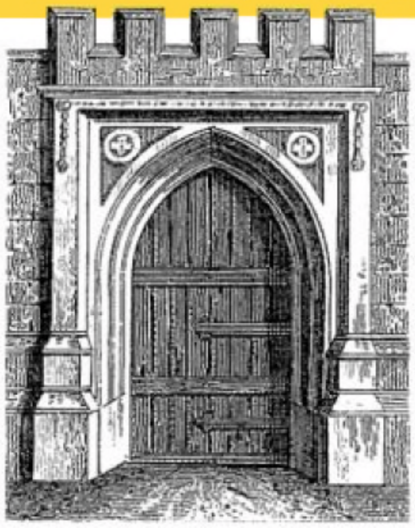
Building Internet

Firewalls



Building Internet Firewalls

Elizabeth D. Zwicky, Simon Cooper & D. Brent Chapman

Second Edition, June 2000

ISBN: 1-56592-871-7, 890 pages

Completely revised and much expanded, the new edition of the highly respected and bestselling Building Internet Firewalls now covers Unix, Linux, and Windows NT.

This practical and detailed guide explains in step-by-step fashion how to design and install firewalls and configure Internet services to work with a firewall.

It covers a wide range of services and protocols and offers a complete list of resources, including the location of many publicly available firewalls construction tools.

Release Team[oR] 2001

20.1.4 DNS Security Problems

Some security problems with DNS are described in the following sections.

20.1.4.1 Bogus answers to DNS queries

The first security problem with DNS is that many DNS servers and clients can be tricked by an attacker into believing bogus information. Many clients and servers don't check to see whether all the answers they get relate to questions they actually asked, or whether the answers they get are coming from the server they asked. A machine that asks for the IP address of "malicioushost" and gets back the requested information plus a false IP address for "trustedhost", as well, may cache the extra answer without really thinking about it and answer later queries with this bogus cached data. This lack of checking can allow an attacker to give false data to your clients and servers. For example, an attacker could use this capability to load your server's cache with information that says that the IP address of a machine the attacker controls maps to the hostname of a host you trust for password-less access via *rlogin*. (This reason is only one of several why you shouldn't allow the BSD "r" commands across your firewall; see the full discussion of these commands in Chapter 18.)

Some Unix DNS implementations will accept and cache answers even when they haven't made a query; some Microsoft implementations will crash if they receive an unrequested answer. Both of these behaviors are undesirable and have been eliminated by recent releases. Windows 2000 by default will only accept answers to queries but will accept those answers from any server. It can be configured to require the response to come from a queried server and should be on security-critical machines.

> Later versions of DNS for Unix (BIND 4.9 and later) check for bogus answers and are less susceptible to these problems. Earlier versions, and DNS clients and servers for other platforms, may still be susceptible.

20.1.4.2 Malicious DNS queries

Not only are some DNS implementations vulnerable to hostile answers, some are vulnerable to hostile questions. In particular, some DNS servers have problems with buffer overflows and may crash or execute hostile code if a query is too long. (See Chapter 13, for more information about buffer overflow attacks.) If your log files show queries that contain very long "hostnames" containing control characters, people are probably attempting buffer overflow attacks. Again, vendors have been working on eliminating these vulnerabilities, but you should check to make certain the appropriate patches have been made in the version you are running. There are known problems with versions of BIND 4 prior to 4.9.7 and BIND 8 prior to 8.1.2 (note that this does not guarantee that later versions don't also have problems that haven't been found yet).

20.1.4.3 Mismatched data between the hostname and IP address DNS trees

The attack that uses bad cached data to give you an apparently trustworthy hostname for an untrusted host points out the problem of mismatched data between the hostname and IP address trees in DNS. In a case like the one we've described, if you look up the hostname corresponding to the attacker's IP address (this is called a *reverse lookup*), you get back the name of a host you trust. If you then look up the IP address of this hostname (which is called a *double-reverse lookup*), you should see that the IP address doesn't match the one the attacker is using, which should alert you that something suspicious is going on. Reverse and double-reverse lookups are described in more detail later in this DNS discussion.

There are perfectly valid reasons for these checks to return inconsistent values; no rules require a forward and reverse lookup to return consistent information. In fact, when DNS is used for load balancing between servers, it is difficult to arrange for this consistency. In these situations, DNS is being used to determine the location of a service rather than the IP address of an individual host.

Any program that makes authentication or authorization decisions based on the hostname information it gets from DNS should be very careful to validate the data with this reverse lookup/double-reverse lookup method. In some operating systems (for example, SunOS 4.x and later), this check is automatically done for you by the *gethostbyaddr( )* library function. In most other operating systems, you have to do the check yourself. Make sure that you know which approach your own operating system takes and that the daemons that are making such decisions in your system do the appropriate validation. (And be sure you're preserving this functionality if you modify or replace the vendor's *libc*.)

Better yet, don't do any authentication or authorization based solely on hostname or even on IP address; there is no way to be sure that a packet comes from the IP address it claims to come from, unless some kind of cryptographic authentication is within the packet that only the true source could have generated.

Some implementations of double-reverse lookup fail on hosts with multiple addresses, (e.g., dual-homed hosts used for proxying). If both addresses are registered at the same name, a DNS lookup by name will return both of them, but many programs will read only the first. If the connection happened to come from the second address, the double-reverse will incorrectly fail even though the host is correctly registered. Although you should avoid using double-reverse implementations that have this flaw, you may also want to ensure that on your externally visible multi-homed hosts, lookup by address returns a different name for each address, and that those names have only one address returned when it is looked up. For example, for a host named "foo" with two interfaces named "e0" and "e1", have lookups of "foo" return both addresses, lookups of "foo-e0" and "foo-e1" return only the address of that interface, and lookups by IP address return "foo-e0" or "foo-e1" (but not simply "foo") as appropriate.

For internal multi-homed hosts, you probably don't want to set things up in the way we've described; if you do, you may end up needing to list them by multiple names anywhere you want to give them permissions, such as in */etc/exports* files.

20.1.4.4 Dynamic update

It's very convenient for clients to be able to update DNS servers. For instance, at sites that use DHCP to dynamically assign addresses to computers, dynamic updates allow clients to have consistent names. When a client gets an address from DHCP, it can then register that address with the name server under the client's usual name. There is a standard for dynamic updates of DNS, but it isn't very widely used because it provides no kind of authentication. Some servers do provide authentication methods for dynamic updates (for instance, Windows 2000 allows you to integrate DNS with Active Directory and use Kerberos to authenticate update requests), but there is no widespread and interoperable standard for this.

Without authentication, dynamic update of DNS is extremely risky. You can't keep hostile clients from stealing addresses from each other or swamping the server in changes. Therefore, dynamic updates in DNS can be used only inside networks where there is a very high degree of trust.

20.1.4.5 Revealing too much information to attackers

Another problem you may encounter when supporting DNS with a firewall is that it may reveal information that you don't want revealed. Some organizations view internal hostnames (as well as other information about internal hosts) as confidential information. They want to protect these hostnames much as they do their internal telephone directories. They're nervous because internal hostnames may reveal project names or other product intelligence, or because these names may reveal the type of the hosts (which may make an attack easier). For example, it's easy to guess what kind of system something is if its name is "lab-sun" or "cisco-gw".

Even the simplest hostname information can be helpful to an attacker who wants to bluff his or her way into your site, physically or electronically. Using information in this way is an example of what is commonly called a *social engineering* attack. The attacker first examines the DNS data to determine the name of a key host or hosts at your site. Such hosts will often be listed as DNS servers for the domain or as MX gateways for lots of other hosts. Next, the attacker calls or visits your site, posing as a service technician, and claims to need to work on these hosts. The attacker will then ask for the passwords for the hosts (on the telephone) or ask to be shown to the machine room (in person). Because the attacker seems legitimate and seems to have inside information about the site - after all, the machine names are right - people will often grant access. Social engineering attacks like this takes a lot of brazenness on the part of the attacker, particularly if they're carried out in person, but you'd be amazed at how often such attacks succeed.

Besides internal hostnames, other information is often placed within the DNS - information that is useful locally but you'd really rather an attacker not have access to. DNS HINFO and TXT resource records are particularly revealing:

HINFO (host information records)

These name the hardware and operating system release that a machine is running: it's very useful information for system and network administrators but also tells an attacker exactly which list of bugs to try on that machine.

TXT (textual information records)

These are essentially short unformatted text records used by a variety of different services to provide various information. For example, some versions of Kerberos and related tools use these records to store information that, at another site, might be handled by NIS.

Attackers will often obtain DNS information about your site wholesale by contacting your DNS server and asking for a zone transfer, as if they were a secondary server for your site. You can prevent this either with packet filtering (by blocking TCP-based DNS queries, which will block more than just zone transfers) or by configuring your DNS server to control which hosts it is willing to do zone transfers to (although it will use the IP address to validate hosts and will be vulnerable to IP spoofing). The various versions of BIND control this in different ways (consult your BIND documentation for more information), while the Microsoft DNS server allows you to specify that only hosts that receive notifications may do zone transfers.

The question to keep in mind when considering what DNS data to reveal is, "Why give attackers any more information than necessary?" The following sections provide some suggestions to help you reveal only the data you want people to have.

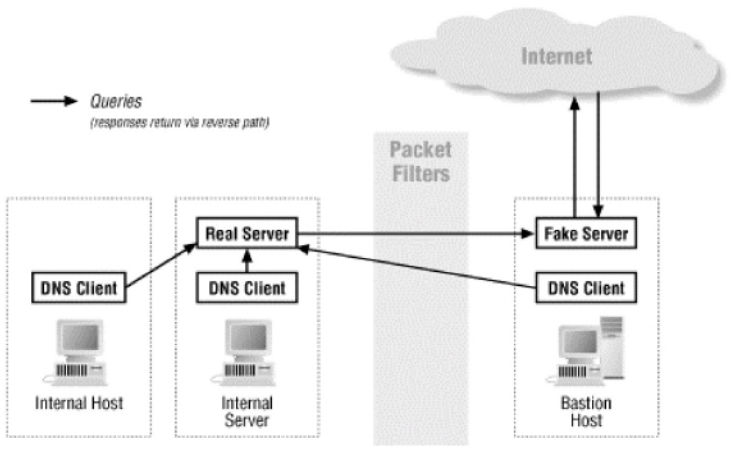
20.1.5 Setting Up DNS to Hide Information, Without Subdomains

We've mentioned that DNS has a query-forwarding capability. By taking advantage of this capability, you can give internal hosts an unrestricted view of both internal and external DNS data, while restricting external hosts to a very limited ("sanitized") view of internal DNS data. You might want to do this for such reasons as:

- Your internal DNS data is too sensitive to show to everybody.
- You know that your internal DNS servers don't all work perfectly, and you want a better-maintained view for the outside world.
- You want to give certain information to external hosts and different information to internal hosts (for example, you want internal hosts to send mail directly to internal machines but external hosts to see an MX record directing the mail to a bastion host).

Figure 20.3 shows one way to set up DNS to hide information; the following sections describe all the details. This mechanism will work only for sites that use a single domain (all hosts are named something like *host.foo.example*, rather than *host.sillywalks.foo.example* and *host.engineering.foo.example*). If you have subdomains, you will need a more complicated configuration, which we discuss in the next section.

Figure 20.3. A firewall can be used to hide DNS information


Internet
Queries
(responses return via reverse path)
Packet Filters
Real Server
Fake Server
DNS Client
DNS Client
DNS Client
Internal Host
Internal Server
Bastion Host