Using Linux VServer

Enrico Scholz

enrico.scholz@informatik.tu-chemnitz.de

Motivation (1)

Administrator:

- lots of services (FTP, HTTP*, LDAP, KRB, DNS, . . .)
- updates without side-effects
- own hostnames and IPs
- access restrictions

Developer:

- tests in different environments (compiler, libraries, programs, distributions)
- providing binaries for different distributions

Motivation (2)

Businessman:

- selling of “root-servers” (IP, CPU power, disk-space, root-account)

Solutions:

- one physical machine per server
  $\rightsquigarrow$ but: hardware costs, room, cooling, UPS
- multiple dedicated servers on the same hardware

$\Rightarrow$ “virtual servers”

Requirements

- behavior like an ordinary server (same binaries, no special syscalls)
- process isolation
  - `kill(2)`, `ptrace(2)`
  - /etc/init.d/sshd restart
- filesystem isolation
  - no collisions when using standard-paths
  - keeping of secrets
- no backdoors
  - direct hardware-access (/dev/hda)
  - direct kernel-access (/dev/kmem)

$\Rightarrow$ no influence on the function of other servers or of the host

Wishes

- effective
  - performance (CPU, I/O)
  - memory (RAM, disk)
- easy manageable
  - creation
  - operation

  $\Rightarrow$ using of known tools
- limits/quotas for diskspace, CPU, net
- migration to other physical machines

Solutions

- special hardware (S/390)

Solutions

- special hardware (S/390)
- vmware/bochs/qemu
  - usable as usual machine
  - but: high resource consumption; often for i386 only

Solutions

- special hardware (S/390)
- vmware/bochs/qemu
  - usable as usual machine
  - but: high resource consumption; often for i386 only
- UML
  - middle till high resource consumption

Solutions

- special hardware (S/390)
- vmware/bochs/qemu
  - usable as usual machine
  - but: high resource consumption; often for i386 only
- UML
  - middle till high resource consumption
- SELinux
  - fulfills (security related) requirements
  - no complete virtualization (hostname, ip)
  - ???

Solutions

- special hardware (S/390)
- vmware/bochs/qemu
  - usable as usual machine
  - but: high resource consumption; often for i386 only
- UML
  - middle till high resource consumption
- SELinux
  - fulfills (security related) requirements
  - no complete virtualization (hostname, ip)
  - ???
- Linux vserver, BSD Jails, SUN Zones, FreeVPS
  - grouping of processes
  - usage of same hardware and kernel
  - new and already existing access control mechanisms
  - nearly no overhead

vserver

Quickstart (1)

1. download and untaring of the kernel sources

```
$ wget http://ftp.kernel.org/pub/linux/kernel/v2.x/linux-2.x.y.tar.bz2
$ tar xjf linux-2.x.y.tar.bz2
$ cd linux-2.x.y
```

2. download of the corresponding vserver-patch[a] from http://www.13thfloor.at/vserver/project/ and applying of this patch

```
$ bzcat patch-2.x.y-vs1.z.diff.bz2 | patch -p1
```

3. configuration, build and installation of the kernel

```
$ make config
$ make dep && make all modules && make install modules_install
```

[a] 1.2x – stable, 1.3x and 1.9x – experimental

Quickstart (2)

4. download of the userspace tools[a] (util-vserver) from http://www.nongnu.org/util-vserver
5. configuration, build and installation

```
$ rpmbuild -ta util-vserver-0.x.y.tar.bz2 \
  [--without xalan] [--without dietlibc]
# rpm -Uvh ...
```

oder

```
$ ./configure [--prefix=...] <options>* && make
# make install
```

6. reboot

http://linux-vserver.org

[a] $0.x.y \longrightarrow$ stable if no y, pre if y$<$90, rc if 90$\leq$y$<$190 and alpha if 190$\leq$y

Properties (Kernel) (1)

- developer: Herbert Pötzl
- one multiswitch syscall for the entire functionality
- attributes for processes:
  - numeric context-ID (xid)
    $\rightarrow$ processes with xid_1 invisible for xid_2-processes
- attributes for process-contexts:
  - hostname resp. a complete utsname entry(2.6)
  - system- & context-specific(2.6) capabilities
  - flags
  - namespace(2.6)
  - scheduling parameters(2.6)

Properties (Kernel) (2)

- large parts of security based on linux-capabilities (/usr/include/linux/capability.h), e.g.
  - no new devices without CAP_MKNOD
  - no interface-configuration without CAP_NET_ADMIN
  - no filesystem-mounting without CAP_SYS_ADMIN
  - ...
- hiding of filesystem-entries
  some entries in /proc without capability protection, e.g. /proc/sysrq-trigger or /proc/scsi/scsi
  $\Rightarrow$ hiding outside of host-context
- context-quotas
- outbreak-safe chroot(2) environments

Userspace

- two toolsets: “vserver” und “util-vserver”
- low-level syscallwrappers
- vserver == chroot-environment + configuration-data
- management of the vservers
  - creation
  - starting/stopping
  - optimizations
- configuration usually under /etc/vservers/
- starting with “vserver *<id>* start”; stopping with “vserver *<id>* stop”

chroot-environments

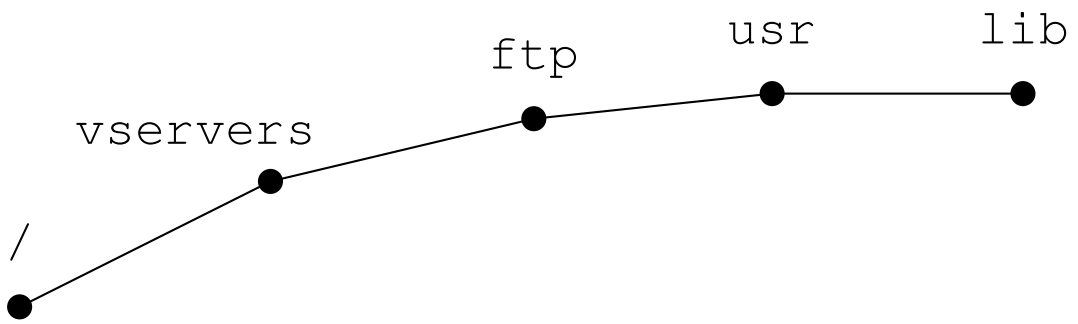
- usually at /vservers/<*id*>
- usually files and directories like in ordinary linuxdistributions, but special installations possible
- distribution within the chroot != host-distribution

chroot-environments

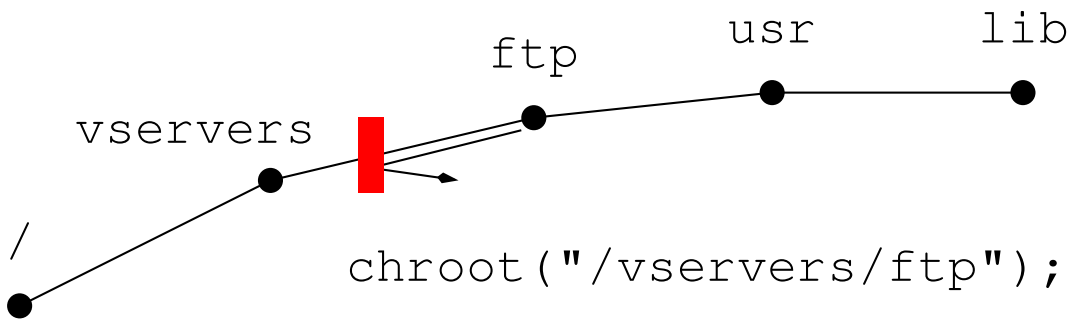
- usually at /vservers/<*id*>
- usually files and directories like in ordinary linuxdistributions, but special installations possible
- distribution within the chroot != host-distribution
- chroot-environment must be assumed as hostile
  - execution of arbitrary programs as root
  - creation, removal, renaming and modification of arbitrary files, symlinks and directories

$\Rightarrow$ special care and kernel-support required

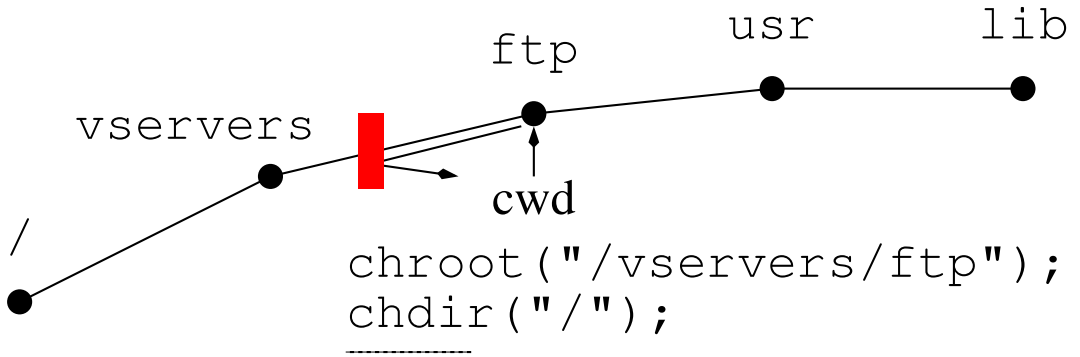
Classical chroot-attack



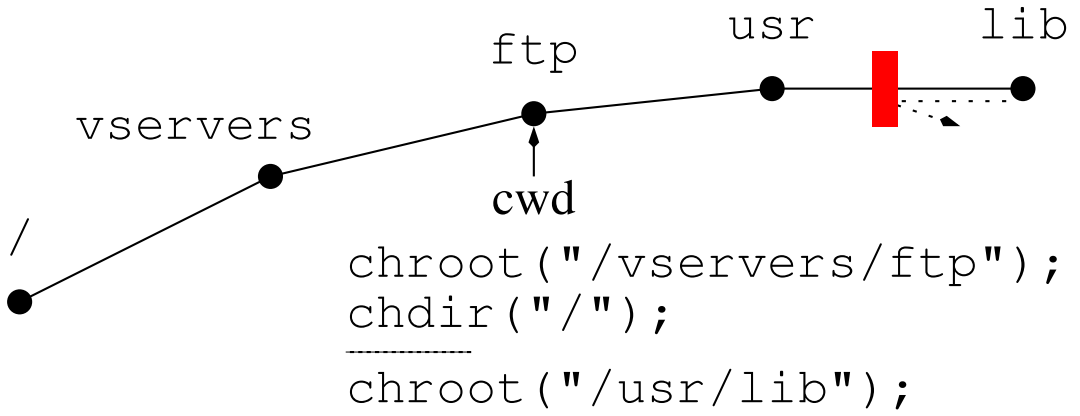
Classical chroot-attack



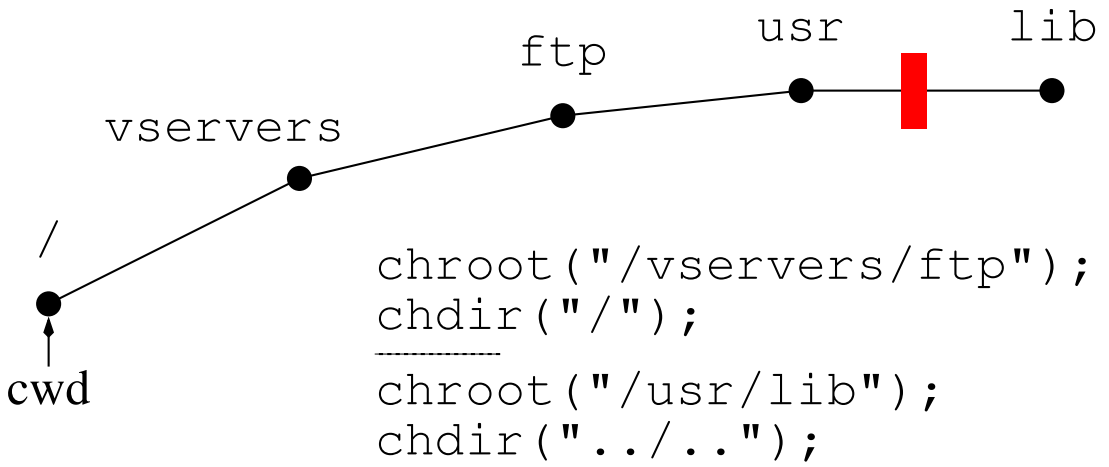
Classical chroot-attack



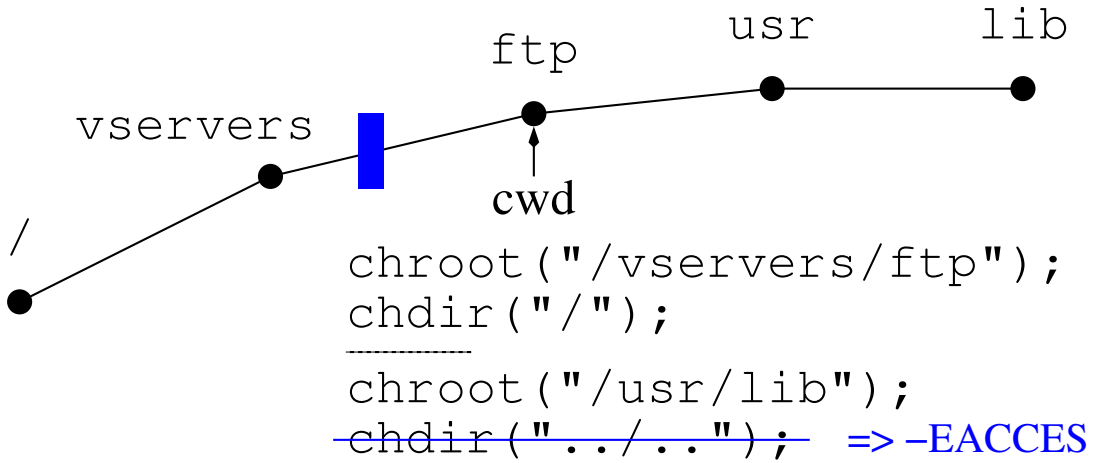
Classical chroot-attack



Classical chroot-attack

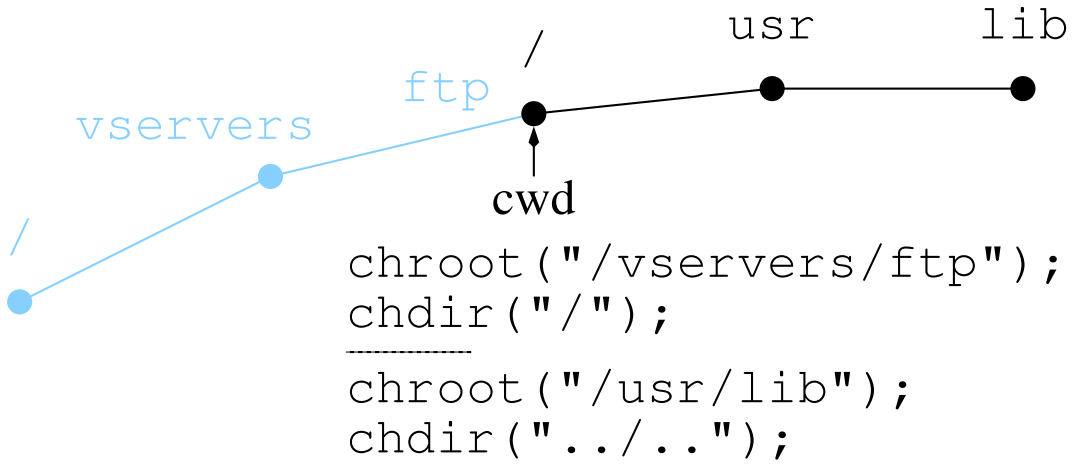


Classical chroot-attack



- static barriers in the filesystem (kernelpatch): not traversable by processes outside of the host-context

Classical chroot-attack



- creation of a new “/”:

```
# mount --rbind /vservers/ftp /
```

⇒ execution in an own namespace

- additional kernel features needed for practial application (migrate())
- cleaning up of /proc/mounts possible
- not fully implemented currently

Excursion: namespaces

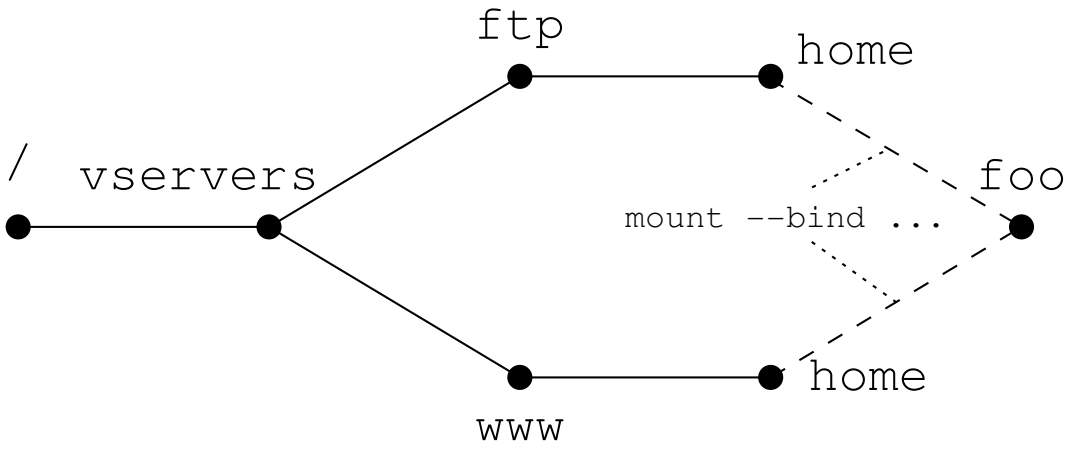
- new namespace with CLONE_NEWNS; documented in clone(2) manpage:
  > Every process lives in a namespace. The namespace of a process is the data (the set of mounts) describing the file hierarchy as seen by that process.
- relative new (kernel 2.4.19); not for vservers only
- conflicts with automounters

Example:

```
[root@kosh root]# vnamespace --new sh
sh-2.05b# mount --bind /bin/rm /bin/ls
sh-2.05b# ls /etc/*
   ... lieber nicht ...
sh-2.05b# exit
[root@kosh root]# ls /etc/*
/etc/DIR_COLORS
...
[root@kosh root]#
```

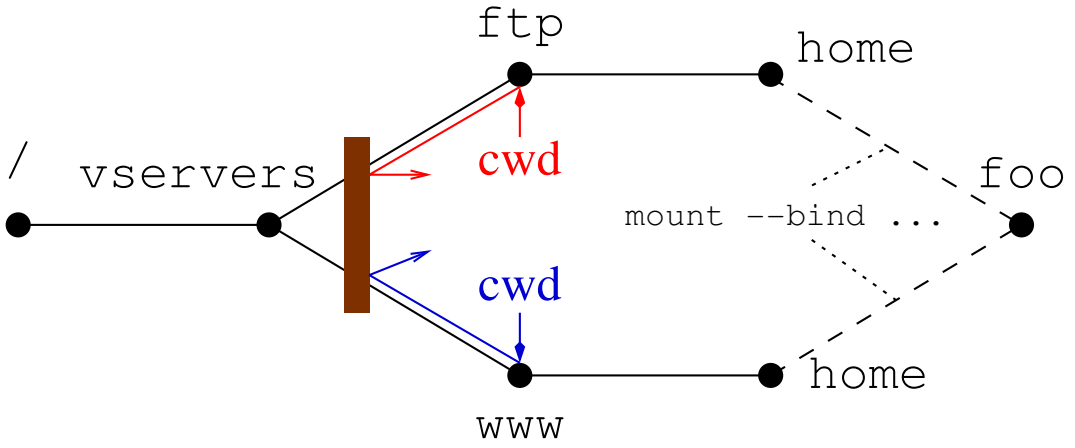
```
[root@kosh root]# ls /etc/*
/etc/DIR_COLORS
...
[root@kosh root]#
```

Infiltrating foreign chroots



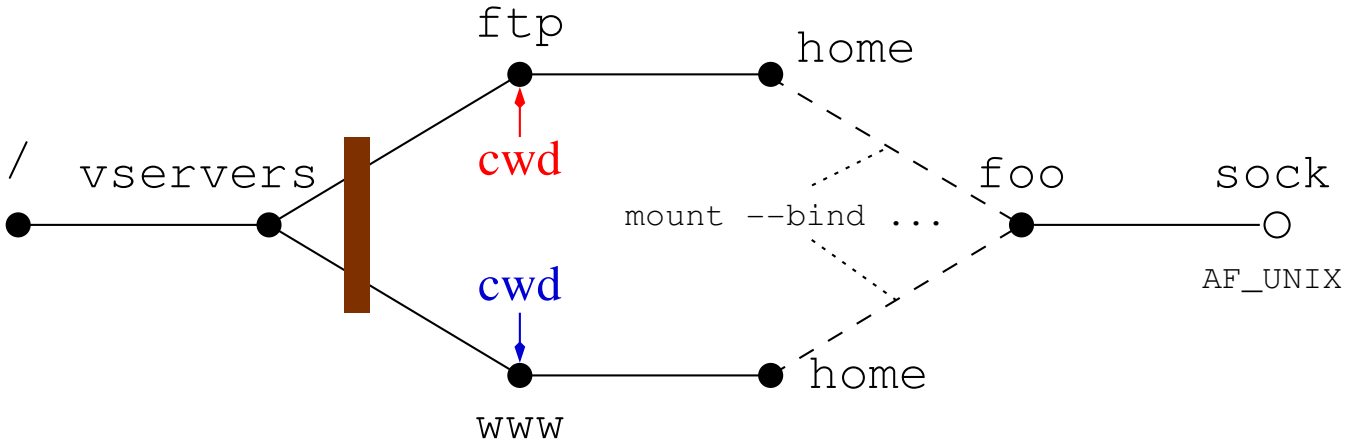
- two vservers “www” und “ftp”
- commonly used /home directory

Infiltrating foreign chroots



- static barrier at /vservers
- root-rights for “red” in “ftp”; “blue” only an ordinary user in “www”

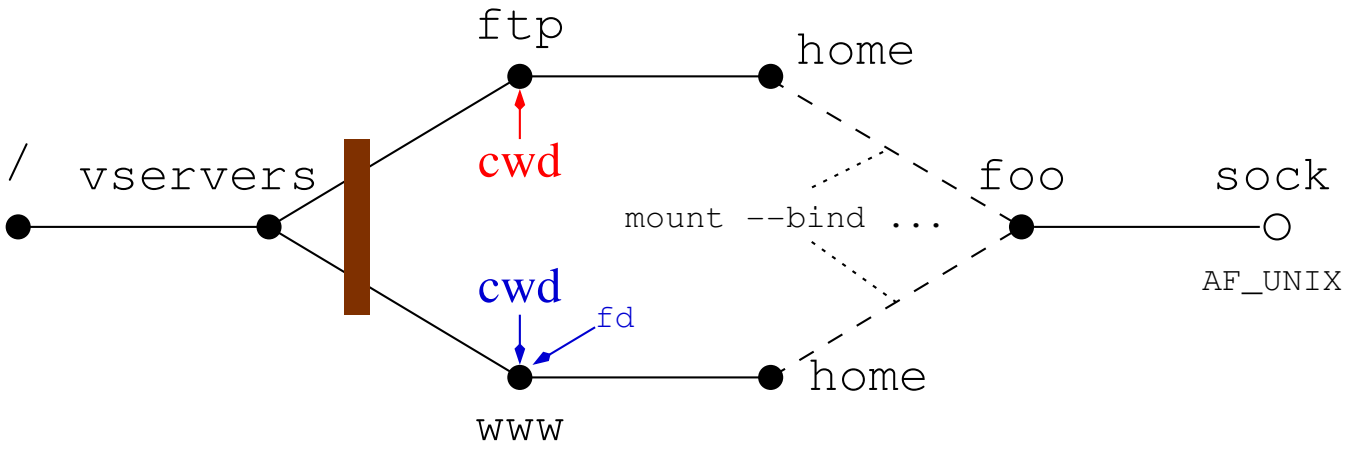
Infiltrating foreign chroots




```
tmp=socket(AF_UNIX,...);
bind(tmp,"/home/foo/sock");
listen(tmp)
s=accept(tmp);
```

```
tmp=socket(AF_UNIX,...)
connect(s,"/home/foo/sock");
```

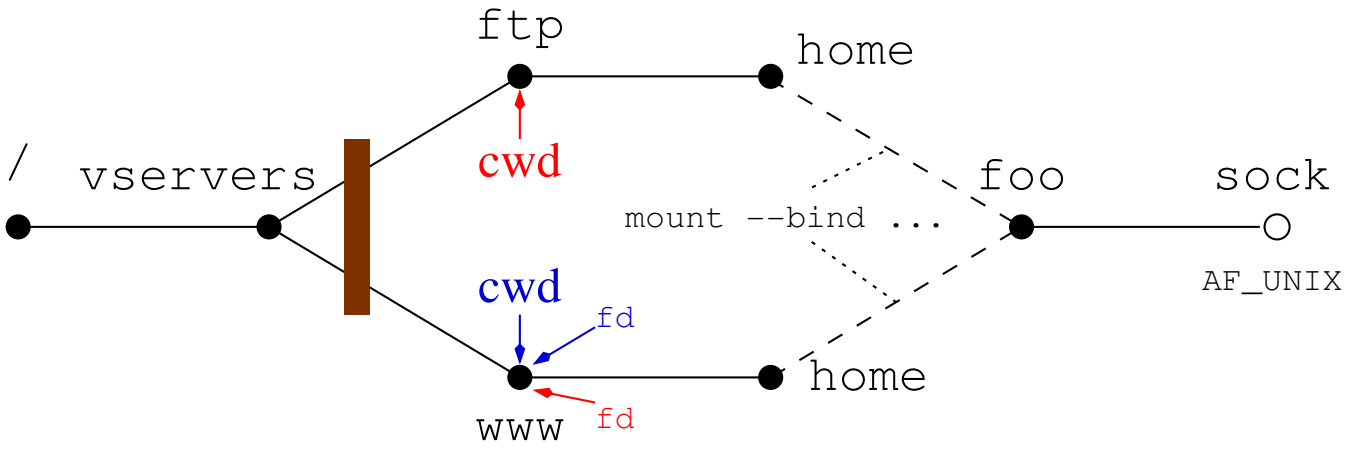
Infiltrating foreign chroots




```
tmp=socket(AF_UNIX,...);
bind(tmp,"/home/foo/sock");
listen(tmp)
s=accept(tmp);
```

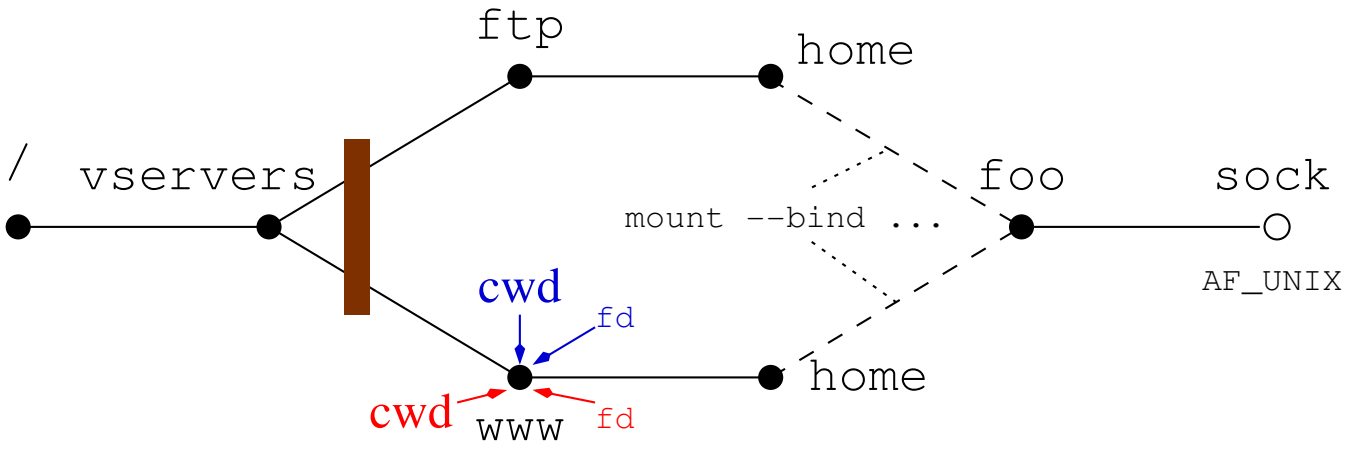
```
tmp=socket(AF_UNIX,...)
connect(s,"/home/foo/sock");
fd=open(".",O_RDONLY);
```

Infiltrating foreign chroots




```
tmp=socket(AF_UNIX,...);          tmp=socket(AF_UNIX,...)
bind(tmp,"/home/foo/sock");       connect(s,"/home/foo/sock");
listen(tmp)                       fd=open(".",O_RDONLY);
s=accept(tmp);
recvmsg(s,{&fd,SCM_RIGHTS});      sendmsg(s,{fd,SCM_RIGHTS});
```

Infiltrating foreign chroots

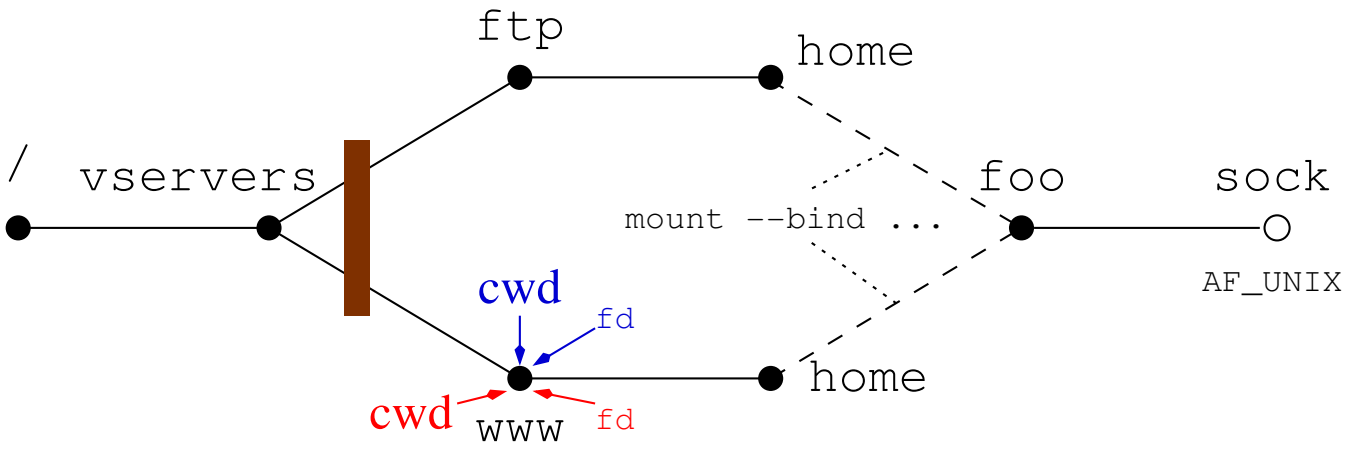



```
tmp=socket(AF_UNIX,...);
bind(tmp,"/home/foo/sock");
listen(tmp)
s=accept(tmp);
recvmsg(s,{&fd,SCM_RIGHTS});
fchdir(fd);
```

```
tmp=socket(AF_UNIX,...)
connect(s,"/home/foo/sock");
fd=open(".",O_RDONLY);

sendmsg(s,{fd,SCM_RIGHTS});
```

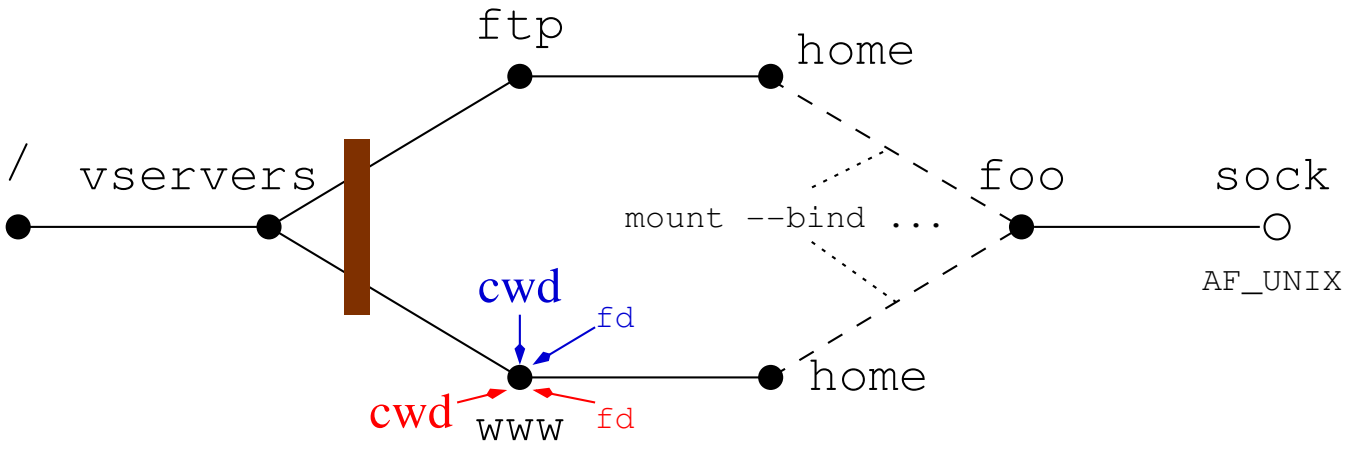
Infiltrating foreign chroots




```
tmp=socket(AF_UNIX,...);          tmp=socket(AF_UNIX,...)
bind(tmp,"/home/foo/sock");       connect(s,"/home/foo/sock");
listen(tmp)                       fd=open(".",O_RDONLY);
s=accept(tmp);
recvmsg(s,{&fd,SCM_RIGHTS});      sendmsg(s,{fd,SCM_RIGHTS});
fchdir(fd);
```

⇒ root-rights for “red” in “www”

Infiltrating foreign chroots

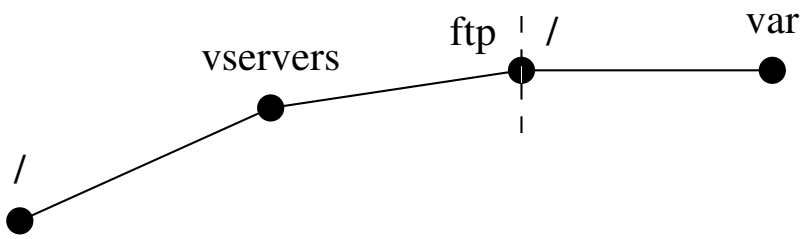



```
tmp=socket(AF_UNIX,...);          tmp=socket(AF_UNIX,...)
bind(tmp,"/home/foo/sock");       connect(s,"/home/foo/sock");
listen(tmp)                       fd=open(".",O_RDONLY);
s=accept(tmp);
recvmsg(s,{&fd,SCM_RIGHTS});      sendmsg(s,{fd,SCM_RIGHTS});
fchdir(fd);
open("etc/passwd",...);
```

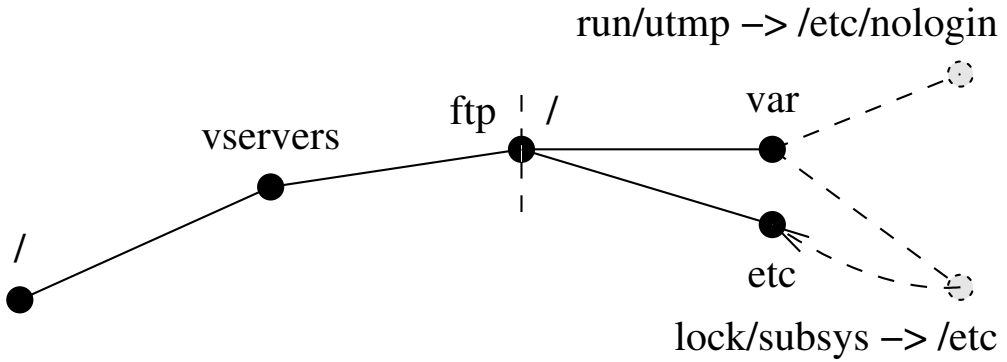
⇒ root-rights for “red” in “www”

- unsolved in vserver; perhaps preventable with SELinux or partly with namespaces

Symlink-attacks (1)



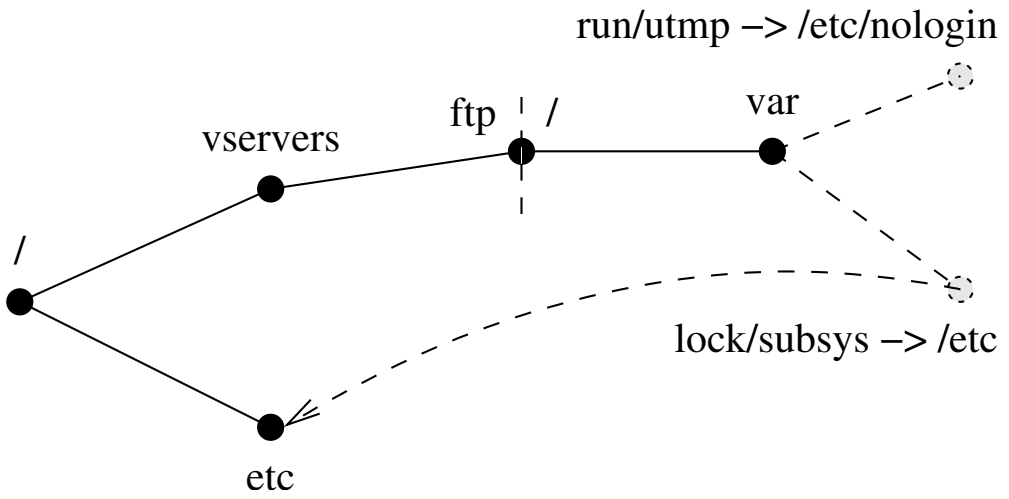
Symlink-attacks (1)



Vserver–Admin:

```
# ln –s /etc /var/lock/subsys
# ln –s /etc/nologin /var/run/utmp
```

Symlink-attacks (1)



Vserver–Admin:

```
# ln –s /etc /var/lock/subsys
# ln –s /etc/nologin /var/run/utmp
```

actions executed by the host-administrator at “/”:

```
# rm -f /vserver/ftp/var/lock/subsys/*
# touch /vservers/ftp/var/run/utmp
# mount /dev/hda1 /vserver/ftp/var/lock/subsys
```

Symlink-attacks (2)

- preventing symlinkattacks by entering the directories in a *secure* way; often implemented by

```
chroot(vserver_rootdir);
chdir(destination_directory);
action();
```

- operations only in "." (exec-cd tool)

```
// Usage: exec-cd <dir> <cmd> <args>*
old_fd = open("/", O_RDONLY);
chroot(".");
chdir(argv[1]);
new_fd = open(".", O_RDONLY);
fchdir(old_fd);
chroot(".");
fchdir(new_fd);
execv(argv[2], argv+2);
```

- e.g.:
  # cd /vservers/ftp && exec-cd /var/lock/subsys mount /dev/hda1 '.'

Other attacks

- modification of files which are usually writable for root only (/etc/passwd, rpm-database)
  $\Rightarrow$ overflows
  $\Rightarrow$ execution of code in host-context
  Solution: Important files outside of VServer; helperprograms in VServer-context
- dynamic library-loading (/lib/libnss_*) (functional deficiencies also)
  Solution: dietlibc instead of glibc
- races when traversing the filesystem
  Solution: secure directory-changing; enforcing of stopped VServers
- no chroot(2) before entering another context
  $\Rightarrow$ hijacking through ptrace(2)
  Solution: do not do this...

util-vserver, stable (1)

- “vserver” and stable-branch of “util-vserver”:
  - nearly the same functionality
  - “util-vserver” forked at “vserver 0.23”
- spread widely
- very good documentation

util-vserver, stable (1)

- “vserver” and stable-branch of “util-vserver”:
  - nearly the same functionality
  - “util-vserver” forked at “vserver 0.23”
- spread widely
- very good documentation
- lots of open wishes
- not applicably in hostile environments because lots of attack-vectors for symlinkattacks and races

$\Rightarrow$ complete redesign necessary

- no active development; only bugfixes
- no support for new kernel-features

util-vserver, stable (2)

configuration in /etc/vservers/<*id*>.conf

```
IPROOT="192.168.5.32 192.168.5.64"
IPROOTDEV=eth0
S_HOSTNAME=ftp.nowhe.re
ONBOOT=yes
S_DOMAINNAME=
S_NICE=5
S_FLAGS="lock nproc fakeinit"
ULIMIT="-HS -u 200"
S_CAPS=""
```

- bash-scriptlet; applied with source
- another script /etc/vservers/<*id*>.sh for tasks after and before starting and stopping and VServers

util-vserver, alpha

- designgoals:
  - easy extensible
  - no races and symlinkattacks
  - embedded solutions for standardtasks
  - support of new kernelfeatures
- retaining of stable's base-commands, but lots of new program and reimplementation of old ones
- new configuration scheme
  - parseable by C and shell
  - manageable with cfengine
  - support of new features

util-vserver, alpha

- designgoals:
  - easy extensible
  - no races and symlinkattacks
  - embedded solutions for standardtasks
  - support of new kernelfeatures
- retaining of stable's base-commands, but lots of new program and reimplementation of old ones
- new configuration scheme
  - parseable by C and shell
  - manageable with cfengine
  - support of new features
- rare documentation

http://www.linux-vserver.org/index.php?page=alpha+util-vserver

Configuration

- configuration in /etc/vservers/<*id*>/ **directory**

```
/etc/vservers/ftp
|-- capabilities
|-- context
|-- flags
|-- fstab
|-- interfaces
|   |-- 00
|   |   |-- ip
|   |   `-- name
|   |-- bcast
|   |-- dev
|   `-- mask
|-- run -> /var/run/vservers/ftp
|-- run.rev -> ../.defaults/run.rev
`-- vdir -> /etc/vservers/.defaults/vdirbase/ftp
```

- files and symlinks; mostly one-entry-per-line/file
- path of configuration-directory identifies a vserver; chroot-path freely chooseable
- only formal documentation

vcontext (1)

- formerly: chcontext, but no support of new kernel-technologies
- creation (`--create`) and entering (`--migrate`) of process-contexts
- Take care about security when entering a context! (ptrace(2))
- usually additional operations between create and migrate
- invocation usually as:

```
vcontext --create -- \
  vattribute --set -- \
  vlimit ... -- \
  vsched ... -- \
  vcontext --migrate-self --endsetup -- \
  <command>
```

vcontext (2)

Example:

```
# vcontext --create ps axh
New security context is 49153
 5440 pts/1    R      0:00 ps axh

# vcontext --migrate --xid 43 ps axh
 5068 ?        S      0:00 /sbin/syslogd
 5102 ?        S      0:00 /usr/sbin/exim4 -bd -q30m
 5108 ?        S      0:00 /usr/sbin/inetd
 5112 ?        S      0:00 /usr/sbin/atd
 5115 ?        S      0:00 /usr/sbin/cron
 5447 pts/1    R      0:00 ps axh
```

vattribute

- setting/removal of attributes and capabilities
- syntax for such values:
  - string
  - number: interpreted as a bitpattern
  - prefix ‘~’ or ‘!’: unsetting of the pattern
  - prefix ‘^’: bitnumber instead of pattern

Example:

```
# vcontext --create vattribute --set --flag hidemount cat /proc/mounts
# vcontext --create -- \
    vattribute --set --secure -- \
    vcontext --endsetup --migrate-self -- \
    mknod /tmp/test c 1 2
New security context is 49183
mknod: ‘/tmp/test’: Operation not permitted
```

chbind

- binding of IPs to processes
- uncertain future. . . perhaps completely different networking or replacing with vnet

Example:

```
# chbind --ip 10.1.0.1 cat /proc/self/status | grep ipv4root
ipv4root is now 10.1.0.1
ipv4root: 0100010a/00ffffff
ipv4root_bcast: ffffffff
ipv4root_refcnt: 2
```

Misc (1)

vkill

- atomic sending of signals to process-contexts

vnamespace

- creation and entering of namespaces

vlimit

- setting and showing of resource-limits

Misc (2)

vuname

- setting and showing of utsname-entries

vserver-info

- querying of single attributes of contexts and vservers
- important for bugreports:
  vserver −info − SYSINFO

Management

vserver creation

- “normal” systeminstallation possible
- BSD Jails:

```
# make -C /usr/src DESTDIR=/vservers/foo install
```

vserver creation

- “normal” systeminstallation possible
- BSD Jails:

```
# make -C /usr/src DESTDIR=/vservers/foo install
```

- Fedora Core:

```
# make -C /usr/src DESTDIR=/vservers/foo install
make: Entering directory ‘/usr/src’
make: *** No rule to make target ‘install’.  Stop.
make: Leaving directory ‘/usr/src’
```

- lots of distributions with different installation-methods

vserver creation

- “normal” systeminstallation possible
- BSD Jails:

```
# make -C /usr/src DESTDIR=/vservers/foo install
```

- Fedora Core:

```
# make -C /usr/src DESTDIR=/vservers/foo install
make: Entering directory ‘/usr/src’
make: *** No rule to make target ‘install’.  Stop.
make: Leaving directory ‘/usr/src’
```

- lots of distributions with different installation-methods

$\Rightarrow$ implementation of *some* of them in util-vserver:

- “apt-rpm” for Fedora/RH vserver
- “debootstrap” for Debian vserver
- “skeleton” for base directory-structure and configuration

vserver . . . build

- documented by “vserver - build --help”
- Examples:
  - ```
    # vserver test0 build -m apt-rpm --hostname test0.nowhe.re \
      --interface 10.0.1.0 --netdev eth0 --netprefix 23 \
      --context 42 -- -d fc1
    ```
  - ```
    # vserver test1 build -m debootstrap --hostname test1.nowhe.re \
      --interface 10.0.1.1 --netdev eth0 --netprefix 23 \
      --context 43 -- -d sarge
    ```
  - ```
    # vserver test2 build -m skeleton --hostname test2.nowhe.re \
      --interface 10.0.1.2 --netdev eth0 --netprefix 23 \
      --context 44
    ```
- configuration of parameters (mirrors, packet-lists) in /etc/vservers/.defaults/apps/debootstrap/* and /etc/vservers/.distributions/*
    ```

vserver . . . start

1. creation of namespaces
2. creation of network-interfaces
3. directory-mounting
4. creation of process- and network-contexts
5. activation of limits and capabilities
6. execution of the init-process
   - shortcut with “/etc/rc.d/rc 3”, or
   - regular /sbin/init – often lots of unwanted actions

   ⇒ `fakeinit` mechanisms needed (getpid()==1)

Attention: at least one process needed in the context

Example:

```
# vserver test0 start
# vserver --debug test1 start
```

vserver . . . stop

- either
  - sending of `SIGINT` to the init-process, or
  - execution of “/etc/rc.d/rc 6”
- explicit “vkill –xid *<xid>* -s 9”
- no explicit unmounting needed when using namespaces

Example:

```
# vserver test0 stop
# vserver --debug test1 stop
```

vserver . . . enter|exec

- execution of commands within the vserver
- similar actions as “vserver . . . start”, but entering instead of creation of namespace and contexts

⇒ no overriding of parameters and restrictions

- only for administration-tasks but not for regular service (e.g. missing /dev/pts entries)

Example:

```
# vserver test0 exec ps axh
  640 ?        S      0:00 syslogd -m 0
 1188 pts/1    R      0:00 ps axh
# vserver test1 enter
test1:/#

# uname -a
Linux delenn 2.6.5ensc-0.3 #1 Thu Apr 15 ... 2004 i686 i686 i386 GNU/Linux
# vserver test1 exec uname -a
SCO UnixWare test1.nowhe.re 7.1 #1 Sat Feb 29 ... 2003 s390 GNU/Linux
```

vps

- displays all processes on host + contexts
- executed in a special watcher-context (XID 1)

Example:

```
# vps ax
  PID CONTEXT             TTY       STAT   TIME COMMAND
    1       0 MAIN        ?         S      0:05 /sbin/minit
    2       0 MAIN        ?         SWN    0:00 [ksoftirqd/0]
 ...
 5068      43 test1       ?         S      0:00 /sbin/syslogd
 5102      43 test1       ?         S      0:00 /usr/sbin/exim4 -bd -q30m
 5108      43 test1       ?         S      0:00 /usr/sbin/inetd
 5112      43 test1       ?         S      0:00 /usr/sbin/atd
 5115      43 test1       ?         S      0:00 /usr/sbin/cron
 ...
 5256      42 test0       ?         S      0:00 syslogd -m 0
 ...
 5276       1 ALL_PROC    pts/1     S      0:00 vps ax
 5277       1 ALL_PROC    pts/1     R      0:00 ps ax
```

vserver-stat

- overview about running vservers resp. process-contexts

Example:

```
# vserver-stat
CTX   PROC    VSZ    RSS  userTIME   sysTIME    UPTIME NAME
0       37  43.8M   4.6K   0m37s86   0m22s52  13h00m44 root server
42       1   1.5M   145    0m00s00   0m00s00   2m56s60 test0
43       5  10.5M   965    0m00s10   0m00s00   8m53s31 test1

# vserver-stat
CTX   PROC    VSZ    RSS  userTIME   sysTIME    UPTIME NAME
0       48 143.9M   3.5K  19h09m59   8h40m24  56d45h05 root server
2        3   6.7M   172    3h01m34   1h22m00  28d29h14 vpn
82      17  90.6M   784    4h44m09   2h13m18  28d26h53 cvs
133      5   1.6M    52   31m03s55   5m46s86  23d33h26 paris
146     11  48.9M     2K  37m07s12   8m55s74  28d26h53 mirror
147      7   3.8M   180   10h46m17   2h48m54  28d21h46 mirror-master
153      9  74.6M     4K  36m31s24   7m08s33  28d25h46 ldap1
```

vrpm (1)

- external or internal rpm-database
- advantage internal: rpm works within the vserver
- advantage external: simple bootstrapping (“vserver . . . build”)
- switching between both methods with
  vserver ... pkgmgmt externalize|internalize

Syntax:

vrpm <vserver>+ −− <rpm−options>+

Internal vrpm:

- realized with “vserver . . . exec rpm”

vrpm (2)

External vrpm:

- `LD_PRELOAD` wrapper for execv(3), getpwnam(3) et.al.
  ⇒ execution of %scriplets im the vserver-context
  ⇒ NSS lookups while unpacking the packages
- complicated mounting of the database so that access through vserver-processes or %scriptlets impossible
- files at /etc/vservers/<*id*>/apps/pkgmgmt/. . . resp. /vservers/.pkg/<*id*>/rpm

Example:

```
# vrpm test0 -- -q glibc fedora-release rpm
glibc-2.3.2-101.4
fedora-release-1-3
package rpm is not installed
# vrpm test0 -- -Uvh /tmp/tetex-2.0.2-13.i386.rpm
```

vapt-get

- for rpm-based vservers: both external and internal management possible
- else: realized with “vserver . . . exec apt-get”

Syntax:

vapt−get <vserver>+ −− <apt−get−options>+

Example:

```
# vapt-get test0 -- install bzip2-libs
...
Preparing...                ########################################### [100%]
   1:bzip2-libs             ########################################### [100%]
Done.

# vapt-get test1 -- install libbz2-1.0
...
Unpacking libbz2-1.0 (from .../libbz2-1.0_1.0.2-1_i386.deb) ...
Setting up libbz2-1.0 (1.0.2-1) ...
```

setattr, showattr (1)

- often: lots of vservers with the same distribution
  $\Rightarrow$ installation and execution of identical packages, binaries and data-files
- idea: copies with hardlinks (“ln A B”)
  $\Rightarrow$ saves diskspace
  $\Rightarrow$ saves memory (mapping of programs and libraries)

setattr, showattr (1)

- often: lots of vservers with the same distribution
  - ⇒ installation and execution of identical packages, binaries and data-files
- idea: copies with hardlinks (“ln A B”)
  - ⇒ saves diskspace
  - ⇒ saves memory (mapping of programs and libraries)
  - ⇝ manipulations possible, as changes visible on every vserver

    # echo mycode >/usr/sbin/httpd
    - no COW oder unionfs in Linux

setattr, showattr (2)

- solution: special immutable-flag; e.g. “chattr +i . . . ”
  - not settable outside of host-context
  - ⇝ package-management (Updates) not possible anymore

setattr, showattr (2)

- solution: special immutable-flag; e.g. “chattr +i . . . ”
  - not settable outside of host-context

  $\rightsquigarrow$ package-management (Updates) not possible anymore
- additional flag
  - preventing modifications
  - allowing to remove files
- low-level functionality in setattr und showattr tools
  - forbidding/allowing of modifications with “–iunlink”
  - changing of visibility
  - setting of the chroot-barrier flag

  $\Rightarrow$ “setattr --help”

setattr, showattr (3)

Example:

```
# touch /vservers/test0/{a,b,c}
# ln /vservers/test0/{a,b,c} /vservers/test1/
# setattr --iunlink /vservers/test0/a
# chattr +i /vservers/test0/b
# showattr /vservers/test0/{a,b,c}
---bUI- /vservers/test0/a
---buI- /vservers/test0/b
---bui- /vservers/test0/c

# vserver test0 enter
[root@test0]# echo a>a
bash: a: Permission denied
[root@test0]# echo a>b
bash: a: Permission denied
[root@test0]# echo a>c
[root@test0]#

[root@test0]# rm -f a b c
rm: cannot remove ‘b’: Operation not permitted

# vserver test1 enter
test1:/# cat /c
a
```

vunify

- applies the settattr-concept to entire directory-trees
- function:
  1. searches same files
  2. sets the `iunlink` flag
  3. creates a hardlink
- uses static exclude-lists and information of package-management about configuration files

vunify

- applies the settattr-concept to entire directory-trees
- function:
  1. searches same files
  2. sets the `iunlink` flag
  3. creates a hardlink
- uses static exclude-lists and information of package-management about configuration files
- complete Fedora Core 1 installation has only approx. 30 MB unsharable files
  $\Rightarrow$ 2.6 GB diskspace for 20 vserver á 2 GB

Prospective

- new network-concept: tagging of network-packets, iptables, routing-tables
- documentation
- testsuits
- alpha $\longrightarrow$ beta $\longrightarrow$ stable (before GNU Hurd??)

References

- project-homepage http://linux-vserver.org
- util-vserver http://www.nongnu.org/util-vserver
- #vserver at oftc.net

Questions?