Issues Regarding DNS and Apache

This page could be summarized with the statement: don't require Apache to use DNS for any parsing of the configuration files. If Apache has to use DNS to parse the configuration files then your server may be subject to reliability problems (it might not boot), or denial and theft of service attacks (including users able to steal hits from other users).

Topics

A Simple Example

```
<VirtualHost www.abc.dom>
ServerAdmin webgirl@abc.dom
DocumentRoot /www/abc
</VirtualHost>
```

In order for Apache to function properly it absolutely needs to have two pieces of information about each virtual host: the `ServerName` and at least one IP address that the server responds to. This example does not include the IP address, so Apache must use DNS to find the address of `www.abc.dom`. If for some reason DNS is not available at the time your server is parsing its config file, then this virtual host **will not be configured**. It won't be able to respond to any hits to this virtual host (prior to Apache version 1.2 the server would not even boot).

Suppose that `www.abc.dom` has address 10.0.0.1. Then consider this configuration snippet:

```
<VirtualHost 10.0.0.1>
ServerAdmin webgirl@abc.dom
DocumentRoot /www/abc
</VirtualHost>
```

Now Apache needs to use reverse DNS to find the `ServerName` for this virtualhost. If that reverse lookup fails then it will partially disable the virtualhost (prior to Apache version 1.2 the server would not even boot). If the virtual host is name-based then it will effectively be totally disabled, but if it is IP-based then it will mostly work. However if Apache should ever have to generate a full URL for the server which includes the server name then it will fail to generate a valid URL.

Here is a snippet that avoids both of these problems.

```
<VirtualHost 10.0.0.1>
ServerName www.abc.dom
ServerAdmin webgirl@abc.dom
DocumentRoot /www/abc
</VirtualHost>
```

Denial of Service

There are (at least) two forms that denial of service can come in. If you are running a version of Apache prior to version 1.2 then your server will not even boot if one of the two DNS lookups mentioned above fails for any of your virtual hosts. In some cases this DNS lookup may not even be under your control. For example, if `abc.dom` is one of your customers and they control their own DNS then they can force your (pre-1.2) server to fail while booting simply by deleting the `www.abc.dom` record.

Another form is far more insidious. Consider this configuration snippet:

```
<VirtualHost www.abc.dom>
  ServerAdmin webgirl@abc.dom
  DocumentRoot /www/abc
</VirtualHost>

<VirtualHost www.def.dom>
  ServerAdmin webguy@def.dom
  DocumentRoot /www/def
</VirtualHost>
```

Suppose that you've assigned 10.0.0.1 to `www.abc.dom` and 10.0.0.2 to `www.def.dom`. Furthermore, suppose that `def.dom` has control of their own DNS. With this config you have put `def.dom` into a position where they can steal all traffic destined to `abc.dom`. To do so, all they have to do is set `www.def.dom` to 10.0.0.1. Since they control their own DNS you can't stop them from pointing the `www.def.dom` record wherever they wish.

Requests coming in to 10.0.0.1 (including all those where users typed in URLs of the form `http://www.abc.dom/whatever`) will all be served by the `def.dom` virtual host. To better understand why this happens requires a more in-depth discussion of how Apache matches up incoming requests with the virtual host that will serve it. A rough document describing this is available[1].

The "main server" Address

The addition of name-based virtual host support [2] in Apache 1.1 requires Apache to know the IP address(es) of the host that httpd is running on. To get this address it uses either the global `ServerName` (if present) or calls the C function `gethostname` (which should return the same as typing "hostname" at the command prompt). Then it performs a DNS lookup on this address. At present there is no way to avoid this lookup.

If you fear that this lookup might fail because your DNS server is down then you can insert the hostname in `/etc/hosts` (where you probably already have it so that the machine can boot properly). Then ensure that your machine is configured to use `/etc/hosts` in the event that DNS fails. Depending on what OS you are using this might be accomplished by editing `/etc/resolv.conf`, or maybe `/etc/nsswitch.conf`.

If your server doesn't have to perform DNS for any other reason then you might be able to get away with running Apache with the `HOSTRESORDER` environment variable set to "local". This all depends on what OS and resolver libraries you are using. It also affects CGIs unless you use `mod_env` to control the environment. It's best to consult the man pages or FAQs for your OS.

Tips to Avoid These Problems

- use IP addresses in `VirtualHost`
- use IP addresses in `Listen`
- ensure all virtual hosts have an explicit `ServerName`

- create a `<VirtualHost _default_:*>` server that has no pages to serve

Appendix: Future Directions

The situation regarding DNS is highly undesirable. For Apache 1.2 we've attempted to make the server at least continue booting in the event of failed DNS, but it might not be the best we can do. In any event requiring the use of explicit IP addresses in configuration files is highly undesirable in today's Internet where renumbering is a necessity.

A possible work around to the theft of service attack described above would be to perform a reverse DNS lookup on the ip address returned by the forward lookup and compare the two names. In the event of a mismatch the virtualhost would be disabled. This would require reverse DNS to be configured properly (which is something that most admins are familiar with because of the common use of "double-reverse" DNS lookups by FTP servers and TCP wrappers).

In any event it doesn't seem possible to reliably boot a virtual-hosted web server when DNS has failed unless IP addresses are used. Partial solutions such as disabling portions of the configuration might be worse than not booting at all depending on what the webserver is supposed to accomplish.

As HTTP/1.1 is deployed and browsers and proxies start issuing the `Host` header it will become possible to avoid the use of IP-based virtual hosts entirely. In this event a webserver has no requirement to do DNS lookups during configuration. But as of March 1997 these features have not been deployed widely enough to be put into use on critical webservers.

URI References

[1] http://httpd.apache.org/docs-2.1/vhosts/details.html
[2] http://httpd.apache.org/docs-2.1/vhosts/name-based.html