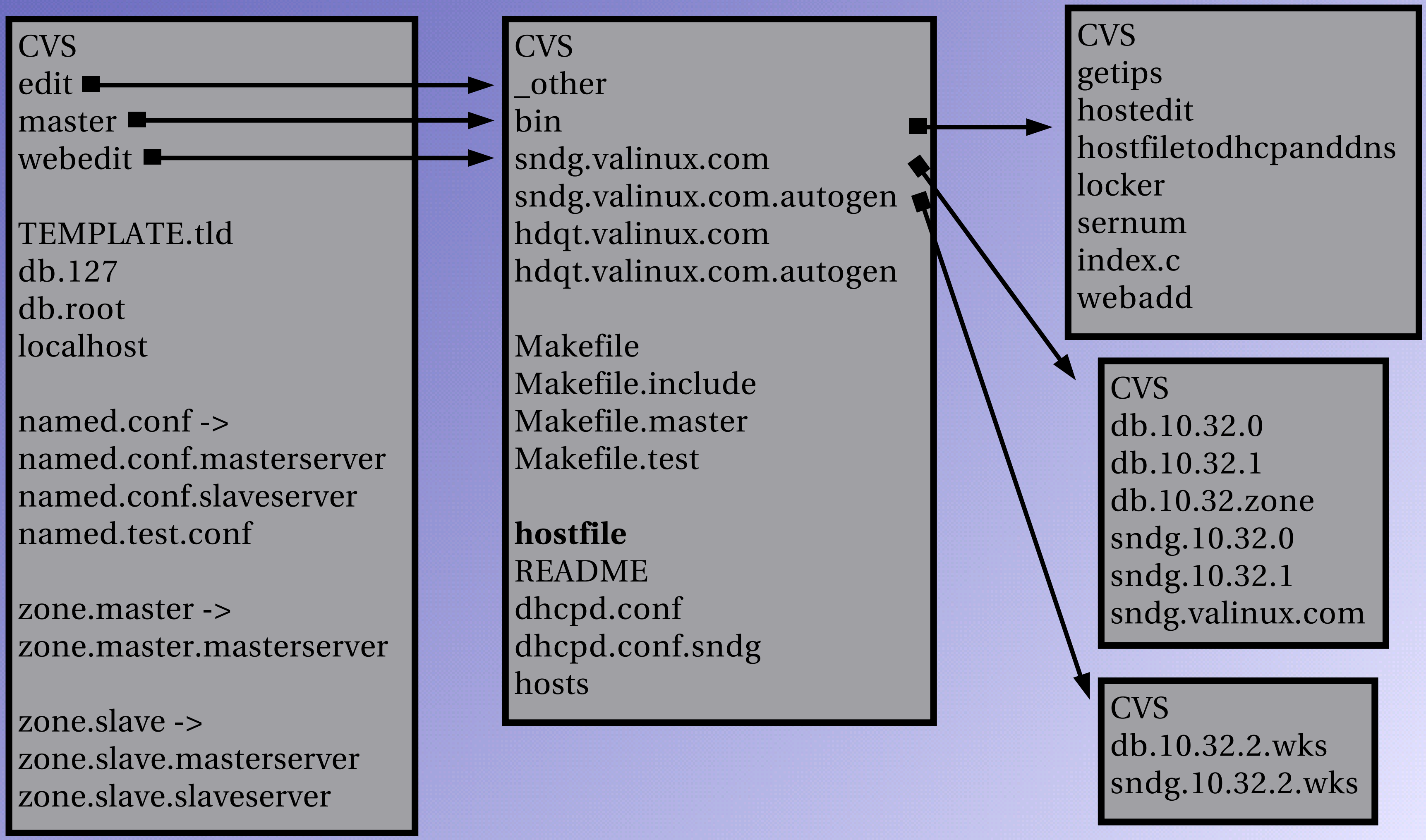
Maintenance of DNS, DHCP, and YP tables from a unique hostfile

Marc MERLIN <marc_bts@valinux.com>

VA LINUX SYSTEMS

- Description of /etc/bind/tree
- What the hostfile looks like
- Why CVS and not RCS
- Problems with CVS
- How DNS files are split in hand and script maintained
- Perl program involved
- The magic of makefiles
- YP hostfile generation
- DNS file checking before restart
- DHCP file checking before restart
- Web front end to let users register their hostnames
- Security concerns

/etc/bind tree



Marc MERLIN <marc_bts@valinux.com>

What the hostfile looks like

- The best thing is to show a sample

IP : 10.x.y.z
MAC : AA:BB:CC:DD:EE:FF
Hostname : hostname (up to 23 chars)
User Login/Other : userlogin (or Server/Switch/Printer/Baan)
RP : emailalias if above isn't a user login
OS : Description of OS (up to 43 chars)
Function : Machine function (see web page for choices) (37 chars)
Notes : Optional notes (up to 40 chars) / NULL
Last Edit Login : Admin user login

IP : 10.1.3.11
MAC : 00:90:27:6F:9B:D8
Hostname : magic
User Login/Other : merlin
RP : NULL
OS : Debian Linux (unstable)
Function : Main workstation
Notes : NULL
Last Edit Login : merlin

From this information, we can generate forward and reverse DNS tables, a DHCP entry and a YP hosts file

CVS vs RCS

- RCS sucks
- CVS sucks less :-)
- RCS' exclusive locks would cause multiple edits to hang (not too bad for a web edit, but bad for an edit by an admin, as it could be locked for hours)
- ci -u vs ci -l is just a pain in the butt. They both suck for different reasons and no one agrees on which one sucks less. They do not mix and match
- Since there is no shared access with RCS, there are no merges either
- CVS allows for storing the history files out of our tree, possibly on a remote machine.
- CVS is the best choice for zone.slave files in /etc/bind (files that are spread over several servers), so we might as well be consistent, the sysadmins will thank you ;-)

Problems with CVS

- CVS will blow away a suid bit
- CVS does not preserve symbolic links
- CVS doesn't allow you to check files back in as root.
  - That's a problem because our makefiles and processes run as root.
  - CVS does a getlogin(3) which looks at utmp, so you can't just su as someone else
  - One option is to patch CVS to remove this restriction
  - Another one is to use a remote repository with cvsuser@machine:/path/to/cvsroot (somewhat slow though).

- Multiple edits means possible conflicts, whoever checks back in first wins
  - For an admin, a failed checkin means that they get to do a cvs co again and hand resolve the conflict if there is one. That's manageable
  - For a web submission, it's bad if the script fails and bails, leaving a file with a failed merge that will screw up any further submissions
  - The "fix" is to do a cvs update just before modifying the file. If the commit fails, we erase the modified file, check it out again, and either temporary fail the submission or retry a second time
  - For admins, the quicker they do the edits, the less chances they'll have to have a conflict :-)

How DNS files are split

- /etc/bind/edit/domain.fdqn.tld

The first net(s) contains routers and servers, and are edited by hand like normal DNS files

  - db.10.x.zone (referenced in zone.master, points to all the other db.10.x.y files)
  - db.10.x.1
  - domain.10.x.1
  - domain.fdqn.tld (referenced in zone.master, points to all the other domain.10.x.y files)

  The reason we do this is that servers and routers are typically not doing DHCP, and they have more complex configurations like IP aliases, multiple A records, CNAMES, MXes, etc.. It's not worth it to modify the simple hostfile format to account for all those possibilities

- /etc/bind/edit/domain.fqdn.tld.autogenerated

  The remaining nets contain DHCP workstations and those files are generated from the main hostfile at the same time than dhcpd.conf

  - db.10.x.2-y
  - domain.10.x.2-y

- /etc/bind/edit/_other

  That's where you fit zones that don't contain DHCP IPs and that aren't split in subfiles

- But my network is different.

While it's simpler to cut on a byte boundary, as long as you aren't mixing up server and workstation IPs (i.e. you want ranges for each), you're fine.

Perl programs involved

- hostfiletodhcpanddns is the main program which reads the hostfile and generates the multiple forward and reverse DNS zones, as well as the dhcpd.conf file(s). It can be called by hand with the --test flag to just validate the syntax of dhcpd.conf, and for debugging purposes, it can be fed an alternate hostfile, and generate files in a different directory

- getips greps all the hostnames from all the DNS zones and generates a flat yp host file. It has basic support for dealing with names that are present in several zones

- sernum is used to increase the serial number in a DNS zonefile. It modifies the zonefile in place, and insists on it having a specific format.

- webadd receives a host change/add request from the CGI. It sits along with the other scripts, on the primary name server.

- hostedit is the actual CGI. It gets input from a web form, validates that input, sshs over to the name server, and runs webadd with the arguments provided.

The magic of Makefiles

- Why Makefiles?
  While the details of the GNU Makefile syntax are not always well known by sysadmins, if you don't mind a few .task files to keep track of milestones that don't produce output, Makefiles are the best way to keep track of jobs with multiple dependencies and targets like this one.

- /etc/bind/edit/Makefile
  Since the whole tree is under CVS, we use a unique Makefile which runs another Makefile depending on which tree make is run

- /etc/bind/edit/Makefile.include
  - The most work is in conveying all the dependencies
  - You can use the built in debug to make sure keyword expansion works as planned
  - You need to set the following major variables:
    - FWDZONES: all the forward dns zones
    - REVZONES: all the reverse ones
    - TOCVSFILES (optional): files that we should automatically try to check back in when make is run
    - BINDCVSFILES (optional): files that we chattr -i and +i in the master tree to give a clue to the unknowledgeable sysadmin that one should never edit files in the master tree :-)

  This was also used to know what to do cvs co in the master tree on, but that turned out to be a problem when zones were added and removed, so the variable isn't actually used for CVS co anymore.

The magic of Makefiles

- /etc/bind/edit/Makefile.test

  This is the magic makefile that makes and tests everything before promoting the data to CVS and to the /etc/bind/master/ directory

  - cvs first runs a cvs update to make sure that the tree is up to date before we try to commit files, and after that it runs through TOCVSFILES to make sure files that are maintained by hand are checked in (and you get to see a cvs diff and cancel if you see something that is amiss).
  - makehostfile regenerates all the automatically maintained DNS files if the hostfile was modified since last time they were updated (fake target is .makehostfile)
  - forward points to all the all the zone files, which in turn should have dependencies on their sub files (subnets)
  - reverse points to the different db.10.x.zone files, which have their dependencies too
  - reloaddhcp will then test the new dhcpd.conf files if there are any
  - reloadnamed does the same thing with DNS zone files :-)
  - For any of the above two, if the test is successful, the relevant target in Makefile.master is run from the /etc/bind/master directory (basically, the successfully tested files are promoted and the real daemon is restarted)
  - There are also several test targets that you can run if you just want to check a specific config without actually checking in and promoting your changes: testhostfile will run the many syntax checks that are applied to the main hostfile, and testnamed / testdhcp will check if the current config files are valid or not with a fake daemon run

The magic of Makefiles

- /etc/bind/edit/Makefile.master
  - By the time you run one of the two targets, reloaddhcp or reloadnamed, the configuration has been fully checked and promoted to CVS.
  - The main point is that no one is doing editing here, all the main files are flagged as immutable
  - Only tested configurations are promoted and allowed to be used for daemon restarting
  - You don't have the problem of someone restarting named while someone else is editing another file, ending up with named not restarting or restarting with incomplete/incorrect data

YP hosts file generation

- In our company, we want to avoid having the same hostname in different subdomains unless it's a service name (like smtp / mail / pop).
- We also thought it wouldn't hurt to have a YP host file to be able to grep names out easily, which you can't do with DNS.
- That said, it had to be simple because parsing all the zone files and knowing which hostname is in which subdomain is actually a lot of work, and being a sysadmin, I have to be lazy whenever possible :-)
- The solution I ended up with was simply to grep all the hostnames from the DNS files, not worry in which subdomain they are, and save the resulting names and IPs in a flat text file which can be used as a YP hostfile. You will have to enhance/fix the script, called getips for your own configuration and domain names
- This flat YP hosts file lets us detect conflicts with names that aren't in the hostfile (i.e. server names that are maintained in a separate file because they're not doing dhcp).
- Getips can be configured to ignore hostname conflicts that are alright (function names that are duplicated across subdomains)

DNS file checking

- You just cannot afford to have either an admin, or especially a web query screw up all of your DNS configuration, take over a server cname, or whatever.
- You don't want your DNS server to not even restart correctly and stop serving some zones altogether because they're too broken.
- The way we solve this is by running a test named process on a different port, and parse its logs to make sure that there are no warning or errors.
- To be able to test the right files, we generate a named.test.conf from the main named.conf (with sed), that way the test named configuration is always up to date
- That said, there are things that aren't illegal DNS wise (like the same name pointing to different A records). hostfiletodhcpanddns for instance will make sure that they aren't duplicate names within its hostfile, but it won't check for names that conflict with names that aren't in the hostfile. That part is checked by our getips script.
- After all those checks, when the DNS files are promoted to the master directory, we check the named.log file after bind is restarted, to make sure that indeed everything went right (we've had failures due to new zone files not checked in DNS, and working in the test directory and failing in the master directory, since the file is absent).

Dhcpd.conf checking

- Making sure that the new dhcpd.conf is valid is vital because dhcpd will plain not restart and new dhcp requests will fail if the new config file has some syntax error
- Checking dhcpd.conf is somewhat similar. It is simpler because it only involves one file and you don't need to parse a log file to scan for errors, but unfortunately dhcpd 2.2 was written in a way that prevents you from running more than one copy on a given machine: you can run it on a different port, but you cannot specify where the pid file goes, which causes subsequent invocations of dhcpd to fail because one process is already running.
- The quick fix was to copy dhcpd as dhcpd.test and patch the binary to use a different pid file. Of course a cleaner way is to patch the source and recompile the binary :-)
- The other thing you want to check is that the dhcpd.conf file(s) you generate from hostfiletodhcpanddns have to have a null subnet declaration for the subnet the name server is on

Web front end for users

- While this whole setup is beneficial for sysadmins (you do not want to have a sysadmin edit 4 different files every time a host record is changed, added or removed. Not only is it a waste of time, but he/she will screw up eventually)
- That said, even with a single hostfile, you can have problems (like multiple people editing the same file in /etc/bind/edit, using different editors that do not agree on how to do locking), or having another admin start editing other files before the first admin runs make (although the cvs diff in make will catch that)
- Of course, what you really want to avoid, is users coming to bug you for mundane tasks, like changing host entries :-) so it makes definite sense to have a web front end.
- The front end has to be carefully written though, because you have to deal with the following problems:
  - multiple people trying to do an edit at the same time (we conveniently ignore this problem for our sysadmin setup because conflicts would be uncommon, and can easily be manually resolved. That said, we can fix this by having multiple /etc/bind/edit_sysadminlogin directories)
  - more than ever, we cannot just fail and leave a non working DNS or DHCP setup because a user fed bad data to the web form
  - We trust the admins not to put shell commands in the hostfile, or do other nasty tricks like this since they already have root access. Needless to say that whenever you end up running commands that take input from a web form, you have to be very careful
  - A failed change has to be rolled back since no admin will be there to say "doh" and fix the hostfile.

Web front end for users

- The web server runs the hostedit script as a CGI. Because it has to ssh over as a specific user with no permissions on the master name server, we cannot run it setuid, as ssh looks at the home directory for the uid of the process, and not its euid. For that reason, we have a small wrapper in C that is setuid root, and sets both the effective and real uids to a nobody user (which has a home directory and a private ssh key, with the public version on the name server)
- For the perl script, we not only use -w and "use strict" (you really do not want to write any perl code without those), and we also need to specify the -T flag to let perl know that all input in tainted (perl does this by itself for suid scripts, but because here our effective and real uids are equal, perl cannot tell that we changed the uid)
- Our perl script, hostedit, is of course hightly custom and would have to be modified for other network configurations. It has enough cases that it should be easily adaptable but basically you want to retreive the different fields that go in hostfile, and enforce the entry of required data (who owns the machine, what is the function, what os, etc...)
- When you have all the data, it is fed via pipe to 'ssh nameserver "cd /etc/bind/webedit; ./webadd"'. It's a cheap way to network transfert data from the web server to the name server, but it has the advantage of being simple
- webadd is the application that receives the data on the other side and takes care of all the locking. So that we can tell what's happening and do debugging if something bad happened, output is done with ShowTimedMesg, which send the output both to STDOUT (which ends up real time in the user's browser window) and to a logfile (otherwise the user will discard the precious error logging and send you an Email telling you that "it don't work" :-)

Web front end for users

- webadd is run as a remote ssh command and reads its input from the command line. Because it has to run make, which does lots of things as root, it is setuid root. However, webadd is only readable by the pwnobody user (thanks to its home directory permissions), so anyone can't just run it and feed it random input.
- That said, we can't trust that input anyway, so perl is again run in tainted mode, but because only the effective uid is changed, any subsequent perl scripts that would be run would complain about unsafe dependencies (taint violation). For that reason, one of the first things webadd does is set the effective uid to root too, which will make the perl scripts that are subsequently called, happy.
- While programs like hostfiletodhcpanddns do their own locking, we don't want several instances of webadd (and hence make) to be run at the same time, so webadd just grabs an additional global lock for 3mn, and we tell the locker program to remove a stale lock in case one was left behind by a prior process that was badly killed.
- Note that stale locks shouldn't happen because webadd is careful to remove the lock before dying. Because removing stale locks is also an unsafe operation as it's not atomic and two new processes can remove the stale lock and successfully grab a new one, you may want to decide that if a lock is left behind, subsequent runs of webadd will fail until an admin investigates.
- In a nutshell, webadd basically does the following:
  - cvs co, to make sure the hostfile we're editing it up to date
  - we then attempt to add the new host entry by feeding it to hostfiletodhcpanddns
  - After that we do a cvs commit of the new hostfile so that we can run make without any user input
  - If the hostfile was successfully modified, but the subsequent cvs commit or make fail, we restore the old known good hostfile before exiting

Security concerns

- Whenever you have a suid bit or your are running things as root with any possible kind of user input, you should take special care of security concerns (i.e. you should always care about security, but in the above cases, you should be even more vigilent)
- The good thing is that perl does its best so that you don't shoot yourself in the foot. The -T (taint) flag is automatic for suid scripts, and perl doesn't let you disable it ;-)
- The basic rules that taint enforces are as follows:
  - Clean your environment, for instance:
    delete @ENV{qw(IFS CDPATH ENV BASH_ENV)}
  - Never run a "system" command line with any variables that are directly or indirectly linked to external input
  - The only way to untaint a variable is to extract a portion of it with `$taintedvar =~ /(pattern)/; $cleanvar=$1`, but don't just get in the habit to use this to copy and untaint a tainted variable
  - Unless you specifically need shell features (like stderr redirection), just don't use system, built a command line with exec instead. So, instead of `system "ls $tained"`, do `exec "ls", "$tainted"`
  - The same goes for open, `open ("ls $tainted |")` should be replaced with `open ("-|") or exec ("ls $tainted")`
- Man perlsec for more info :-)

URLs

- The talk slides can be found here.
  kpresenter format:
  http://marc.merlins.org/linux/talks/Hostfile/DNS-DHCP-YP.kpr

  HTML:
  http://marc.merlins.org/linux/talks/Hostfile/html/

- the browsable tree with files is here:
  http://marc.merlins.org/linux/talks/Hostfile/etc_bind/

- and the archive is here:
  http://marc.merlins.org/linux/talks/Hostfile/etc_bind.tar.gz