Безопасность ОС

Безопасность Linux. Процессы

Доступность ресурсов

- командой ulimit -a можно отобразить системные ограничения по процессорному времени, виртуальной памяти, размеру создаваемых файлов и др.
- аналогичная информация доступна в виртуальных файлах limits, в подкаталогах /proc
- жёсткие ограничения задаются для всех пользователей
- мягкие ограничения пользователь может задать для своих процессов

Атаки на доступность

- cat /dev/zero > /tmp/abcd
- while 1; mkdir 1; cd 1; touch 2; end
- yes 12345 > /dev/null &
- порождение процессов в цикле мешает ps -ef
- ulimit позволяет задать разумные ограничения
- жёсткие ограничения задаются в /etc/profile

Службы по умолчанию

расходуют процессоров время, занимают оперативную память, содержат уязвимости

- portmap, rpc.mountd, rpc.nfsd – сетевая файловая система NFS
- nmbd, smbd – сетевые службы Windows
- named – служба доменных имён
- telnet, rlogin, rexec – сетевое управление системой

<service> stop – отключение неиспользуемой службы

Уровень выполнения

- режим функционирования ОС с инициализацией в стиле Unix
- система в определённый момент находится на одном из уровней
- суперпользователь может перевести ОС на другой уровень командой init N и узнать уровень командой runlevel
  - уровень 0 – остановка системы
  - уровень 1 – однопользовательский режим
  - уровни 2-3 – обычное функционирование
  - уровень 6 – перезагрузка

Запуск по расписанию

- периодический запуск задач обеспечивается службой cron
- cron запускается во время загрузки и остается активным до выключения
- ежеминутно читает файлы заданий и выполняет строки оттуда
- для формирования заданий используется crontab

Запуск по расписанию

- файлы заданий хранятся в /var/spool/cron/tabs
- имя файла совпадает с именем пользователя, по нему же создаётся процесс оболочки
- невыполненные в срок задания игнорируются
- в /etc/cron.deny перечислены пользователи, которым запрещено использование cron
- обычно там находятся имена псевдопользователей
- можно создать cron.allow, имеющий приоритет перед .deny

Запуск по расписанию

- однократное выполнение задач обеспечивается диспетчером очередей задач at
- при выполнении команды at в каталоге /var/spool/atjobs создаётся файл
- в /etc можно создать файлы at.allow и at.deny
- ещё один способ создания периодических процессов: watch -n 60 ps

Командная оболочка

- существует ряд интерпретаторов командной строки, напр., bash, tcsh

- небольшое число команд реализовано в самой оболочке, они называются внутренними, напр., fg, alias, limits, history, echo, jobs

- остальные команды означают запуск процесса из исполняемого файла

- обычно хранящегося в одном из специальных каталогов, напр., /bin, /sbin, /usr/bin, /usr/local/bin

- переменная PATH позволяет найти эти файлы

Командная оболочка

- команда может состоять из трёх частей: имени, опций и аргументов

- опции могут записываться в коротком и длинном виде, напр., -lia --all

- можно объявить псевдоним для любой команды, напр., alias lshome="ls -la ~»

- можно запустить команду в фоновом режиме, завершив её символом & (после пробела)

- узнать запущенные фоновые процессы можно при помощи команды jobs, а вернуть на передний план – командой fg

Командная оболочка

- в одной строке можно ввести несколько команд, напр., clear; pwd; date

- && в качестве разделителя требует успешного выполнения предыдущей команды

- || используется, если команду команду надо выполнить при ошибке в предыдущей, напр., ls -l /root || ls -l /home

Командная оболочка

- с помощью клавиш «вверх» и «вниз» можно перемещаться по истории команд
- команда history покажет последние введённые команды, данные записываются в конце сеанса в ~/.bash_history
- ln -s /dev/null ~/.bash_history
- export HISTSIZE=0

Завершение процессов

- exit – завершить сеанс работы в оболочке
- для завершения работы системы требуются полномочия суперпользователя
- команды: shutdown, poweroff или halt
- завершение работы может быть длительным, например, из-за работы с жёстким диском

Межпроцессное взаимодействие

- сигнал сообщает процессу о наступлении события
- для отправки сигнала используется kill
- процессы сами решают, как им обрабатывать сигналы
- кроме сигнала KILL – принудительного завершения, что реализует планировщик задач
- пользователь может послать сигнал, только процессам, которые запустил сам

Перенаправление потока

- все программы запускаются с тремя открытыми файлами: стандартным вводом, стандартным выводом и стандартным выводом сообщений об ошибках
- оболочка позволяет перенаправлять в другой файл ввод символом «<» и вывод символом «>»
- ls -la /home/user1 > /root/user1.ls
- echo "Содержимое /home/user1" >> /root/user1.ls
- cat /root/user1.ls > /dev/lp0

Каналы

- канал – средство межпроцессного взаимодействия, ограниченный буфер памяти, в который один процесс может писать, а другой – читать

- неименованный канал обозначается символом «|» и часто называется конвейером:
  ls -la /bin | less
  dd if=/dev/hdc | grep "Linux"

- процесс приостанавливается, если буфер полон или пуст

- утилиты ps и top отображают каждый процесс, участвующий в конвейере, отдельной строкой

Терминальный режим

- терминалы появились в 70-х годах XX века
- предназначались для обмена текстовой информацией
- терминальный режим предоставляет необходимый минимум для управления системой пользователем, в том числе и удалённо, напр., по telnet и ssh
- обозначаются ttyN и vcN и определяются в /etc/inittab
- локально переключаются по Ctrl+Alt+FN, выход в графический режим по Ctrl+Alt+F7

Терминальный режим

- регистрацию суперпользователя можно ограничить при помощи /etc/securetty
- Ctrl+C посылает сигнал завершения процессу
- можно ограничить время простоя переменной TMOUT
- можно отправить сигнал STOP терминалу, чтобы прекратить активность в нём, сигнал CONT восстанавливает его работу
- права на консоль по умолчанию – rw--w----, менять их нельзя

Терминальный режим

- запретить запись сообщений в свою консоль членам группы tty можно командой mesg n, а разрешить – mesg y
- писать сообщения можно командами write и wall
- узнать кто работает в консолях можно командами w и who