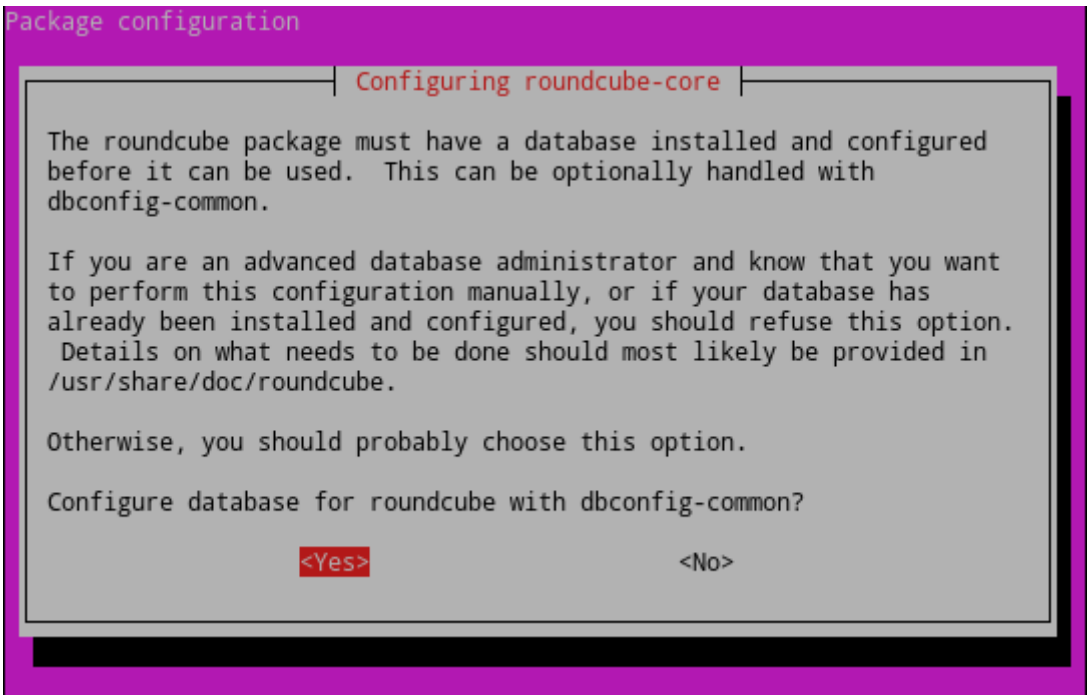
 # apt install mysql-server mysql-client



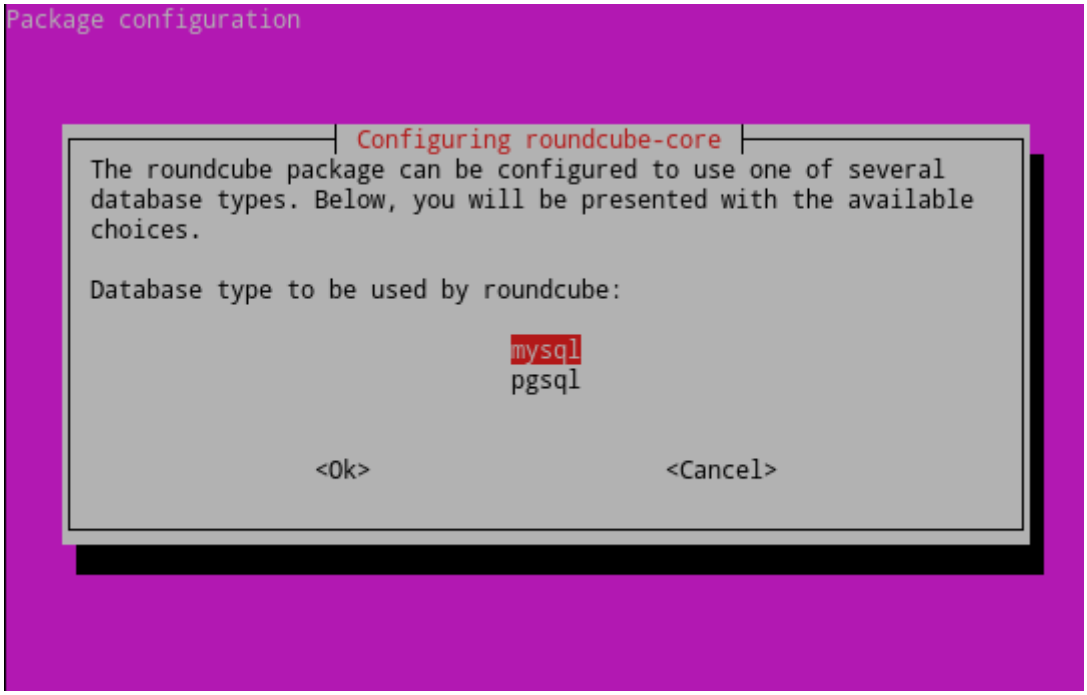
В процесі встановлення введемо пароль для користувача root нашого MySQL-сервера

Тепер можна встановлювати безпосередньо і сам RoundCube
 # apt install roundcube

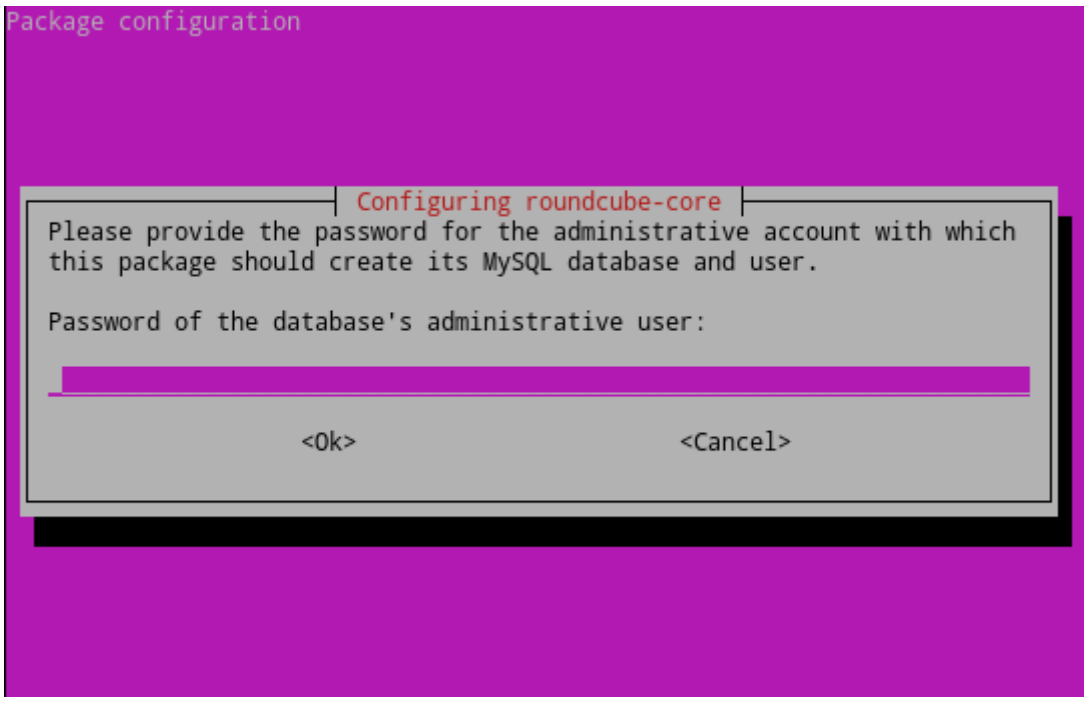
В процесі встановлення нам буде запропоновано автоматично налаштувати базу даних для RoundCube за допомогою *dbconfig-common*



Оберемо тип бази даних - mysql



Та вводимо пароль користувача root сервера MySQL



Потім нам запропонують ввести пароль для користувача roundcube. Його можна залишити порожнім. В цьому випадку пароль буде створений випадковим чином.

На цьому встановлення завершене.

Зараз необхідно налаштувати RoundCube для коректної роботи з нашим поштовим сервером.

Налаштування RoundCube

Відкриємо файл конфігурацфї RoundCube — /etc/roundcube/main.inc.php
В ньому необхідно змінити кілька рядків:

В розділі imap
$rcmail_config['default_host'] = array("127.0.0.1");

В розділі smtp
$rcmail_config['smtp_server'] = '127.0.0.1';

В розділі system
$rcmail_config['mail_domain'] = 'study.local';

Налаштування сервера Apache

Налаштуємо ваш веб-сервер, для того, щоб поштовий інтерфейс працював по захищеному протоколу https.

Для цього підключимо до Apache необхідні розширення

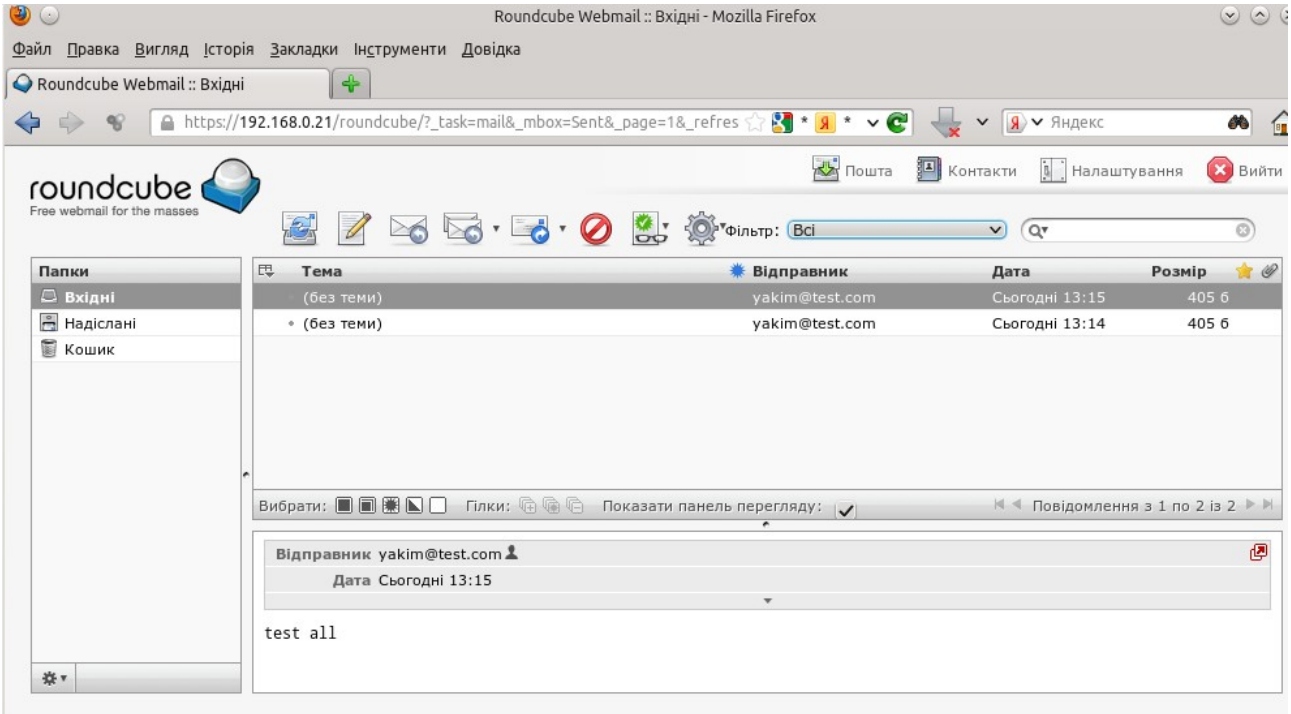
*# a2enmod ssl**

Тепер підключимо https-сайт за замовчанням
 # a2ensite default-ssl
Та ще в файлі */etc/apache2/conf.d/roundcube* раскоментуємо рядок
Alias /roundcube /var/lib/roundcube

Після цих дій необхідно перезапустити веб-сервер
#service apache2 restart

Після цих дій необхідно перезапустити веб-сервер
#service apache2 restart

Тепер веб-інтерфейс для нашого поштового сервера доступний за посиланнямhttps://ip-addr/roundcube



Налаштування поштового антивірусу clamav

Встановимо антивірусну систему для поштового сервера:

#apt install clamsmtp

Відкриємо файл конфігурації */etc/clamsmtpd.conf* та запишемо туди потрібні параметри:

OutAddress: 10026

Listen: 127.0.0.1:10025

ClamAddress: /var/run/clamav/clamd.ctl

Header: X-AV-Checked: ClamAV using ClamSMTP

TempDirectory: /var/spool/clamsmtp

PidFile: /var/run/clamsmtp/clamsmtpd.pid

Quarantine: on

User: clamsmtp

#VirusAction: /etc/clamav/script.sh

Насправді, параметрів в цьому файлі може бути більше, але я вказав лише необхідні.

Для більш детального вивчення рекомендую почитати man clamsmtpd.conf.

Для застосування змін необхідно перезапустити сервіс антивірусу:

#service clamsmtp restart

Налаштування поштового сервера для роботи з антивірусом

В файл */etc/postfix/main.cf* додамо 2 рядки:

content_filter = scan:[127.0.0.1]:10025

receive_override_options = no_address_mappings

Перша говорить postfix'у про те, що необхідно пересилати всю пошту через сервіс (фільтр) 'scan' на 10025-ий порт, на якому слухає clamsmtpd. Друга строчка каже, щоб postfix не робив ніяких маніпуляцій з адресами до того, як вони дійдуть до content_filter. Так що виходить, що фільтр працює з реальними поштовими адресами, а не з результатами переведення у віртуальні псевдоніми, маскарадингом і т.п.

В файл /etc/postfix/master.cf необхідно додати такі рядки:

 # AV scan filter (used by content_filter)

scan unix - - n - 16 smtp

-o smtp_send_xforward_command=yes

 # For injecting mail back into postfix from the filter

127.0.0.1:10026 inet n - n - 16 smtpd

-o content_filter=

-o receive_override_options=no_unknown_recipient_checks,no_header_body_checks

-o smtpd_helo_restrictions=

-o smtpd_client_restrictions=

-o smtpd_sender_restrictions=

-o smtpd_recipient_restrictions=permit_mynetworks,reject

-o mynetworks_style=host

-o smtpd_authorized_xforward_hosts=127.0.0.0/8

Лишилось перезапустити сервіс Postfix:

#service postfix restart

На цьому базове налаштування антивірусу завершене.

Налаштування повідомлень антивірусу

Тепер залишилася остання дія — налаштувати відсилання повідомлень антивірусу. Для цього створимо файл script.sh:

#nano /etc/clamav/script.sh

Та запишемо в нього:

```
#!/bin/sh
DOMAIN=study.local
# Email address to send alerts to
ADMIN=postmaster@study.local
# formail should be in PATH
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin
LINE="-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-"
if [ X`echo $SENDER | egrep $DOMAIN` != "X" ];
then MAILTO=$SENDER,$ADMIN
else MAILTO=`echo "$RECIPIENTS" | egrep $DOMAIN | tr '\n' ','`$ADMIN
fi
(echo "Virus name: $VIRUS"
echo "Sender: $SENDER"
echo "Recipient(s): $RECIPIENTS"
echo
if [ "x$EMAIL" != "x" ] && [ -f $EMAIL ]
then
echo "Quarantined to: $EMAIL"
fi
) | cat -v | mail -s "$VIRUS found on mailserver" $MAILTO
```

Розкоментуємо рядок *VirusAction: /etc/clamav/script.sh* у файлі */etc/clamsmtpd.conf* та перезапустимо сервіс clamsmtp:

#service clamsmtp restart

Зараз у нас антивірус не тільки перевіряє пошту, але й складає заражені листи в карантин, відсилаючи повідомлення про це адміністратору і користувачам нашого домену.

Для відправлення повідомлень треба встановити пакет mailutils

 # apt install mailutils

Налаштування DNS та DHCP серверів

Що таке DNS

DNS — комп'ютерна розподілена система для отримання інформації про домени. Найчастіше використовується для отримання IP-адреси за іменем хоста, отримання інформації про маршрутизацію пошти, вузлах, які обслуговують протоколи у домені.

Розподілена база даних DNS підтримується за допомогою ієрархії DNS-серверів, що взаємодіють за певним протоколом.

Основою DNS є уявлення про ієрархічну структуру доменного імені та зони. Кожен сервер, який відповідає за ім'я, може делегувати відповідальність за подальшу частину домену іншому серверу (з адміністративної точки зору — іншій організації або людині), що дозволяє покласти відповідальність за актуальність інформації на сервери різних організацій (людей), що відповідають тільки за «свою» частину доменного імені.

DNS важливий для роботи Інтернету, тому що для з'єднання з вузлом необхідна інформація про його IP-адресу, а для людей простіше запам'ятовувати буквені (зазвичай осмислені) адреси, ніж послідовність цифр IP-адреси. В деяких випадках це дозволяє використовувати віртуальні сервери, наприклад, HTTP-сервери, розрізняючи їх по імені запиту. Спочатку перетворення між доменними та IP-адресами проводилося з використанням спеціального текстового файлу hosts, який складався централізовано й автоматично розсилався на кожну з машин у своїй локальній мережі. З ростом Мережі виникла необхідність в ефективному, автоматизованому механізмі, яким і став DNS.

DNS сервер BIND

BIND (*Berkeley Internet Name Domain*) — відкрита і найбільш поширена реалізація DNS-сервера, що забезпечує виконання перетворення DNS-імені в IP-адресу і навпаки.

BIND був створений студентами і вперше був випущений в BSD 4.3. В Unix цей сервер де-факто є стандартом.

Що таке DHCP

DHCP — це мережевий протокол, що дозволяє комп'ютерам автоматично отримувати IP-адресу та інші параметри, необхідні для роботи в мережі TCP/IP. Даний протокол працює за моделлю «клієнт-сервер». Для автоматичної конфігурації комп'ютер-клієнт на етапі конфігурації мережевого пристрою звертається до так званого серверу DHCP, і отримує від нього потрібні параметри. Адміністратор може задати діапазон адрес, що розподіляються сервером серед комп'ютерів. Це дозволяє уникнути ручного налаштування комп'ютерів мережі й зменшує кількість помилок. Протокол DHCP використовується в більшості мереж TCP/IP.

DHCP є розширенням протоколу BOOTP, що використовувався раніше для забезпечення бездискових робочих станцій IP-адресами при їх завантаженні. DHCP зберігає зворотню сумісність з BOOTP.

Деякі реалізації служби DHCP здатні автоматично оновлювати записи DNS, відповідні клієнтським комп'ютерам, при виділенні їм нових адрес. Це здійснюється за допомогою протоколу поновлення DNS.

Встановлення серверів

Встановимо сервери:

#apt install bind9 dhcp3-server

Для того, що б сервіс bind мав доступ на запис до теки /etc/bind/ необхідно змінити його профіль в AppArmor. Для цього відкриємо файл
#nano /etc/apparmor.d/usr.sbin.named
та змінимо в ньому рядок
*/etc/bind/** r,*
на
*/etc/bind/** rw,*

Після цього слід перезапустити сервіс apparmor:
#service apparmor restart

Перейдемо до теки /etc/bind
cd /etc/bind

та згенеруємо ключ, за яким будуть оновлюватися наші зони.
#dnssec-keygen -a HMAC-MD5 -b 128 -r /dev/urandom -n USER rndc-key

Ключ, що створився в результаті виконання цієї команди буде міститися у файлі */etc/bind/Krndc-key....key* в останньому його полі.
Його ми і будемо вносити у файли конфігурації bind та dhcp3-server

Налаштування Bind

Відкриємо файл конфигурації bind:
 # nano /etc/bind/named.conf.local

І приведемо його до наступного вигляду:

```
key  dnsupdater {
algorithm hmac-md5;
secret "4oz+o1d5QT/7QIH/Zqz1nw==";
};

zone "56.168.192.in-addr.arpa" IN {
type master;
check-names ignore;
file "/etc/bind/db.56.168.192";
allow-update { key dnsupdater; };
};
```

```
zone "study.local" IN {
type master;
check-names ignore;
file "/etc/bind/db.study.local";
allow-update { key dnsupdater; };
};
```

Далі, в теці /etc/bind створимо два файли з описом наших зон — прямої та зворотньої.

#nano /etc/bind/db.study.local

```
$ORIGIN .
$TTL 604800     ; 1 week
study.local             IN SOA  pdc.study.local. root.study.local. (
                        4541       ; serial
                        604800     ; refresh (1 week)
                        86400      ; retry (1 day)
                        2419200    ; expire (4 weeks)
                        604800     ; minimum (1 week)
                        )
                NS      pdc.study.local.
$ORIGIN study.local.
pdc             A       192.168.56.10
```

 # nano /etc/bind/db.56.168.192

```
$ORIGIN .
$TTL 604800     ; 1 week
56.168.192.in-addr.arpa IN SOA  pdc.study.local. root.study.local. (
                        2337       ; serial
                        604800     ; refresh (1 week)
                        86400      ; retry (1 day)
                        2419200    ; expire (4 weeks)
                        604800     ; minimum (1 week)
                        )
                NS      localhost.
$ORIGIN 56.168.192.in-addr.arpa.
10              PTR     pdc.study.local.
```

Пояснення до полів файлів зон:

Serial — серійний номер. Кожного разу при зміні яких-небудь даних його потрібно обов'язково міняти. Коли міняється серійний номер, зона оновлюється на всіх серверах. Використовуйте наступний формат: РРРРММДДнн (рік, місяць, день, нн - порядковий номер

зміни за день). Якщо ви вже другий раз за день вносите зміни в файл зони, вкажіть "нн" рівним 01, якщо третій — 02, і т.д.

Refresh — інтервал, через який slave сервери повинні звертатися до primary сервера та перевіряти оновлення зони.

Retry — якщо slave серверу не вдалося звернутися до primary сервера, через цей час він повинен повторити свій запит.

Expiry — якщо протягом цього часу slave сервер так і не зміг оновити зону з primary сервера, то slave повинен припинити обслуговувати цю зону.

TTL — час кешування негативних відповідей (відповідь "домен неможливо розв'язати в IP адресу").

NS — NS сервери, що обслуговують даний домен.

MX — запис для поштових серверів та пріоритет для них.

A — вказує домени.

PTR — зворотня зона

Налаштування DHCP

Налаштуємо сервер DHCP

Відкриємо його файл конфігурації:

#nano /etc/dhcp/dhcpd.conf

І приведемо його до ось такого ось вигляду:

```
key dnsupdater {
 algorithm hmac-md5;
 secret "4oz+o1d5QT/7QIH/Zqz1nw==";
};

option domain-name "study.local";
option domain-name-servers 192.168.56.10;
ddns-update-style interim;
ddns-updates              on;

zone study.local. {
primary 127.0.0.1;
key dnsupdater;
}
zone 56.168.192.IN-ADDR.ARPA. {
primary 127.0.0.1;
key dnsupdater;
}


default-lease-time 600;
max-lease-time 7200;
#update-static-leases on;
log-facility local7;

subnet 192.168.56.0 netmask 255.255.255.0 {
  range 192.168.56.11 192.168.56.50;
```

```
  option broadcast-address 192.168.56.255;
  option routers 192.168.56.10;
}

#host client {
#  hardware ethernet 08:00:27:3a:6e:5a;
#  fixed-address 192.168.56.20;
#}
```

Зверніть увагу!

Останній закоментований блок можна використовувати як приклад для резервування адрес за MAC-адресами комп'ютерів.

У такому випадку слід розкоментувати рядок

update-static-leases on;

Це дасть можливість оновлювати в DNS дані про комп'ютери із зарезервованою адресою.

Зараз необхідно змінити власника теки */etc/bind* та її вмісту:

chown bind:bind -R /etc/bind

Залишилося перестартувати сервіси bind та dhcp

#service bind9 restart

#service isc-dhcp-server restart

Сервер готовий до роботи!

Поради з налаштування сервера для роботи з DNS

При одержанні зовнішньої адреси по DHCP періодично буде відпадати використання локального DNS-серверу. Це пов'язано з тим, що у файл */etc/resolv.conf* прописуються тільки ті dns-сервери, що отримуються від провайдера.

Що б там постійно був і наш власний dns-сервер створюємо файл */etc/resolvconf/resolv.conf.d/tail*

Та записуємо в нього:

nameserver 127.0.0.1

Це стосується дистрибутива Ubuntu 12.04 та більш нових. В інших дистрибутивах у файлі /etc/dhcp3/dhclient.conf прописуємо рядок:

prepend domain-name-servers 127.0.0.1;

Частина ІІ. Додаткові офісні сервери

Налаштування файлового сервера на основі серверу Samba

Samba — пакет програм, які дозволяють звертатися до мережевих дисків і принтерів на різних операційних системах по протоколу SMB/CIFS. Має клієнтську і серверну частини. Є вільним програмним забезпеченням, випущена під ліцензією GPL.

Починаючи з третьої версії Samba надає служби файлів і друку для різних клієнтів Microsoft Windows і може інтегруватися з операційною системою Windows Server, або як основний контролер домену (PDC), або як член домену. Вона також може бути частиною домену Active Directory.

Samba працює на більшості Unix-подібних систем, таких, як Linux, POSIX-сумісних Solaris і Mac OS X Server, на різних варіантах BSD. Samba включена практично в усі дистрибутиви Linux.

Установка сервера Samba проводиться командою:
#apt install samba

Розглянемо варіанти налаштування Samba-сервера.

Налаштування безпарольного доступу до спільних тек.

Відкриємо файл */etc/samba/smb.conf* і заповнимо його наступним чином:

```
[global]
  workgroup = WORKGROUP
  hosts allow = 192.168.7.0/24
  interfaces = eth0
  log file = /var/log/samba/log.%m
  max log size = 1000
  syslog = 0
  guest account = nobody
  security = user
  map to guest = Bad User
  dos charset = cp866
  unix charset = UTF-8

[share]
  path = /var/local
  comment = share
  printable = no
  writable = yes
  guest ok = yes
  create mask = 0666
  directory mask = 0777
```

Тепер розглянемо докладно файл конфігурації.

У секції Global описуються загальні налаштування роботи сервера.

workgroup — записуємо ім'я нашої робочої групи або домену.

hosts allow — описуємо підмережу, яка буде мати доступ до нашого серверу.

interfaces — перераховуємо мережеві інтерфейси, на яких будуть прийматися з'єднання (можливе значення all).

log file = /var/log/samba/log.%m
max log size = 1000
syslog = 0
В цих змінних описуються параметри ведення логів нашого Samba-сервера

guest account — задаємо відповідність між користувачем Гість і системним користувачем.

dos charset — вказує кодування, в якій Samba буде спілкуватися з клієнтами, що не підтримують Unicode.

unix charset — вказує кодування комп'ютера на якому працює Samba.

Далі йдуть секції, які описують відповідні теки загального доступу. У нас ця секція одна — [share]

path — шлях до теки на сервері.

comment — мережеве ім'я теки.

writable — чи дозволений запис в теку загального доступу.

guest ok — чи дозволений гостьовий доступ.

create mask — описуються права для новоствореного файлу.

directory mask — описуються права для новоствореної теки.

Після створення нашого файлу конфігурації необхідно перезапустити сервіс Samba:
#service smbd restart

Тепер можна перевірити з'єднання з нашим файловим сервером з будь-якого комп'ютера в мережі.

Для цього в Windows відкриваємо в explorer посилання *\\samba-ip*
або в Linux виконуємо в консолі команду
smbclient -L samba-ip -N
В останньому випадку просто буде виведений список папок загального доступу на нашому сервері.

Налаштування парольного доступу до спільних тек.

Змінимо деякі параметри в секції *Global.* А точніше, замість *security = share* напишемо *security = user.*

Також слід внести зміни в секцію опису теки загального доступу. Видалимо рядок *guest ok = yes.*

та додамо рядок
write list = user1

Де user1 користувач, який має повний доступ до теки.
Тепер потрібно створити користувача user1 в системі:
#adduser user1
та додати системного користувача в користувачі Samba
#smbpasswd -a user1

Зверніть увагу! Пароль користувача в системі і пароль користувача до ресурсів Samba можуть бути різними.
Тепер доступ до нашої теки загального доступу буде з паролем.
Якщо завести аналогічним чином другого користувача, то в нього доступ до даної папці буде тільки для читання.

Можна повністю обмежити доступ користувача user2 до нашої теки загального доступу. Для цього на сервері створюється група, наприклад, share:
#addgroup share

І в цю групу додається користувач user1:
#addgroup user1 share

Далі на теку в системі виставляються права 770:
#chmod 770 /var/local
І змінюється власник теки:
#chgrp share /var/local/

Тепер користувач user2 взагалі не зможе зайти в нашу теку загального доступу.

Файловий сервер з доступом по SSH

Робота з Samba, і взагалі по протоколу SMB в Windows-мережі стала вже стандартом. Та й замислювалася вона, швидше за все, так само. Але сьогодні вже немає сенсу прив'язуватися тільки до цього протоколу. Він несе в собі занадто багато обмежень.

Якщо розглядати офіс, в якому встановлені тільки стаціонарні комп'ютери і робота з дому або взагалі з-за меж офісу не планується ніколи — то в такому випадку дійсно варто обмежитися роботою за SMB і не морочити собі голову. Зовсім інша справа, коли користувач працює за ноутбуком. Тут вже будувати систему потрібно так, що б у працівника були мінімальні (а краще, що б їх не було взагалі) відмінності в трудовому процесі, при роботі через інтернет.

Робота більшості сучасних серверних додатків (1С, GroupWare і т.д.) абсолютно спокійно робиться за допомогою веб-інтерфейсу. З поштою теж зазвичай проблем не буває. Доступ в інтернет при віддаленій роботі системного адміністратора взагалі не хвилює. Залишається остання проблема — доступ до файлів на файловому сервері.

В якості альтернативи протоколу SMB, розглянемо переваги роботи з файловим сховищем по SSH:

1. Тільки авторизований доступ.
2. Від початку жорстке розділення прав користувачів на доступ.
3. Відсутність відмінностей в роботі користувача, незалежно від його місцезнаходження.
4. Можливість авторизації користувача як за паролем (можлива інтеграція з Microsoft AD), так і по ключу.
5. Відсутність необхідності в закупівлі, налаштуванні та підтримці сервера VPN.
6. Відсутність потреби у закупівлі серверної ліцензії Microsoft Windows.

З недоліків можна знайти тільки один — це все таки незвичне для більшості системних адміністраторів рішення.

Підключення віддалених тек по SSHFS за допомогою AutoFS

На клієнтському комп'ютері встановимо sshfs і autofs. Виконаємо команду:
$sudo apt install sshfs autofs

Тепер налаштування.

Для початку зробимо можливим підключення по ssh з віддаленим сервером по ключу, а не за паролем:

Заходимо на свою машину з правами root:
$sudo su

Введемо свій пароль.

Генеруємо RSA ключі:

#ssh-keygen -t rsa

Відповідаємо на питання:

1. Enter file in which to save the key (/root/.ssh/id_rsa): — погоджуємося на значення за замовчуванням.

2. Enter passphrase (empty for no passphrase): — Залишаємо значення порожнім. Паролем користуватися не будемо.

3. Enter same passphrase again: — Знову залишаємо значення порожнім.

Скрипт створив 2 ключа: приватний і публічний.

Your identification has been saved in /root/.ssh/id_rsa. — секретний приватний ключ для декодування.

Your public key has been saved in /root/.ssh/id_rsa.pub. — публічний ключ для кодування.

Зараз потрібно скопіювати на сервер наш публічний ключ:
#ssh-copy-id -i ~/.ssh/id_rsa.pub user@server

Все. Тепер спробуємо залогінитися:
#ssh user@server

Тепер беремося за налаштування, власне, autofs.
Відкриваємо на редагування файл */etc/auto.master* і додаємо туди:
/home/user/server /etc/auto.sshfs --timeout=30,--ghost

де */home/user/server* - це тека, за якою буде стежити наш autofs.

Тепер створимо файл */etc/auto.sshfs* і запишемо в нього в один рядок:
home -fstype=fuse,rw,nodev,nonempty,noatime,allow_other,max_read=65536, reconnect,uid=1000,gid=1000 :sshfs\#user@server\:/home/user

Розлогінимось з-під root:

#exit

Створюємо теку, куди буде монтуватися тека на віддаленому сервері:
$mkdir /home/user/server

Та перезапускаємо сервіс autofs:

$sudo service autofs restart

Тепер у нас все налагоджено. При заході в теку */home/user/server* у нас автоматично примонтується сервер, що можна відразу ж перевірити.

Налаштування сервера.

Займемося налаштуванням сервера.
Для початку створимо теку, в якій і будуть розміщуватися теки загального доступу.
Нехай вона знаходиться в /home/share
#mkdir /home/share

Тепер створимо в ній дві підтеки — public та sales
#cd /home/share

#mkdir public

#mkdir sales

Відповідно потрібно буде створити і дві групи користувачів на сервері — public і sales
#addgroup public

#addgroup sales

Зараз створимо двох тестових користувачів — user1 і user2:
#adduser user1

#adduser user2

Тепер зробимо так, що б користувач user1 мав доступ тільки в public, а user2 — і в public, і в sales. Для цього введемо їх у відповідні групи:
#addgroup user1 public

#addgroup user2 public

#addgroup user2 sales

Налаштування користувачів на сервері закінчена. Тепер потрібно правильно встановити права на самі теки:
#chown root:public /home/share/public

#chown root:sales /home/share/sales

#chmod 770 /home/share/public

#chmod 770 /home/share/sales

Для того, щоб був нормальний доступ до файлів в теках, на ці теки потрібно ще встановити біт SGID.
#chmod g+ws,o= /home/share/public

#chmod g+ws,o= /home/share/sales

Залишився останній момент. Потрібно контролювати, що б у всіх файлів всередині цих тек були права 660, а у вкладених тек — 770. Якби у нас завідомо файли тільки створювалися, то не було б жодних проблем. Можна було б обійтися використанням umask. Однак користувачі періодично копіюють з різних джерел файли, у яких вже встановлені деякі права, що відрізняються від необхідних.

Для вирішення цієї проблеми потрібно встановити пакет inotify-tools:
 # apt install inotify-tools

Тепер в /etc/rc.local додамо рядок:
/usr/bin/inotifywait -mr --format '%w%f' -e close_write -e moved_to -e create /home/share | while read file; do /root/share.sh "$file"; done

Змінну $file в лапки брати обов'язково! Якщо цього не зробити, й у шляху зустрінуться імена файлів або тек з пробілами — скрипт не виконає необхідних дій.
Скрипт */root/share.sh* буде виглядати так:
#!/bin/bash
if [-d "$1"]; then
chmod 770 "$1"
elif [-f "$1"]; then
chmod 660 "$1"
fi

Налаштування сервера віртуальної приватної мережі на основі OpenVPN

VPN

VPN — віртуальна приватна мережа — узагальнена назва технологій, що дозволяють забезпечити одне або декілька мережевих з'єднань (логічну мережу) поверх іншої мережі (наприклад, Інтернет). Незважаючи на те, що комунікації здійснюються по мережах з меншим невідомим рівнем довіри (наприклад, по публічних мереж), рівень довіри до побудованої логічної мережі не залежить від рівня довіри до базових мереж завдяки використанню засобів криптографії (шифрування, аутентифікації, інфраструктури відкритих ключів, засобів для захисту від повторів і змін переданих по логічної мережі повідомлень).

В залежності від вживаних протоколів і призначення, VPN може забезпечувати з'єднання трьох видів: вузол-вузол, вузол-мережа та мережа-мережа.

Зазвичай VPN розгортають на рівнях не вище мережевого, так як застосування криптографії на цих рівнях дозволяє використовувати в незмінному вигляді транспортні протоколи (такі як TCP, UDP).

OpenVPN

OpenVPN — вільна реалізація технології Віртуальної Приватної Мережі (VPN) з відкритим вихідним кодом для створення зашифрованих каналів типу точка-точка або сервер-клієнти. Вона дозволяє встановлювати з'єднання між комп'ютерами, що знаходяться за NAT-firewall, без необхідності зміни їх налаштувань. OpenVPN була створена Джеймсом Йонаном (James Yonan) і розповсюджується під ліцензією GNU GPL.

Для забезпечення безпеки керуючого каналу і потоку даних, OpenVPN використовує бібліотеку OpenSSL. Завдяки цьому задіюється весь набір шифрів, доступних в даній бібліотеці. OpenVPN використовується на операційних системах Solaris, OpenBSD, FreeBSD, NetBSD, GNU/Linux, Apple Mac OS X, QNX і Microsoft Windows.

OpenVPN проводить всі мережеві операції через TCP, або UDP порт. Також можлива робота через більшу частину проксі серверів, через NAT і мережеві фільтри. Сервер може бути налаштований на призначення мережевих налаштувань клієнта. Наприклад: IP адресу, налаштування маршрутизації та параметри з'єднання. Також можливе використання бібліотеки компресії LZO, для стиснення потоку даних.

Використання в OpenVPN стандартних протоколів TCP і UDP дозволяє йому стати альтернативою IPsec в ситуаціях, коли Інтернет-провайдер блокує деякі VPN протоколи.

Налаштування OpenVPN

Встановлюємо необхідні пакети

 # apt install openvpn easy-rsa

Створюємо конфігураційний файл для сервера */etc/openvpn/server.conf*:

```
port 443
proto tcp
dev tun
ca /etc/openvpn/ca.crt
cert /etc/openvpn/server.crt
key /etc/openvpn/server.key
dh /etc/openvpn/dh2048.pem
server 10.10.10.0 255.255.255.0
ifconfig-pool-persist ipp.txt
push "route 10.0.0.0 255.255.255.0"
;duplicate-cn
keepalive 10 120
;cipher BF-CBC        # Blowfish (default)
;cipher AES-128-CBC   # AES
;cipher DES-EDE3-CBC  # Triple-DES
comp-lzo
user nobody
group nogroup
persist-key
persist-tun
status openvpn-status.log
log-append  openvpn.log
verb 4
mute 20
client-to-client
```

Роздивимось докладніше цей файл конфігурації.

Вказуємо, на якому порту приймати з'єднання:

port 443

Вказуємо протокол (tcp чи udp):

proto tcp

Зазначаємо, що саме будемо інкапсулювати в тунелі (ethernet фрейми — tap або ip пакети — tun)

dev tun

Прописуємо шляхи до ключів і сертифікатами:

ca /etc/openvpn/ca.crt

cert /etc/openvpn/test.crt

key /etc/openvpn/test.key

dh /etc/openvpn/dh2048.pem

Виділяємо внутрішній діапазон адрес для VPN:

server 10.10.10.0 255.255.255.0

В який файл будемо записувати видані для VPN адреси?:
ifconfig-pool-persist ipp.txt
Прописуємо на клієнті маршрут до домашньої мережі (тієї яка за сервером):
push "route 192.168.7.0 255.255.255.0"
Пінгувати кожні 10 секунд, якщо хост не відповідає протягом 120 секунд, вважати його недоступним:
keepalive 10 120
Дозволити стиснення пакетів:
comp-lzo
Прописуємо користувача і групу, від імені яких буде запускатися демон:
user nobody
group nogroup
Вказує не перечитувати файли ключів при перезапуску тунелю.
persist-key

Дана опція залишає без зміни пристрої tun/tap при перезавантаженні OpenVPN.
persist-tun
Рівень деталізації логів (від 0 до 9):
verb 4
Не писати в логи повідомлення, що повторюються:
mute 20
Дозволити пересилання пакетів між клієнтами:
client-to-client

Створення ключів та сертифікатів

Тепер необхідно створити ключі і сертифікати для шифрування та авторизації. Відповідні скрипти для цього знаходяться в папці /usr/share/doc/openvpn/examples/easy-rsa/2.0.
Створюємо CA (авторитетний сертифікат):
#cd /usr/share/easy-rsa
#. ./vars
#./clean-all
#./build-ca

Тепер створимо сертифікат і приватний ключ для сервера:
#./build-key-server server

Створюємо ключ для клієнта (якщо клієнтів декілька, процедуру доведеться повторити):
#./build-key client1

Примітка: для кожного клієнта повинно бути зазначено своє унікальне ім'я (в даному випадку client1).
Генеруємо параметри Діффі-Хеллмана:
#./build-dh

Розміщуємо наступні файли в теці */etc/openvpn/*

- *ca.crt*
- *server.crt*
- *dh2048.pem*

- *server.key*

Налаштування клієнта OpenVPN

Конфігураційний файл клієнтської машини */etc/openvpn/client.conf* буде виглядати наступним чином:

remote XX.XX.XX.XX 443
client
dev tun
proto tcp
resolv-retry infinite # this is necessary for DynDNS
nobind
user nobody
group nogroup
persist-key
persist-tun
ca /etc/openvpn/ca.crt
cert /etc/openvpn/client1.crt
key /etc/openvpn/client1.key
comp-lzo
verb 4
mute 20

Тепер необхідно скопіювати з сервера в теку */etc/openvpn/* згенеровані клієнтські ключі і авторитетний сертифікат сервера:

- *ca.crt*
- *client1.crt*
- *client1.key*

Після цього можна запускати клієнт OpenVPN і перевіряти зв'язок з сервером.

Налаштування FTP-сервера

Протокол FTP

FTP (англ. File Transfer Protocol — протокол передачі файлів) — стандартний протокол, призначений для передачі файлів через TCP-мережі (наприклад, Інтернет). FTP часто використовується для завантаження веб-сторінок та інших документів з приватного пристрою розробки на відкриті сервера хостингу.

Протокол побудований на архітектурі "клієнт-сервер" і використовує різні мережеві з'єднання для передачі команд і даних між клієнтом і сервером. Користувачі FTP можуть пройти аутентифікацію, передаючи логін і пароль відкритим текстом, або ж, якщо це дозволено на сервері, вони можуть підключитися анонімно. Можна використовувати протокол SSH для безпечної передачі, що приховує (шифрує) логін і пароль, а також шифрує вміст.

Перші клієнтські FTP-додатки були інтерактивними інструментами командного рядка, що реалізовують стандартні команди і синтаксис. Графічні інтерфейси з тих пір були розроблені для багатьох операційних систем. Серед цих інтерфейсів як програми загального веб-дизайну наприклад Microsoft Expression Web, так і спеціалізовані FTP-клієнти (наприклад, CuteFTP).

FTP є одним з найстаріших прикладних протоколів, що з'явилися задовго до HTTP, в 1971 році. Він і сьогодні широко використовується для розповсюдження ПЗ та доступу до віддалених хостів.

ProFTPd - FTP-сервер для Linux та UNIX-подібних операційних систем.

Сервер може бути налаштований для роботи декількох віртуальних хостів, також підтримує chroot. Може бути запущений у вигляді окремого серверу (демона) або в складі суперсервера inetd.

Встановлення та налаштування ftp-сервера ProFTPd

Встановимо ftp-сервер proftpd
 # apt install proftpd

Тепер відредагуємо файл конфігурації:
 # nano /etc/proftpd/proftpd.conf

DefaultRoot ~
RequireValidShell off

Відредагуємо файл з шеллами
 # nano /etc/shells

Додамо туди рядок:
/bin/false

Тепер перезапустимо FTP-сервер:
 # service proftpd restart

Створення користувачів для ftp-сервера

Тепер створимо тестового користувача на ім'я *testuser* та теку */home/ftp/testuser*, яка буде його домашньої текою.

Створюємо теку:
 # mkdir -p /home/ftp/testuser

Створюємо користувача *testuser* та призначаємо йому домашню теку */home/ftp/testuser*, а також забороняємо використовувати шелл:
 # useradd testuser -d /home/ftp/testuser -s /bin/false

Задаємо пароль для користувача *testuser*
 # passwd testuser

Та виставляємо правильного власника й права на створену нами теку:
 # chown -R testuser /home/ftp/testuser
 # chmod 755 /home/ftp/testuser

Тепер ми зможемо зайти на FTP-сервер з логіном testuser та паролем, який ми встановили для цього користувача.

Зверніть увагу на те, що ВСІ користувачі системи можуть заходити по ftp на наш сервер в свою домашню теку.
Для того, що б заборонити деяким користувачам логін по ftp служить файл */etc/ftpusers*
У ньому перераховані користувачі сервера, яким заборонений логін по FTP.

Моніторинг мережевих сервісів за допомогою Nagios

Система моніторингу — це система, яка виконує постійне спостереження за комп'ютерною мережею в пошуках повільних або несправних систем і яка при виявленні збоїв повідомляє про них системного адміністратора за допомогою пошти, SMS або інших засобів сповіщення.

Метою моніторингу серверів є:

- зниження виробничих витрат в сфері IT
- підвищення продуктивності роботи
- захист доходів клієнтів
- збереження клієнтської бази
- поліпшення якості життя системного адміністратора
- допомога в плануванні пропускної здатності мережі
- відповідність заявленому рівню обслуговування клієнтів

Nagios — це програма моніторингу комп'ютерних систем і мереж з відкритим кодом. Призначена для спостереження, контролю стану обчислювальних вузлів і служб, сповіщає адміністратора в тому випадку, якщо якісь із служб припиняють (або відновлюють) свою роботу.

Nagios спочатку була створена під ім'ям Netsaint, розроблена Етаном Галстадом. Він же підтримує і розвиває систему сьогодні, спільно з командою розробників, які займаються як офіційними, так і неофіційними плагінами.

Спочатку Nagios була розроблена для роботи під Linux, але вона також добре працює і під іншими ОС, такими як Sun Solaris, FreeBSD, AIX і HP-UX.

Можливості Nagios дуже широкі.

Ця система може:

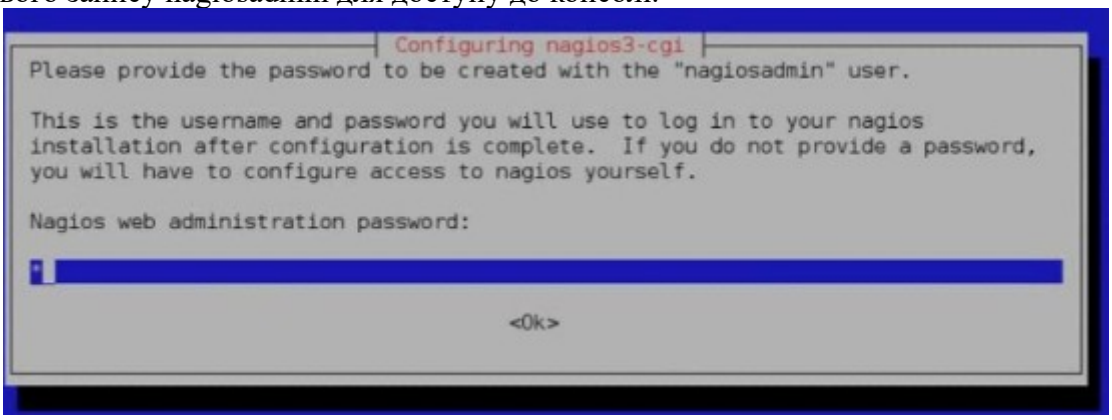
- Моніторити мережеві служби (SMTP, POP3, HTTP, NNTP, ICMP, SNMP).
- Моніторити стан хостів (завантаження процесора, використання диска, системні логи) в більшості мережевих операційних систем.
- Проста архітектура модулів розширень (плагінів) дозволяє, використовуючи будь-яку мову програмування за вибором (Shell, C + +, Perl, Python, PHP та інші), легко розробляти свої власні способи перевірки служб.
- Для повідомлень системному адміністратору підтримується відправлення повідомлення у разі виникнення проблем зі службою або хостом (за допомогою пошти, пейджера, SMS, або будь-яким іншим способом, визначеним користувачем через модуль системи).
- Так само є можливість організації спільної роботи декількох систем моніторингу з метою підвищення надійності та створення розподіленої системи моніторингу.

Встановлення Nagios

Встановлення сервера Nagios проводиться командою:
#apt install nagios3 nagios-plugins-extra

Зверніть увагу, що крім цих двох пакунків по залежностям встановлюється ще досить багато додаткових програм і сервісів.

В процесі налаштування сервера Nagios буде запропоновано ввести пароль для облікового запису nagiosadmin для доступу до консолі.



Після закінчення встановлення і налаштування всіх компонентів можна зайти на веб-інтерфейс Nagios за адресою http://server-ip/nagios3 і після переходу на закладку "Service Details" ми побачимо інформацію про два сервери, які додаються автоматично при установці — локального сервера, на який встановлено Nagios і сервера, що виконує роль інтернет-шлюзу.

Service Status Details For All Hosts

Host	Service	Status	Last Check	Duration	Attempt	Status Information
gateway	PING	OK	2012-02-12 11:21:41	0d 0h 6m 36s	1/4	PING OK - Packet loss = 0%, RTA = 0.96 ms
localhost	Current Load	OK	2012-02-12 11:23:07	0d 0h 5m 10s	1/4	OK - load average: 0.00, 0.13, 0.13
	Current Users	OK	2012-02-12 11:19:33	0d 0h 3m 44s	1/4	USERS OK - 2 users currently logged in
	Disk Space	OK	2012-02-12 11:20:58	0d 0h 2m 19s	1/4	DISK OK
	HTTP	OK	2012-02-12 11:22:24	0d 0h 5m 53s	1/4	HTTP OK: HTTP/1.1 200 OK - 453 bytes in 0.003 second response time
	SSH	OK	2012-02-12 11:18:50	0d 0h 4m 27s	1/4	SSH OK - OpenSSH_5.3p1 Debian-3ubuntu7 (protocol 2.0)
	Total Processes	OK	2012-02-12 11:20:16	0d 0h 3m 1s	1/4	PROCS OK: 94 processes

Додавання нового хоста в систему моніторингу

Для додавання нового хоста слід створити його файл конфігурації. Налаштуємо моніторинг сервера, наприклад, 192.168.0.30. Для цього створимо файл */etc/nagios3/conf.d/192.168.0.30.cfg* і запишемо в нього:

```
define host {
        host_name   my_server
        alias       my_server
        address     192.168.0.30
```

use generic-host
}

Крім цього необхідно описати які сервіси на сервері слід моніторити.
Опис того, які сервіси моніторяться на серверах знаходиться у файлі:
/etc/nagios3/conf.d/hostgroups_nagios2.cfg
Відкриємо цей файл і додамо наш сервер в групу ping-servers.
define hostgroup {
hostgroup_name ping-servers
alias Pingable servers
members gateway, my_server
}

Після цих змін слід дати команду сервісу Nagios перечитати конфігурацію:
#service nagios3 reload

Тепер, після оновлення веб-сторінки Nagios ми побачимо інформацію про наш сервер, який ми щойно додали в систему моніторингу.

Service Status Details For All Hosts

Host	Service	Status	Last Check	Duration	Attempt	Status Information
gateway	PING	OK	2012-02-12 11:51:41	0d 0h 35m 36s	1/4	PING OK - Packet loss = 0%, RTA = 0.70 ms
localhost	Current Load	OK	2012-02-12 11:48:07	0d 0h 34m 10s	1/4	OK - load average: 0.00, 0.01, 0.02
	Current Users	OK	2012-02-12 11:49:33	0d 0h 32m 44s	1/4	USERS OK - 2 users currently logged in
	Disk Space	OK	2012-02-12 11:50:58	0d 0h 31m 19s	1/4	DISK OK
	HTTP	OK	2012-02-12 11:47:24	0d 0h 34m 53s	1/4	HTTP OK: HTTP/1.1 200 OK - 453 bytes in 0.002 second response time
	SSH	OK	2012-02-12 11:48:50	0d 0h 33m 27s	1/4	SSH OK - OpenSSH_5.3p1 Debian-3ubuntu7 (protocol 2.0)
	Total Processes	OK	2012-02-12 11:50:16	0d 0h 32m 1s	1/4	PROCS OK: 96 processes
my_server	PING	OK	2012-02-12 11:50:51	0d 0h 11m 26s	1/4	PING OK - Packet loss = 0%, RTA = 0.70 ms

Додавання моніторингу нових сервісів

Опис усіх сервісів, які можуть моніторитися в мережі знаходяться у файлі */etc/nagios3/conf.d/services_nagios2.cfg*. За замовчуванням їх там всього три — HTTP, SSH та Ping. Звісно цього не достатньо для повноцінного моніторингу. Тому ми зараз розглянемо, як додати нові сервіси.

Для прикладу налаштуємо моніторинг поштового сервера, тобто будемо перевіряти те, що наш сервер відповідає на 25 порту по протоколу SMTP.

Відкриємо файл */etc/nagios3/conf.d/services_nagios2.cfg* і додамо в нього опис нового сервісу:

```
define service {
        hostgroup_name              smtp-servers
        service_description         smtp
        check_command               check_smtp
        use                           generic-service
        notification_interval       0 ; set > 0 if you want to be renotified
}
```

Далі необхідно включити перевірку даного сервісу на нашому сервері. Для цього у файлі */etc/nagios3/conf.d/hostgroups_nagios2.cfg* необхідно описати нову групу і включити в неї наш сервер:

```
define hostgroup {
        hostgroup_name  smtp-servers
        alias           SMTP servers
        members         my_server
        }
```

Після перезавантаження конфігурації Nagios в веб-інтерфейсі можна буде побачити результати моніторингу сервісу SMTP.

Service Status Details For All Hosts

Host	Service	Status	Last Check	Duration	Attempt	Status Information
gateway	PING	OK	2012-02-12 12:11:41	0d 0h 58m 48s	1/4	PING OK - Packet loss = 0%, RTA = 0.74 ms
localhost	Current Load	OK	2012-02-12 12:13:07	0d 0h 57m 22s	1/4	OK - load average: 0.03, 0.02, 0.00
	Current Users	OK	2012-02-12 12:14:33	0d 0h 55m 56s	1/4	USERS OK - 2 users currently logged in
	Disk Space	OK	2012-02-12 12:10:58	0d 0h 54m 31s	1/4	DISK OK
	HTTP	OK	2012-02-12 12:12:24	0d 0h 58m 5s	1/4	HTTP OK: HTTP/1.1 200 OK - 453 bytes in 0.003 second response time
	SSH	OK	2012-02-12 12:13:50	0d 0h 56m 39s	1/4	SSH OK - OpenSSH_5.3p1 Debian-3ubuntu7 (protocol 2.0)
	Total Processes	OK	2012-02-12 12:15:16	0d 0h 55m 13s	1/4	PROCS OK: 96 processes
my_server	PING	OK	2012-02-12 12:10:51	0d 0h 34m 38s	1/4	PING OK - Packet loss = 0%, RTA = 0.65 ms
	smtp	OK	2012-02-12 12:13:07	0d 0h 2m 22s	1/4	SMTP OK - 0.400 sec. response time

Для того, щоб з'ясувати як описується команда перевірки різних сервісів, варто подивитися вміст файлів в */etc/nagios-plugins/config*. Там вже є готові перевірки для великої кількості різноманітних сервісів.

Моніторинг серверів за протоколом SNMP

SNMP — це протокол керування мережами зв'язку на основі архітектури UDP.

Ця технологія покликана забезпечити управління і контроль за пристроями та програмами в мережі зв'язку шляхом обміну керуючою інформацією між агентами, що розташовуються на мережевих пристроях, і менеджерами, що розташовані на станціях

керування.

SNMP визначає мережу як сукупність мережевих керуючих станцій та елементів мережі (головні машини, шлюзи і маршрутизатори, термінальні сервери), які спільно забезпечують адміністративні зв'язки між мережевими керуючими станціями та мережевими агентами.

SNMP не визначає, яку інформацію керована система повинна надавати. Навпаки, SNMP використовує розширювану модель, в якій доступна інформація визначається Базами Керуючої Інформації (MIB). Бази Керуючої Інформації описують структуру керуючої інформації пристроїв. Вони використовують ієрархічний простір імен, що містить унікальний ідентифікатор об'єкта (OID). Грубо кажучи, кожен унікальний ідентифікатор об'єкта ідентифікує змінну, яка може бути прочитана чи встановлена через SNMP.

Ієрархія MIB може бути зображена як дерево з безіменним коренем, рівні якого присвоєні різними організаціями. На найвищому рівні MIB OIDи належать різним організаціям, що займаються стандартизацією, в той час як на більш низькому рівні OIDи виділяються асоційованими організаціями. Ця модель забезпечує управління на всіх рівнях мережевої моделі OSI, так як MIBи можуть бути визначені для будь-яких типів даних і операцій.

Встановлення SNMP

Для того, щоб тестовий сервер можна було моніторити по SNMP необхідно на нього встановити два пакети:

#apt install snmp snmpd

Тепер потрібно налаштувати наш сервіс SNMP. Для цього в файл */etc/snmp/snmpd.conf* запишемо:

```
syslocation: test, Ukraine
syscontact     admin@yakim.org.ua
#community                  #hosts allowed
rwcommunity    private      192.168.0.30/32
rocommunity    public       192.168.0.0/24
disk /
disk /home
```

Зверніть увагу, ми описуємо підмережі з яких дозволений доступ в різних режимах (читання або читання і запис), а також вказуємо пароль для цього доступу (у нас це private і public). Звісно в реальних умовах пароль не повинен бути настільки простим.

Крім цього необхідно внести невеликі зміни в файл */etc/default/snmpd*. У ньому параметр SNMPDOPTS потрібно привести до наступного вигляду:

SNMPDOPTS='-Lsd -Lf /dev/null -u snmp -g snmp -I -smux -p /var/run/snmpd.pid -c /etc/snmp/snmpd.conf'

Тобто ми явно прописуємо в ньому файл конфігурації демона SNMP. Тепер потрібно перезапустити сервіс:

#service snmpd restart

Конфігурування Nagios для роботи з SNMP

Створимо правило моніторингу завантаження процесора.

За замовчуванням Nagios вміє тільки показувати завантаження і не має параметра

критичного порога завантаження. Для того, щоб це виправити відкриємо файл */etc/nagios-plugins/config/snmp.cfg*, знайдемо в ньому блок опису snmp_cpustats та змінимо параметр command_line. В кінці цього рядка допишемо, що другим параметром приймається критичне значення завантаження:

command_line /usr/lib/nagios/plugins/check_snmp -H '$HOSTADDRESS$' -C '$ARG1$' -o .1.3.6.1.4.1.2021.11.9.0,.1.3.6.1.4.1.2021.11.10.0,.1.3.6.1.4.1.2021.11.11.0 -l 'CPU usage (user system idle)' -u '%' -c '$ARG2$'

Створимо групу моніторингу процесора по snmp. Для цього додамо опис сервісу в файл */etc/nagios3/conf.d/services_nagios2.cfg:*

```
define service {
        hostgroup_name              cpustats
        service_description         cpustats
        check_command               snmp_cpustats!public!1
        use                     generic-service
        notification_interval       0 ; set > 0 if you want to be renotified
}
```

У цьому описі перевірки в рядку check_command вказуємо пароль доступу до сервісу SNMP і критичне значення у відсотках, що повертається перевіркою.

Далі додамо опис групи моніторингу в файл */etc/nagios3/conf.d/hostgroups_nagios2.cfg*

```
define hostgroup {
        hostgroup_name  cpustats
        alias           cpustats
        members         my_server
        }
```

Тепер знову перечитуємо конфігурацію Nagios.

Service Status Details For All Hosts

Host	Service	Status	Last Check	Duration	Attempt	Status Information
gateway	PING	OK	2012-02-12 13:56:41	0d 2h 40m 54s	1/4	PING OK - Packet loss = 0%, RTA = 0.65 ms
localhost	Current Load	OK	2012-02-12 13:53:07	0d 2h 39m 28s	1/4	OK - load average: 0.02, 0.02, 0.00
	Current Users	OK	2012-02-12 13:54:33	0d 2h 38m 2s	1/4	USERS OK - 2 users currently logged in
	Disk Space	OK	2012-02-12 13:55:58	0d 2h 36m 37s	1/4	DISK OK
	HTTP	OK	2012-02-12 13:53:58	0d 2h 40m 11s	1/4	HTTP OK: HTTP/1.1 200 OK - 453 bytes in 0.003 second response time
	SSH	OK	2012-02-12 13:53:50	0d 2h 38m 45s	1/4	SSH OK - OpenSSH_5.3p1 Debian-3ubuntu7 (protocol 2.0)
	Total Processes	OK	2012-02-12 13:56:59	0d 2h 37m 19s	1/4	PROCS OK: 96 processes
my_server	PING	OK	2012-02-12 13:57:21	0d 2h 16m 44s	1/4	PING OK - Packet loss = 0%, RTA = 1.71 ms
	cpustats	CRITICAL	2012-02-12 13:56:51	0d 0h 13m 44s	4/4	CPU usage (user system idle) CRITICAL - *8* % 9 81

В якості критичного порога завантаження ми вказали 1% і бачимо, що значення завантаження показується правильно, тому що завантаження процесора на сервері, що перевіряється досягає 8%.

Ще додамо перевірку вільного місця на диску.

Створимо групу моніторингу дискового простору по snmp. Для цього додамо опис

сервісу в файл */etc/nagios3/conf.d/services_nagios2.cfg*

```
define service {
        hostgroup_name                  snmp disk
        service_description             snmpdisk
        check_command                   snmp_disk!public!1!!50!!90
        use                             generic-service
        notification_interval           0 ; set > 0 if you want to be renotified
}
```

У цьому описі перевірки в рядку *check_command* вказуємо пароль доступу до сервісу SNMP, номер диска, який ми перевіряємо, і розмір вільного місця у відсотках для параметрів warning і critical.

Далі додамо опис групи моніторингу в файл */etc/nagios3/conf.d/hostgroups_nagios2.cfg:*

```
define hostgroup {
        hostgroup_name  snmp disk
    alias           SNMP disk
      members         my_server
        }
```

Service Status Details For All Hosts

Host	Service	Status	Last Check	Duration	Attempt	Status Information
gateway	PING	OK	2012-02-12 14:16:41	0d 3h 2m 31s	1/4	PING OK - Packet loss = 0%, RTA = 0.68 ms
localhost	Current Load	OK	2012-02-12 14:18:07	0d 3h 1m 5s	1/4	OK - load average: 0.00, 0.02, 0.00
	Current Users	OK	2012-02-12 14:14:33	0d 2h 59m 39s	1/4	USERS OK - 2 users currently logged in
	Disk Space	OK	2012-02-12 14:15:58	0d 2h 58m 14s	1/4	DISK OK
	HTTP	OK	2012-02-12 14:18:58	0d 3h 1m 48s	1/4	HTTP OK: HTTP/1.1 200 OK - 453 bytes in 0.003 second response time
	SSH	OK	2012-02-12 14:18:50	0d 3h 0m 22s	1/4	SSH OK - OpenSSH_5.3p1 Debian-3ubuntu7 (protocol 2.0)
	Total Processes	OK	2012-02-12 14:16:59	0d 2h 58m 56s	1/4	PROCS OK: 96 processes
my_server	PING	OK	2012-02-12 14:15:46	0d 2h 38m 21s	1/4	PING OK - Packet loss = 0%, RTA = 0.47 ms
	cpustats	CRITICAL	2012-02-12 14:16:51	0d 0h 35m 21s	4/4	CPU usage (user system idle) CRITICAL - *7* % 9 81
	snmpdisk	WARNING	2012-02-12 14:14:34	0d 0h 7m 38s	4/4	disk space WARNING - 4922448 kB free (*73* % used)

Тепер знову перечитуємо конфігурацію Nagios.

Як ми бачимо перевірка місця на диску працює і повертає правильні результати. В даному випадку ми отримали попередження.

Аналогічним чином можна контролювати будь-який сервіс і пристрій, які вміють віддавати про себе інформацію по SNMP.

Сервер централізованого збирання логів з серверів Windows та Linux на базі rsyslog та LogAnalizer

У багатьох фірмах, особливо великого розміру, виникає завдання централізованого збору та аналізу логів. Для цього існує багато різних продуктів — Quest, Splunk і т. д. Можливості у них, звичайно, досить великі. Але, за те, і ціна не маленька. Та і продаються вони не маленькими пакетами. Наприклад Quest — від 200 ліцензій. А що робити, якщо така кількість не потрібно? Купувати зайве не хочеться ж. От і приходять на допомогу можливості OpenSource.

У Linux є чудовий демон — rsyslog. Він може збирати дані не тільки локально, але й по мережі. Було б кому віддавати ці дані. Але, у зв'язку з тим, що сьогодні віддавати логи на rsyslog вміють багато хто (і Windows в тому числі), саме на ньому ми й розглянемо встановлення та налаштування сервера збору логів.

На нашому Linux-сервері встановимо в якості операційної системи Ubuntu Linux 10.04. Rsyslog на ньому вже є. Залишилося це все налаштувати.

Для того, щоб ці логи можна було зручно дивитися і аналізувати будемо зберігати їх в базі даних — MySQL. Відповідно встановимо базу даних:

$sudo apt install mysql-server mysql-client.

При налаштуванні введемо пароль root для MySQL і постараємося його не забути — він нам ще знадобиться.

Тепер навчимо rsyslog зберігати свої дані (тобто всі логи) в MySQL. Тля цього встановимо syslog-mysql:

$sudo apt install rsyslog-mysql

В процесі налаштування в MySQL буде створена база Syslog і буде потрібно ввести пароль для користувача rsyslog.

Цей пароль теж не забуваємо.

Далі ми навчимо наш сервер приймати логи по мережі. Для цього редагуємо файл */etc/rsyslog.conf*.

Розкоментуємо в ньому такі рядки:

```
# provides UDP syslog reception
module(load="imudp")
#nput(type="imudp" port="514")

# provides TCP syslog reception
module(load="imtcp")
input(type="imtcp" port="514")
```

І, звісно, перезавантажимо сервіс rsyslog:

$sudo service rsyslog restart

У випадку, якщо потік логів буде занадто великим, деякі події можуть загубитися, тому що просто їх не встигнуть обробити. Що б такого не сталося налаштуємо буферизацію повідомлень.

Створимо теку, в якому rsyslog буде зберігати чергу повідомлень:

$ sudo mkdir -p /var/rsyslog/work

І додамо в */etc/rsyslog.d/mysql.conf* такі рядки:
 # Buffering stuff:
$WorkDirectory /var/rsyslog/work # default location for work (spool) files
$ActionQueueType LinkedList # use asynchronous processing
$ActionQueueFileName dbq # set file name, also enables disk mode
$ActionResumeRetryCount -1 # infinite retries on insert failure

І знову перезапустимо наш демон:
$sudo service rsyslog restart

Ось на цьому, власне, настройка сервера збирання логів закінчена. Якби не одне «АЛЕ». Хотілося б якось візуалізувати звітність. Не всі хочуть і можуть читати логи в консолі. Ось тут нам на допомогу приходить LogAnalizer — веб-інтерфейс для роботи з логами.

Налаштування LogAnalizer

Для роботи LogAnalizer нам знадобиться веб-сервер, php та MySQL. Так як MySQL мі вже встановили — довстановимо відсутні компоненти:
$sudo apt install apache2 php php-mysql php-gd

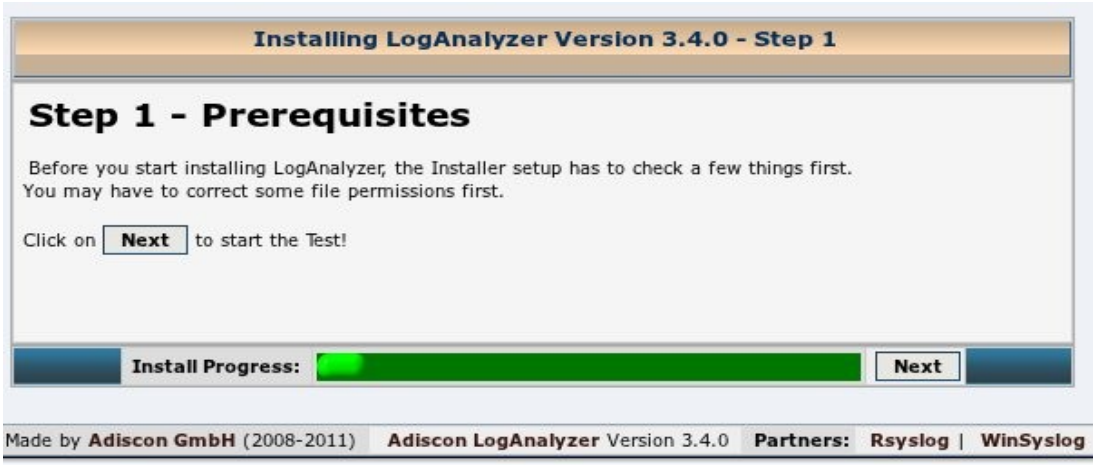
Завантажимо останню версію LogAnalizer:
$wget http://download.adiscon.com/loganalyzer/loganalyzer-4.1.7.tar.gz
Розпакуємо його:
$tar -xzf loganalyzer-4.1.7.tar.gz

Та перейдемо в його теку

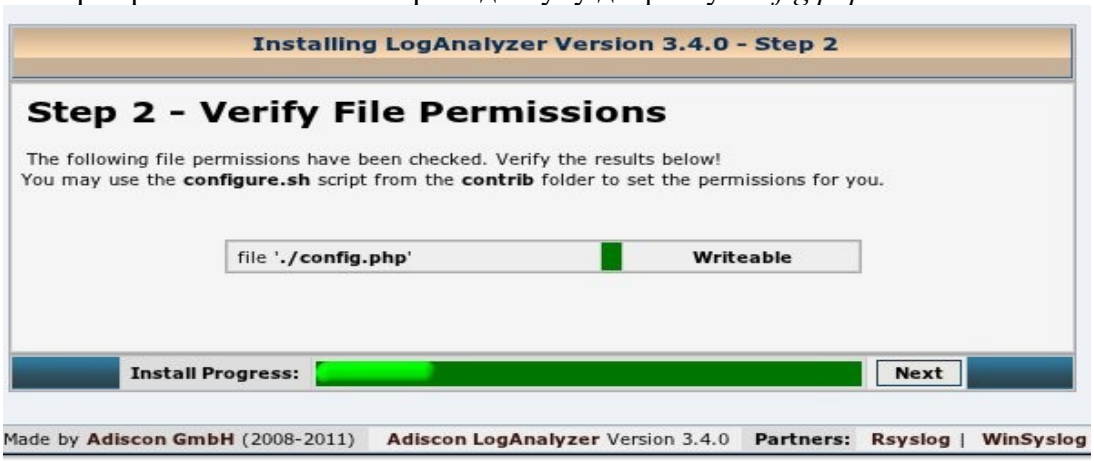
$cd loganalyzer- 4.1.7

Тепер встановимо його в теку на нашому веб-сервері:
$sudo mkdir /var/www/logs
$sudo cp -R src/ /var/www/logs/*
$sudo cp contrib/ /var/www/logs/*
$cd /var/www/logs/
$sudo chmod +x configure.sh secure.sh
$sudo ./configure.sh

Тепер створимо базу даних для роботи LogAnalizer:
$mysql -u root -p
mysql> create database LogAnalyzerUsers;
mysql> show databases;
І створимо користувача la з паролем la для доступу до цієї бази:
mysql> grant all on LogAnalyzerUsers. to la@'localhost' identified by 'la';*
mysql> quit
Для продовження відкриємо веб-інтерфейс LogAnalizer'a:
http://ourserver/logs
І починаємо встановлення:



Далі перевіряється наявність і права доступу до файлу *config.php*



Для того, щоб була можливість створювати користувачів всередині LogAnalizer'a і роздавати їм права доступу, вибираємо Enable User Database — yes

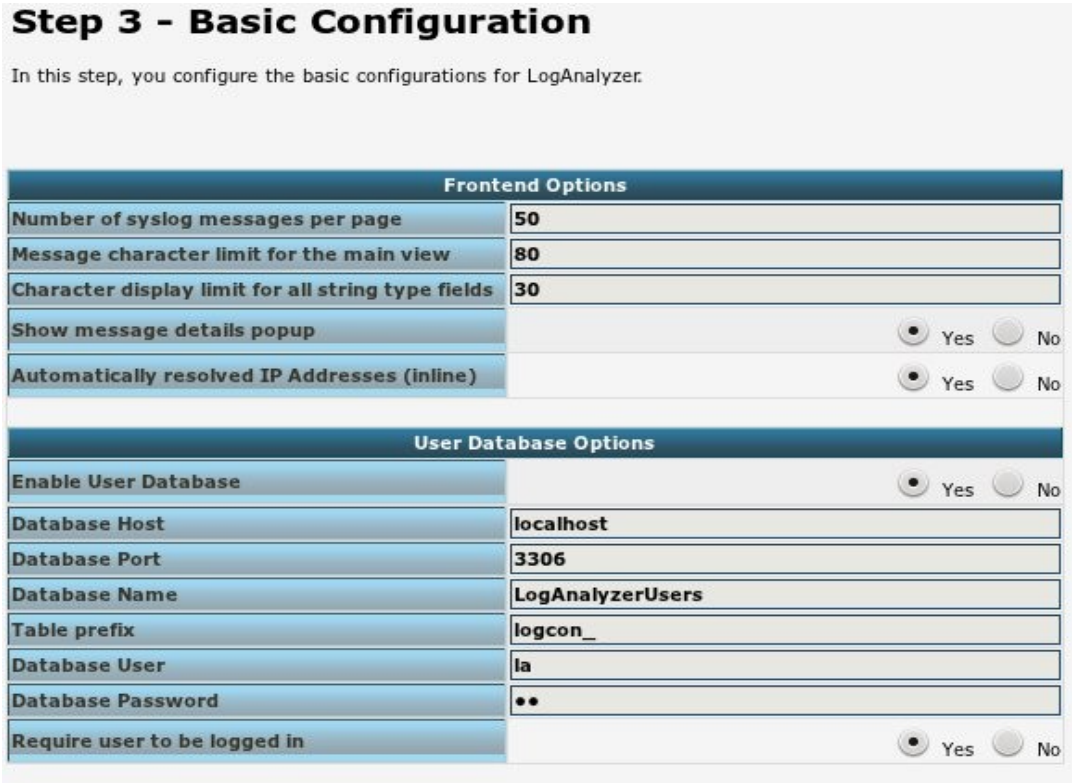
Заповнюємо логін, пароль та ім'я бази, як ми їх створили. Тобто в моєму випадку:

назва бази LogAnalizerUsers

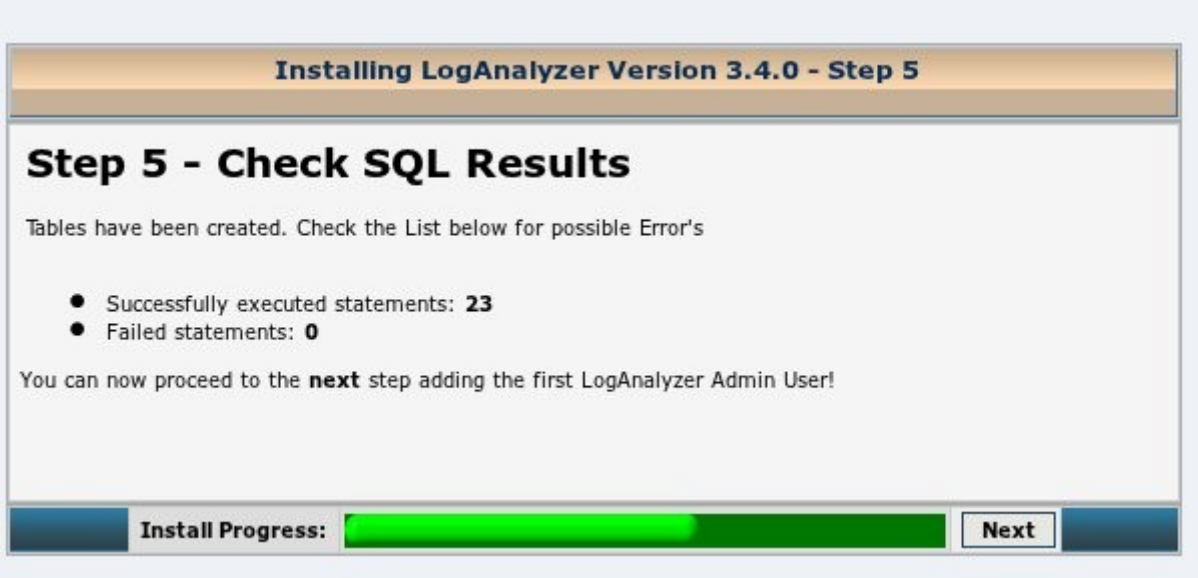
логін — la

пароль — la

Будьте уважні з регістром символів! Він має значення!



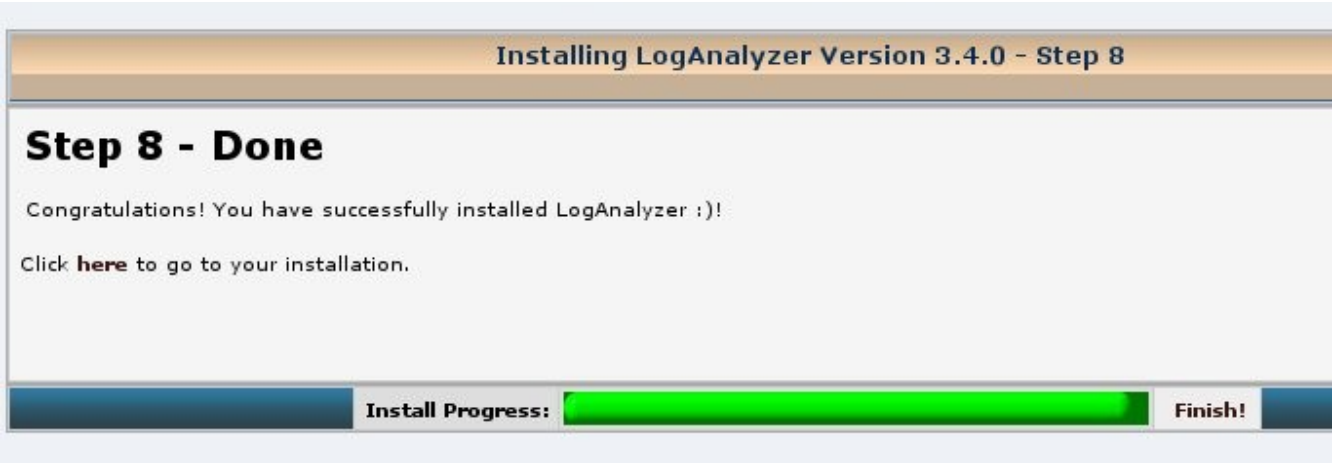
У наступних двох кроках перевіряється з'єднання з базою даних і в ній створюються відповідні таблиці.



Далі створюється адміністративний обліковий запис (я його назвав admin)

Тепер ми описуємо параметри з'єднання з базою rsyslog.
Вводимо всі параметри саме так, як показано на малюнку.
Знову не забуваємо про те, що регістр символів має велике значення.

И, нарешті, все. Налаштування завершене.



Тиснемо Finish та дивимось логи через веб-інтерфейс.

Якщо щось налаштували невірно, нічого страшного. Завжди можна провести налаштування заново.

Для цього в теці */var/www/logs* видаляємо файл *config.php* і потім в ньому ж запускаємо скрипт *configure.sh*

Після цього відкриваємо знову в браузері *http://ourserver/logs* та конфігуруємо все спочатку.

rsyslog та Windows

Є можливість передавати в rsyslog журнали Windows-серверів і робочих станцій.

Для цього завантажуємо eventlog-to-syslog, (http://code.google.com/p/eventlog-to-syslog/) і копіюємо evtsys.exe та evtsys.dll в *\%WinDir%\system32*

Після чого запускаємо:

C:\Windows\system32>evtsys.exe -i -h ip.address.of.myserv

Після перезавантаження (або ручного запуску сервісу evtsys) всі журнали Windows будуть передаватися в rsyslog нашого сервера і, відповідно, будуть доступні у веб-інтерфейсі LogAnalizer.

Мови перекладу

Інтерфейс цього продукту поки є тільки англійською, німецькою та українською мовами.

Резервне копіювання серверів та робочих станцій в офісній мережі

Всі системні адміністратори поділяються на дві категорії — ті хто не робить резервне копіювання і ті, хто вже робить.

Необхідність резервного копіювання зрозуміла практично всім адміністраторам. Але, на жаль, вельми часто трапляється так, що не виходить пояснити необхідність закупівлі комерційного програмного забезпечення для цих цілей.

Звичайно існує велика кількість комерційних рішень для створення резервних копій чого завгодно: як окремих файлів, так і конкретно поштових баз, структури AD або навіть всього сервера відразу. Так само є чимало і вільних і безкоштовних рішень для цього. Наприклад Bacula, Amanda, dar, BackUpPC та інші.

Найбільш потужним і універсальним рішенням є, звичайно, Bacula. Але вона ж і найбільш складна в налаштуваннях. Для більшості ситуацій буде достатньо використовувати BackUpPC

BackupPC - це вільне ПЗ (поширюється під GNU General Public License) для резервного копіювання даних з керуванням через веб-інтерфейс. Багатоплатформовий програмний сервер може працювати на будь-якому сервері під керуванням GNU / Linux, Solaris або UNIX. Немає необхідності в клієнтській частині, так як сервер сам по собі є клієнтом для декількох протоколів, які підтримуються рідними службами клієнтської ОС.

Наприклад, BackupPC є SMB-клієнтом, що може використовуватися для резервного копіювання спільно використовуваних даних в мережевих папках на комп'ютерах з Microsoft Windows. Подібний сервер BackupPC може бути встановлений за фаєрволом, який виконує функції мережевої трансляції адрес (NAT), коли Windows-комп'ютер має публічну IP адресу. Так як це не рекомендується через велику кількість SMB трафіку, то більш зручним є використання веб-серверів, що підтримують SSH та можливість роботи з tar та rsync, що дозволяє серверу BackupPC перебувати в підмережі, що відокремлена від веб-серверів демілітаризованою зоною.

Встановлення системи резервного копіювання BackUpPC

Так як керування BackupPC проводиться через веб-інтерфейс, то спочатку необхідно встановити веб-сервер:

 # apt install apache2

Далі встановимо сам сервер резервного копіювання та пакет smbclient для роботи з windows-комп'ютерами:

 # apt install backuppc smbclient

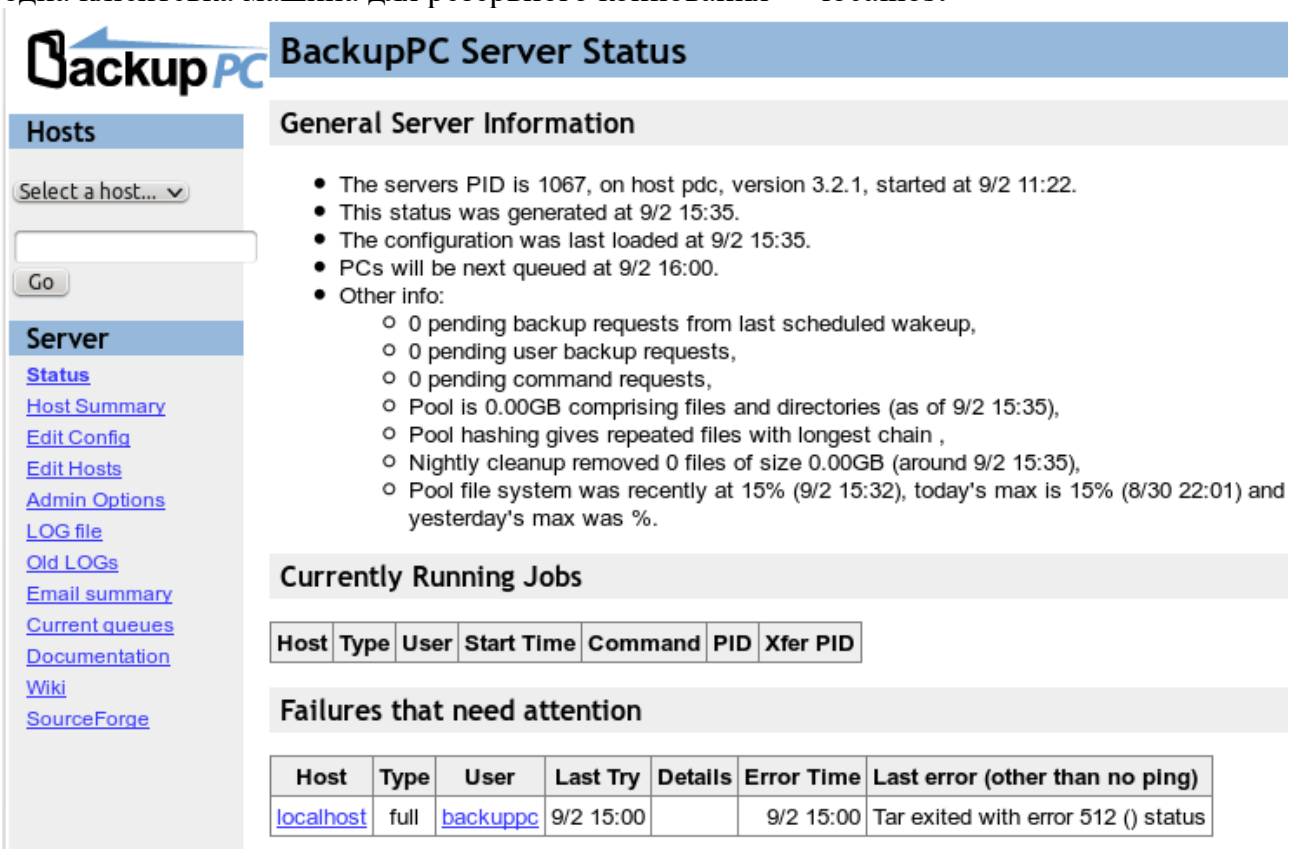
Після закінчення встановлення його веб-інтерфейс буде доступний за адресою *http://ip-addr/backuppc*

Для входу в систему будемо використовувати логін backuppc та пароль password.

Для того, щоб змінити пароль для цього користувача слід ввести команду:
 # htpasswd /etc/backuppc/htpasswd backuppc

Додати нового користувача (наприклад admin) та задати йому пароль можна командою:
 # htpasswd /etc/backuppc/htpasswd admin

Після першого логіна в систему ми побачимо вікно, в якому прописана тільки одна клієнтська машина для резервного копіювання — localhost



Файли та шляхи, що використовуються в BackupPC

Основний файл налаштувань даної системи резервного копіювання — */etc/backuppc/config.pl*

В цій же теці знаходиться файл *hosts*, в якому прописані всі клієнтські комп'ютери з яких буде проводиться резервне копіювання.

Також в цій теці повинні знаходитися файли з розширенням pl, які називаються за іменем клієнтських комп'ютерів. Саме в них і будуть зберігатися всі індивідуальні налаштування резервного копіювання.

В теці */var/lib/backuppc/pc* будуть зберігатися резервні копії всіх серверів. Тому необхідно або передбачити достатню кількість дискового простору по цьому шляху, або перевизначити його в файлі */etc/backuppc/config.pl*

Ще один важливий шлях — */usr/share/backuppc/lib/BackupPC/Lang*

Саме там зберігаються всі файли локалізації інтерфейсу. Переклад є на українську, англійську, чеську, іспанську і деякі інші мови.

Конфігурування клієнтського Linux-хоста

Додаємо в кінець файлу */etc/backuppc/hosts* рядок:

linuxsrv1 0 backuppc

Перше поле, це зрозуміле нам ім'я хоста, друге — включення (1) або відключення (0) dhcp, і третє — користувач, який має доступ до керування даним хостом.

Далі створимо файл конфігурації цього сервера:

#nano /etc/backuppc/linuxsrv1.pl

Та заповнимо його наступним чином:

```
$Conf{BackupsDisable} = '1';
$Conf{ClientNameAlias} = '192.168.0.27';
$Conf{BackupFilesOnly} = {
  '*' => [
    '/etc'
  ]
};

$Conf{XferMethod} = 'rsync';
$Conf{RsyncClientPath} = '/usr/bin/rsync';
$Conf{RsyncClientCmd} = '$sshPath -q -x -l bcuser $host sudo $rsyncPath $argList+';
$Conf{RsyncClientRestoreCmd} = '$sshPath -q -x -l bcuser $host sudo $rsyncPath $argList+';
$Conf{RsyncShareName} = [
  '/'
];
$Conf{RsyncCsumCacheVerifyProb} = '0.01';
$Conf{XferLogLevel} = '9';
$Conf{ClientCharsetLegacy} = 'utf8';
$Conf{BackupFilesExclude} = {
  '/' => [
```

'/etc/ssh'

]

};

В цьому файлі ми вказуємо адресу клієнтського сервера
$Conf{ClientNameAlias} = '192.168.0.27';

Шлях копіювання

$Conf{BackupFilesOnly} = { '' => ['/etc']};*

Шлях виключень

$Conf{BackupFilesExclude} = { '/' => ['/etc/ssh']};

Та тип з'єднання

$Conf{XferMethod} = 'rsync';

Далі потрібно зробити можливість нашому сервера резервного копіювання авторизуватися на віддаленій системі по протоколу ssh не за паролем, а за ключем.

Створюємо ключ для користувача backuppc, з правами якого і працює наш сервер резервного копіювання:

Переключаємося в контекст користувача *backuppc*
 # su backuppc

$ ssh-keygen -t rsa

Запитаний пароль на ключ залишаємо порожнім.

Далі на клієнтському сервері створюємо користувача *bcuser* та в налаштуваннях sudo (#visudo) прописуємо:
bcuser ALL=(ALL) NOPASSWD:/usr/bin/rsync

Тобто цього користувачеві ми даємо можливість без пароля запускати rsync з підвищеними правами.

Тепер повертаємося на наш сервер резервного копіювання. Ключ для авторизації по ssh ми вже створили і тепер потрібно передати його на клієнтський сервер.

$ ssh-copy-id -i /var/lib/backuppc/.ssh/id_rsa.pub bcuser@192.168.0.27

де 192.168.0.27 — IP-адреса клієнтського Linux-сервера

Намагаємося з'єднатися з авторизацією по ключу:
$ ssh bcuser@192.168.0.27

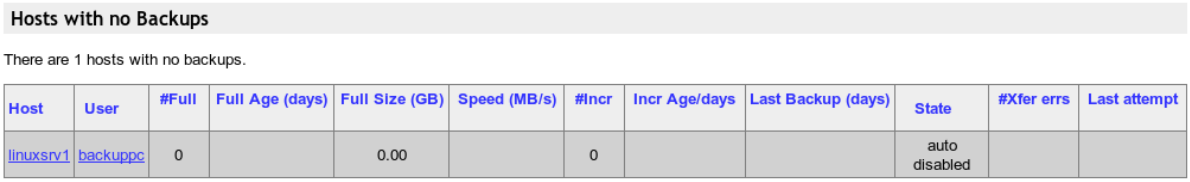
Якщо з'єднання пройшло успішно, значить на цьому налаштування завершене.

Виходимо з контексту користувача backuppc

$ exit

Перезапускаємо наш сервер резервного копіювання:
#service backuppc restart

Відкривши його веб-інтерфейс і перейшовши на закладку «Host Summary» в списку клієнтських серверів ми побачимо наш тільки що описаний сервер linuxsrv1



Hosts with no Backups

There are 1 hosts with no backups.

Host	User	#Full	Full Age (days)	Full Size (GB)	Speed (MB/s)	#Incr	Incr Age/days	Last Backup (days)	State	#Xfer errs	Last attempt
linuxsrv1	backuppc	0		0.00		0			auto disabled		

Зайдемо всередину цього хоста і натиснемо кнопку «Start Full Backup»

Host linuxsrv1 Backup Summary

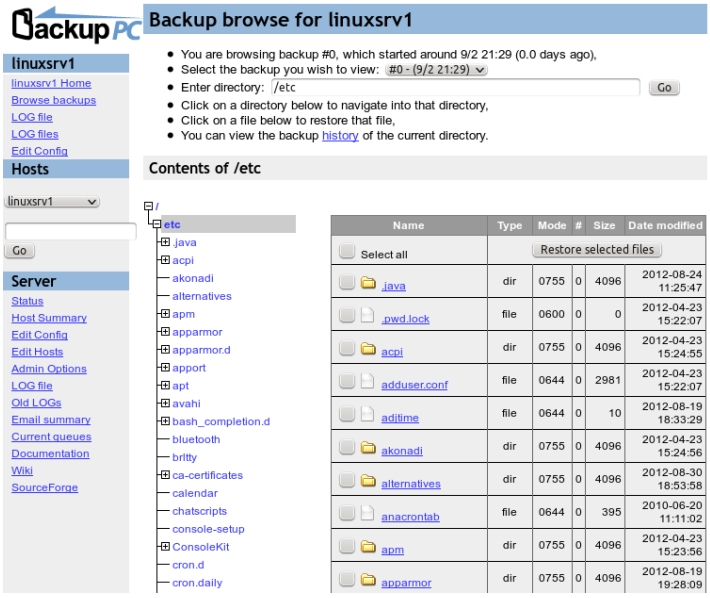
This PC has never been backed up!!

- This PC is used by backuppc.
- Last status is state "idle" (idle) as of 9/2 21:00.

User Actions

Start Full Backup | Stop/Dequeue Backup

Через деякий (в нашому випадку близько 15-20 секунд) час, перейшовши на закладку «Browse backups» Ми зможемо побачити все дерево резервної копії з підлеглого сервера.



BackupPC — **Backup browse for linuxsrv1**

linuxsrv1 | linuxsrv1 Home | Browse backups | LOG file | LOG files | Edit Config

Hosts: linuxsrv1 — Go

Server: Status | Host Summary | Edit Config | Edit Hosts | Admin Options | LOG file | Old LOGs | Email summary | Current queues | Documentation | Wiki | SourceForge

- You are browsing backup #0, which started around 9/2 21:29 (0.0 days ago),
- Select the backup you wish to view: #0 - (9/2 21:29)
- Enter directory: /etc — Go
- Click on a directory below to navigate into that directory,
- Click on a file below to restore that file,
- You can view the backup history of the current directory.

Contents of /etc

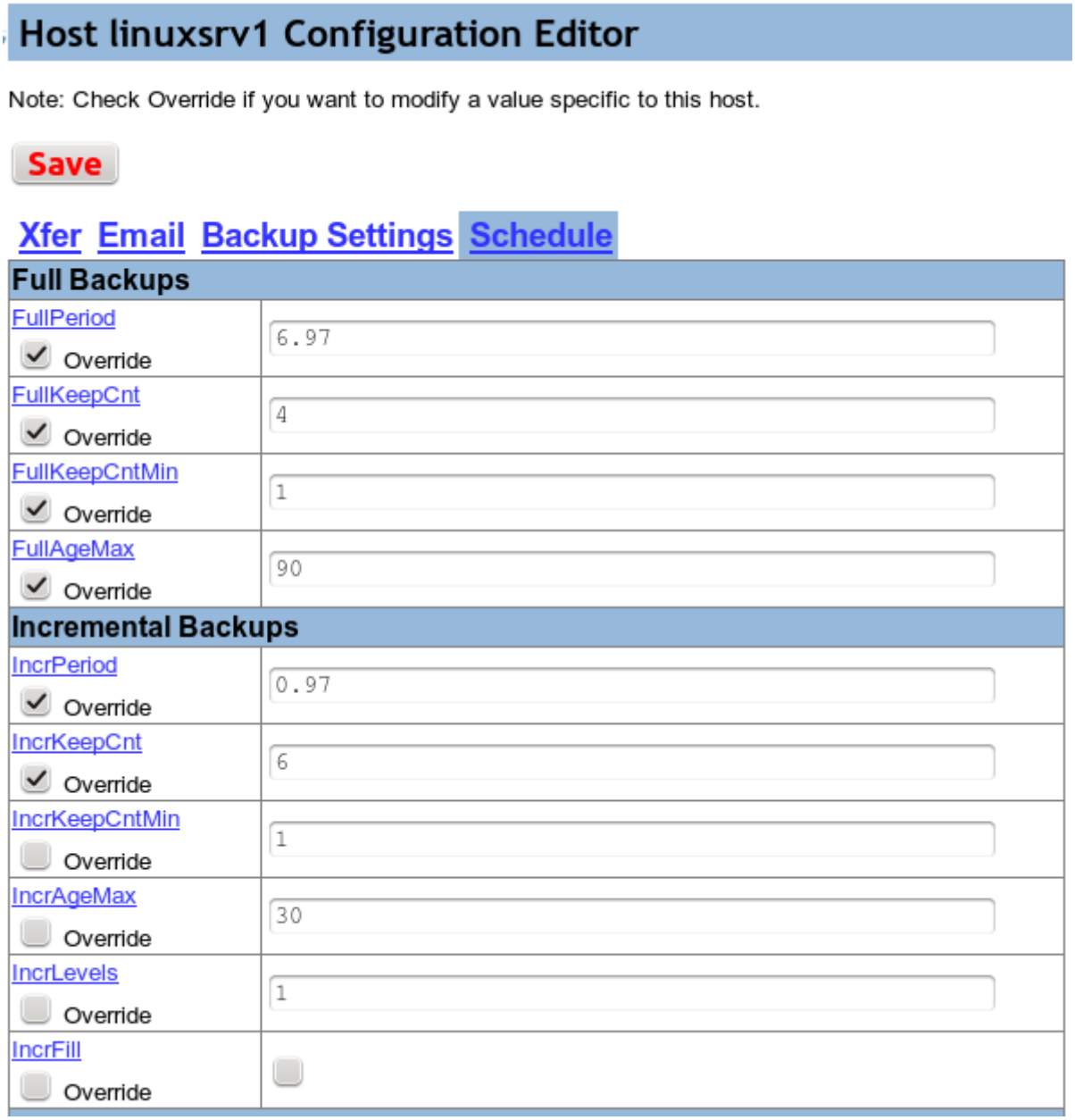
/
- etc
  - java
  - acpi
  - akonadi
  - alternatives
  - apm
  - apparmor
  - apparmor.d
  - apport
  - apt
  - avahi
  - bash_completion.d
  - bluetooth
  - brltty
  - ca-certificates
  - calendar
  - chatscripts
  - console-setup
  - ConsoleKit
  - cron.d
  - cron.daily

Name	Type	Mode	#	Size	Date modified
Select all	Restore selected files				
java	dir	0755	0	4096	2012-08-24 11:25:47
.pwd.lock	file	0600	0	0	2012-04-23 15:22:07
acpi	dir	0755	0	4096	2012-04-23 15:24:55
adduser.conf	file	0644	0	2981	2012-04-23 15:22:07
aditime	file	0644	0	10	2012-08-19 18:33:29
akonadi	dir	0755	0	4096	2012-04-23 15:24:56
alternatives	dir	0755	0	4096	2012-08-30 18:53:58
anacrontab	file	0644	0	395	2010-06-20 11:11:02
apm	dir	0755	0	4096	2012-04-23 15:23:56
apparmor	dir	0755	0	4096	2012-08-19 19:28:09

Якщо уважно подивитися на список вкладених тек, то ми побачимо, що теки */etc/ssh* там немає, що говорить про те, що списки виключення працюють коректно.

Створення розкладу автоматичного копіювання.

Налаштування розкладу резервного копіювання проводиться у властивостях конкретного хоста на вкладці «Shedule»



Тут є такі поля налаштувань:
Повна резервна копія:
FullPeriod — мінімальний час в днях між повними бекапами

FullKeepCnt — скільки повних резервних копій необхідно зберігати

FullKeepCntMin — мінімальна кількість збережених повних бекапів

FullAgeMax — максимальний вік зберігання повного бекапа

Інкрементальна резервна копія:

IncrPeriod — мінімальний час в днях між інкрементальними бекапами

IncrKeepCnt — скільки інкрементальних резервних копій необхідно зберігати

IncrKeepCntMin — мінімальна кількість збережених інкрементальних резервних копій

IncrAgeMax — максимальний вік зберігається інкрементального бекапа

IncrLevels — рівень інкрементального бекапа

IncrFill — використовувати в системі хард-лінки, щоб інкрементальний бекап виглядав повним.

Епілог

Сподіваюся, ця книга допомогла Вам в освоєнні прекрасної і багатофункціональної операційної системи Linux. Звісно в неї не ввійшли, та й не могли увійти всі питання, які можуть з'явитися перед Вами, як перед системними адміністраторами, але я сподіваюся, що налаштування найбільш часто вживаних сервісів для Вас стане простішим і зрозумілішим.

Додаток. Список найбільш часто вживаних команд.

Робота з файловою системою

Перейти в теку /home/yakim/test
cd /home/yakim/test

Піднятися в теку рівнем вище
cd ..

Повернутися в попередню теку
cd -

Створити теку /home/yakim/newdir
mkdir /home/yakim/newdir

Створити все дерево тек /home/yakim/newdir/test1/test2
mkdir -p /home/yakim/newdir/test1/test2

Видалити теку
rmdir /home/yakim/newdir/test1/test2

Створити порожній файл /home/yakim/newdir/test.txt
touch /home/yakim/newdir/test.txt

Видалити файл
rm /home/yakim/newdir/test.txt

Відкрити файл home/yakim/newdir/test.txt на редагування
nano /home/yakim/newdir/test.txt

Продивитися вміст файлу
cat test.txt

Посторінково переглянути вміст файлу
less test.txt
cat test.txt | more

Переглянути кінець файлу
tail /var/log/mail.log

Виводити останні рядки файлу на консоль в міру їх появи (дуже зручно переглядати лог в процесі пошуку помилок)
tail -f /var/log/mail.log

Копіювати файл test.txt в test2.txt
cp test.txt test2.txt

Дисковий простір

Переглянути інформацію про розділи і місце на них
df -h

Переглянути вміст теки
ls /home/yakim/newdir

Переглянути вміст теки з подробицями
ls -la /home/yakim/newdir

Підрахувати розмір теки
du -sh dir1

Права доступу

Встановити права доступу на файл або теку
chmod 770 /home/yakim/newdir

Змінити власника файлу або теки
chown yakim:pub /home/yakim/newdir

Встановити на файл атрибут виконуваності
chmod +x test.sh

Встановити на теку SGID-біт
chmod g+s /home/yakim/newdir

Переглянути додаткові атрибути
lsattr /home/yakim/newdir

Встановити додатковий атрибут невидаляємості.
chattr +i test.sh

Робота з архівами

Розпакувати файл архіву tar.gz
tar xzf test.tar.gz

Стиснути два файли в архів
tar czf test.tar.gz file1 file2

Зібрати без стиснення файли і теки в один файл
tar cf test.tar file1 directory1

Розпакувати файл архіву tar.bz2
tar xjf test.tar.bz2

Стиснути всі файли і теки в поточній теці в архів .tar.bz2
*tar cjf test.tar.bz2 **

Створити архів rar з файлу test.txt
rar a test.rar test.txt

Розпакувати test.rar
unrar x test.rar

Створити zip-архів
zip test.zip test.txt

Розпакувати zip-архів
unzip test.zip

Користувачі та групи

Створити користувача user
adduser user

Створити групу newgroup
addgroup newgroup

Додати користувача user в групу newgroup
addgroup user newgroup

Змінити свій пароль
passwd

Змінити пароль користувача user (виконується з правами суперкористувача)
passwd user

Робота з системою

Показати інформацію про ядро
uname -a

Список завантажених модулів ядра
lsmod

Завантажити модуль modulename
modprobe modulename

Вивантажити модуль modulename
rmmod modulename

Показати інформацію про процесор
lscpu
cat /proc/cpuinfo

Показати інформацію про пам'ять
cat /proc/meminfo

Показати інформацію про доступну пам'ять та swap
free -m

Переглянути інформацію про PCI-пристрої
lspci

Переглянути інформацію про USB-пристрої
lsusb

Налаштування мережі

Показати налаштування всіх інтерфейсів
ifconfig

Показати налаштування інтерфейсу eth0
ifconfig eth0

Встановити налаштування основної адреси інтерфейсу eth0
ifconfig eth0 192.168.50.254 netmask 255.255.255.0

Встановити налаштування додаткової адреси інтерфейсу eth0
ifconfig eth0:0 192.168.51.254 netmask 255.255.255.0

Показати таблицю маршрутизації
route -n

Додати новий маршрут в таблицю
route add -net 10.10.10.0 netmask 255.255.255.0 dev eth0

Конвертування тексту

Показати список доступних кодувань
iconv -l

Переконвертувати файл з кодування KOI-8R в UTF-8
iconv -f KOI8-R -t UTF-8 -o KOI8-R.txt > UTF-8.txt

Використані матеріали

Під час підготовки матеріалу для учбового курсу та, відповідно, учбового матеріалу, широко використовувались різні матеріали, які доступні в інтернеті. Наразі спробую перерахувати їх.

Якщо когось випадково забув — пишіть і я включу ваш сайт в список використаних матеріалів.

1. **http://www.wikipedia.org/** — на різних мовах
2. **http://proft.me** — матеріал про рівні завантаження системи
3. **http://opennet.ru** — там взагалі багато цінної інформації
4. **http://docstore.mik.ua** — чудовий мануал по IPTables
5. **http://6uestsblog.blogspot.com** — гарна стаття по налаштуванню openvpn
6. **http://yakim.org.ua** — ну як же я міг не взяти матеріали з власного сайту.