HOWTO - Jeanne, redirector for Squid Reverse Proxy

Vincent Berk (c)2001 vberk@ists.dartmouth.edu GNU/GPL
Marion Bates (c) 2001 mbates@ists.dartmouth.edu
Installation Guide with Examples (ALPHA distribution)

1. What is this? Why would I want it?

This document describes how you can protect your webserver against certain nasty vulnerabilities using the Jeanne Reverse Proxy program. Its value was proven with the recent IIS unicode attacks, Code Red/II/Blue and NIMDA. We were shielded from all these before we even knew about their existence. Why is it called `jeanne`? Ask Chris Brenton.

2. Then what does it do?

The webserver is placed behind your firewall, and the firewall only allows `jeanne` to communicate with it. `Jeanne` will verify each HTTP request before passing it on to the actual webserver.

3. Installation

This is a step by step installation of the concept described above. For this document, a general network setup is assumed, with a firewall in front of the webserver. It does not matter whether you have a DMZ/localnet with a dedicated firewall, or simply a hostbased firewall on the webserver, as long as you know how to work with it. See the two examples at the end of this document.

3.1. Get a box

Hardware requirements are fairly simple. Anything capable of running UNIX/Linux should be fine. The proxy has been used successfully under RedHat Linux and Debian Linux (on Intel hardware) and Solaris (on Sparc). It is recommended that you use a machine similar to your webserver to simplify maintenance.

3.2. Install the Operating System

Anything that can run Squid and has a working C compiler will work. An out-of-the-box installation of Linux will usually meet this requirement, as will Solaris. In our test installation, we

used a PC with an Intel Pentium III running at 600MHz with 128 Mb of RAM and a 9 GB 10,000RPM UW SCSI hard disk. We installed Debian Linux 2.2 release 3 and upgraded to the 2.2.4 kernel. We installed only the base system plus the latest version of OpenSSH.

3.3 Make it Secure

Shut down all services, except possibly an up-to-date version of sshd (Secure Shell daemon). Refer to the SANS document "Securing Linux (or Solaris) Step-by-Step" or a similar how-to for detailed information on how to shut down internet services. SECURING THIS BOX IS CRUCIAL! Unneeded/unpatched services are the easiest way for servers to become compromised by an attacker. Some commonly-vulnerable services include SunRPC things like `portmapper`, programs launched by `inetd, lpd, Xwindows, telnetd,` and `ftpd`.

If you run a portmapper such as `nmap` against this system, you should find no ports open except those which you know you definitely want, like SSH. Also make sure that any installed webservers are shut down, and disable these processes from starting up again when the box is rebooted.

3.4. Obtain and Install Squid

Download the Squid source tarball. Refer to the highly-comprehensive Squid documentation for installation details. Basically:

```
ftp://ftp.squid-cache.org
cd pub/squid-2/STABLE/
get squid-2.4.STABLE2-src.tar.gz (or whatever's the latest version)

tar -xvzf squid-2.4.STABLE2-src.tar.gz
cd squid-2.4.STABLE2

./configure --prefix=/usr (unless you wanted it in /usr/local, which is
default)
make
```

(become `root`)

```
make install
```

3.5. Configure Squid

This is where it gets a little more complicated. By default, Squid is set up to be a normal webcache, yet we want it to work as a reverse proxy. To configure Squid, edit the file: `/etc/squid.conf`.

Take a look at the following example configuration:

```
# squid.conf - basic httpd reverse proxy server configuration

# Make reverse proxy listen on port 80 like normal webserver.
# Hostname will be www.
http_port 80
visible_hostname www.my-domain.com

# The IP address and port number of the real webserver located behind
# the firewall.
httpd_accel_host 10.20.30.4
httpd_accel_port 80

# Turn off the original proxy server.
httpd_accel_with_proxy off

# Configure the cache to be valid for 30 seconds.
refresh_pattern . 0 0% 30

# The Access Control Lists
# Everyone is allowed to use the GET method on the HTTP port 80.
# For this example, assume your local net is 10.20.30.0.
# All other methods and ports are denied.
# For a more accurate description see the Squid documentation.
acl all src 0.0.0.0/0.0.0.0
acl localhost src 127.0.0.1/255.255.255.255
acl localnet  src 10.20.30.0/255.255.255.0
acl safeports port 80
acl safemethods method GET
http_access deny !safeports
http_access deny !safemethods
http_access allow all

# We'll get to this later on. Must be correct path to jeanne.
redirect_program /usr/mrp/jeanne

# Make sure we never bypass the redirector
redirector_bypass off

# Give us 5 children
redirect_children 5

# Make cache directories here:
# use the ufs filetype (Squid likes it), make the max cache size
# 1024 Mb, allow 16 level 1 cache subdirectories, and allow 256
# level 2 cache subdirectories (meaning that each of the 16
# level 1 directories can have 256 directories within itself.)
cache_dir ufs /var/squid/cache 1024 16 256

# End of squid.conf reverse proxy server configuration
```

Make sure your original webserver (`wwwfiles` in our case) reports as its hostname `www` (the name of the MRP). This will prevent the real webserver from building redirect requests with its new hostname (i.e. redirect to: `wwwfiles.your-domain.com/index.html`), which is now behind the firewall. Most webservers have this as an option (`ServerName` in Apache).

We installed an edited skeleton script into `/etc/rc.d/init.d` and made the proper runlevels' start and kill symlinks to that script. Squid is started with the flags `-D -sYC`.

`-D` = disable initial DNS tests
`-s` = enable syslog
`-Y` = Only return `UDP_HIT` or `UDP_MISS_NOFETCH` during fast reload
`-C` = Do not catch fatal signals. (Can't be killed/won't crash.)

Create `/var/squid/cache` (or whatever you want to call it, as long as it matches the `cache_dir` line in `squid.conf`) and `chown nobody:nogroup` it. Then run `squid -z` to create the cache directory.

Make a directory, such as `/usr/mrp`, where backups of scripts and the redirector program will be stored. Make sure the following line in `squid.conf` contains the right path to where you will put `jeanne`:

```
redirect_program /usr/mrp/jeanne
```

3.6. Compile/configure `Jeanne`

Put `jeanne.c` in `/usr/mrp`. Edit the `jeanne` makefile, if needed. Currently there is only one useful compile option (define):

```
-DMS_IGN_CASE
```

Depending on the OS of your webserver, case of filenames is treated differently. In Unix, 'TEST' and 'test' are two different filenames, while in Windows, they're the same. With this option you can force `jeanne` to ignore case. You would want to enable this if your webserver is Windows-based (IIS).

Edit `jeanne.c` such that `DEF_URL_FILE` variable points to your allowed URLs file and `DEF_REJECT_URL` is set to the webpage you want to use as your reject page. (If the reject page you define here does not exist, users will get a generic "`error 404: Not found`" error if they request invalid URLs.)

Compile `jeanne.c` by typing "`make`".

Put the `getlist` script (and its cronjob script) into `/usr/mrp`. You will need to modify some variable definitions such that paths to files are defined properly. Modify `getlist` such that:

```
HOME=/usr/mrp
URLSFILE=$HOME/urls
```

and set the IP address be the IP of the actual webserver. Edit `getlist.cron` such that:

`MAILTO` is set to what you want and
`HOME=/usr/mrp`

3.7. Build the URLs File

Install makeurls and its cronjob on the webserver and edit the path definitions like you did with getlist. If you want the URLs list on the webserver to be updated more or less often, alter the crontab appropriately. Keep in mind though that the `find` function is pretty processor-intensive, so you may not want to run it too often. Make sure to add the cron job to the root crontab (`crontab makeurls.cron`).

The URLs file has a very simple setup and syntax. All lines starting with a pound (`#`) are ignored as comments. All other lines are the valid URLs that are passed on to the webserver. In this file you put only those URLs that are to be retrieved from the original webserver. The program will only verify the filename, and will add the hostname later.

If the URL ends with a question mark (`?`), then everything that comes in with that request is copied to the request made by the reverse proxy server to the webserver. This is used as input for CGI scripts, like search requests.

Example:

```
# URLs file for my webserver wwwfiles.domain.com
# The reverse proxy will be called www.domain.com
#
# General files
http://wwwfiles.domain.com/
http://wwwfiles.domain.com/index.html
http://wwwfiles.domain.com/nextpage.html
http://wwwfiles.domain.com/somethingelse.html
http://wwwfiles.domain.com/mypicture.gif
http://wwwfiles.domain.com/myotherpicture.jpg
#
# My search cgi:
http://wwwfiles.domain.com/search.cgi?
```

3.8. Modify DNS Records

You will need to change your DNS server(s) such that all requests made to "`www.domain.com`" go to the IP of the reverse proxy server running `jeanne`. The original webserver can be called "`wwwfiles.domain.com`," as in the following examples.

If you have your own DNS you can make this modification yourself. If your ISP has control over your DNS records, then probably the easiest method is to change the IP address of your reverse proxy to be that of your original webserver. Then change the IP of your webserver to that of "`wwwfiles.domain.com`." In other words:

Old setup: DNS thinks that www.domain.com -> 12.34.56.78.

New setup: **Assign 12.34.56.78 to proxy.** DNS still thinks that **www.domain.com -> 12.34.56.78** and we make actual webserver have different hostname and IP, which DNS doesn't need to know (and we don't even want it to know!): **wwwfiles.domain.com -> 12.34.56.89 (webserver)**

3.9. Modify Firewall Rules

Modify your firewall rules to reflect your new setup. The original webserver, now named "`wwwfiles`," can only be contacted by "`www`," the new reverse proxy. "`www`" can be reached by the whole world, but only via the HTTP port (80).

3.10. Test Everything

Restart the reverse proxy server with the redirector. Make some requests with your favorite browser:

```
http://www.domain.com/
```

Do a search:

```
http://www.domain.com/search.cgi?my_search_terms
```

Try an exploit or two:

```
http://www.domain.com/../../../../etc/passwd
http://www.domain.com/scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir
```

(These won't work, even if they did before! :)

4. Examples

a. Local webserver hosted behind an ISP; DNS provided by ISP

Network Layout BEFORE Reverse Proxy:

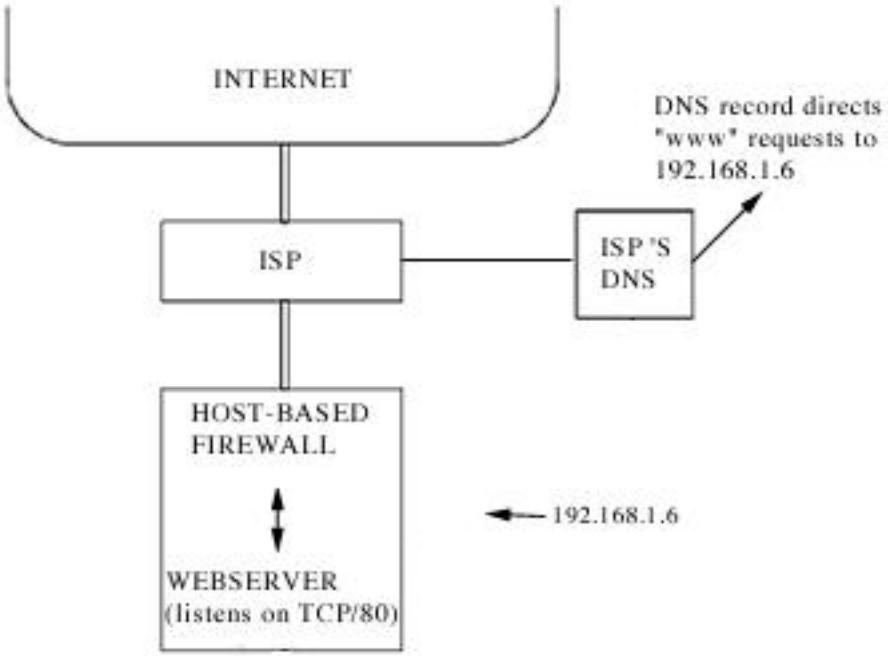

Fig. 1: Typical Network Layout

Firewall Rules BEFORE Reverse Proxy:

SOURCE	DESTINATION	SERVICE	ACTION
Internet	WWW	HTTP	Accept
ANY	ANY	ANY	Deny

Fig. 2: Typical Firewall Rulebase

Requests to www go directly to the webserver/firewall. DNS records are handled by the ISP.

Network Layout AFTER Reverse Proxy:

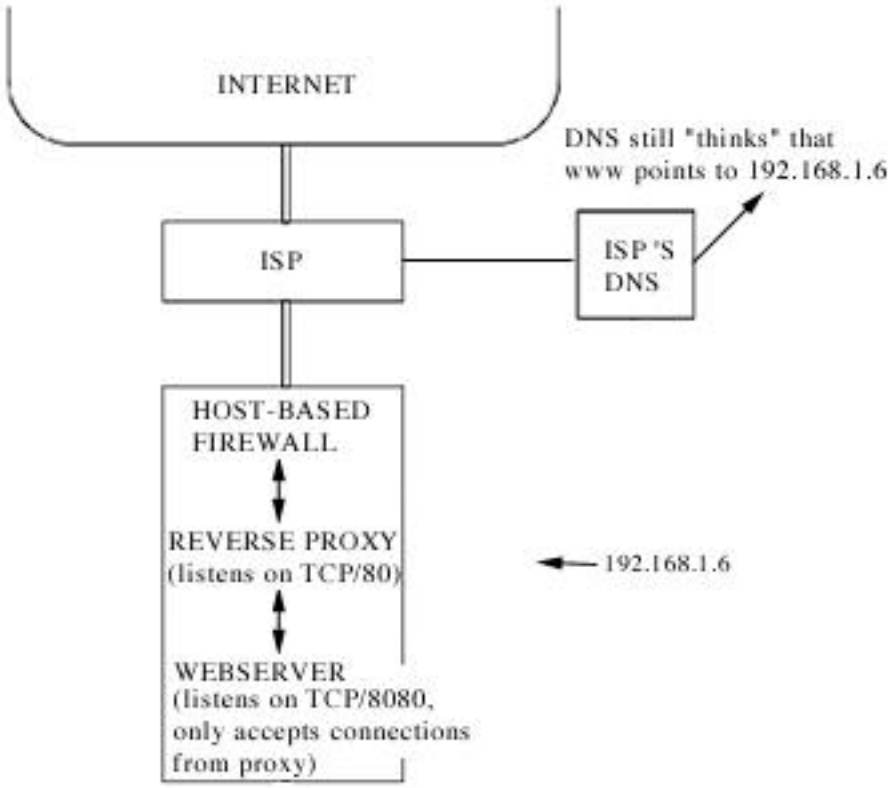

Fig. 3: Revised Network Layout

Firewall Rules AFTER Reverse Proxy:

SOURCE	DESTINATION	SERVICE	ACTION
Internet	Reverse Proxy	HTTP	Accept
ANY	ANY	ANY	Deny

It would also be possible to have a network setup similar to above, but with a separate machine as the reverse proxy, sitting in front of the webserver/host-based firewall. Ideally, the proxy machine would have two network cards, one facing the ISP and the other dedicated to communicating with the webserver.

b. Network with dedicated firewall and its own DNS servers

Network Layout BEFORE Reverse Proxy:

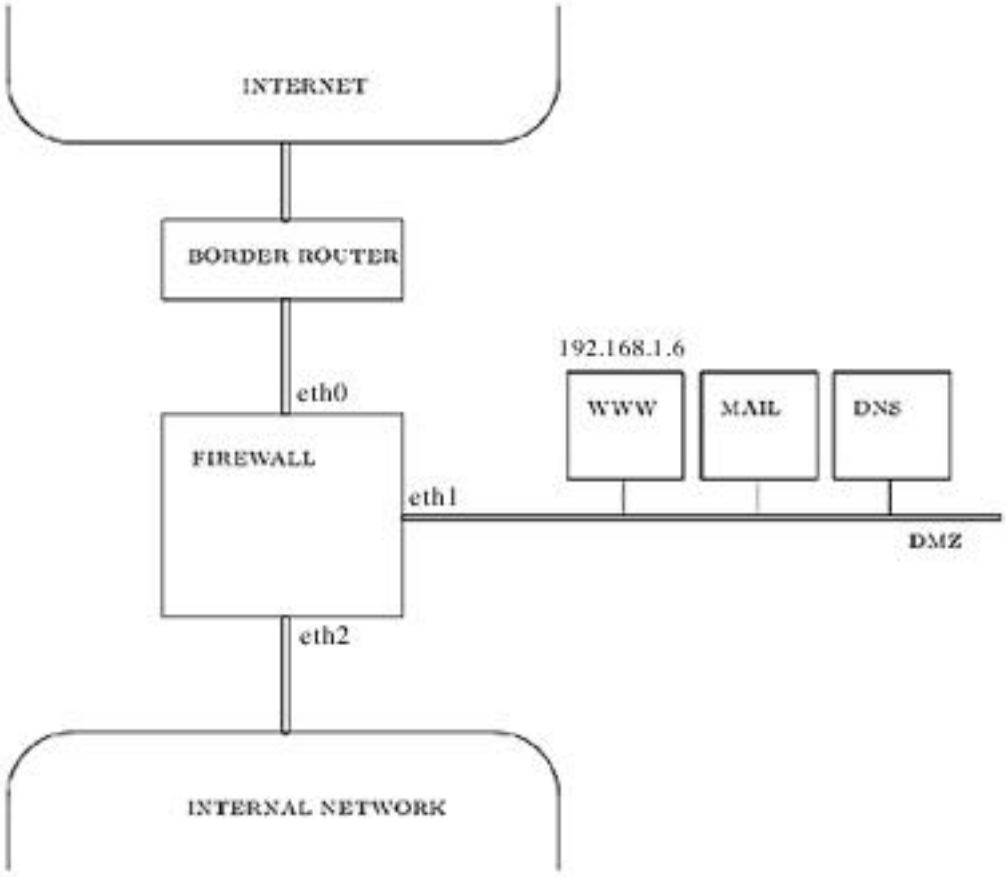

Fig. 5: Typical Network Layout

Firewall Rules BEFORE Reverse Proxy:

SOURCE	DESTINATION	SERVICE	ACTION
Internet	WWW	HTTP	Accept
Internet	Mail	SMTP/IMAP/POP	Accept
Internet	DNS	DNS	Accept
Internet	Local Net	ANY	Deny
DMZ	Local Net	ANY	Deny
Local Net	Internet	ANY	Accept
Local Net	DMZ	ANY	Accept
ANY	ANY	ANY	Deny

Fig. 6: Typical Firewall Rulebase

The webserver, `www`, sits on the relatively-unprotected DMZ. Requests for "`www.domain.com`" are passed directly to the webserver. DNS entry for `www` points to IP address of webserver, `192.168.1.6`.

Network Layout AFTER Reverse Proxy:

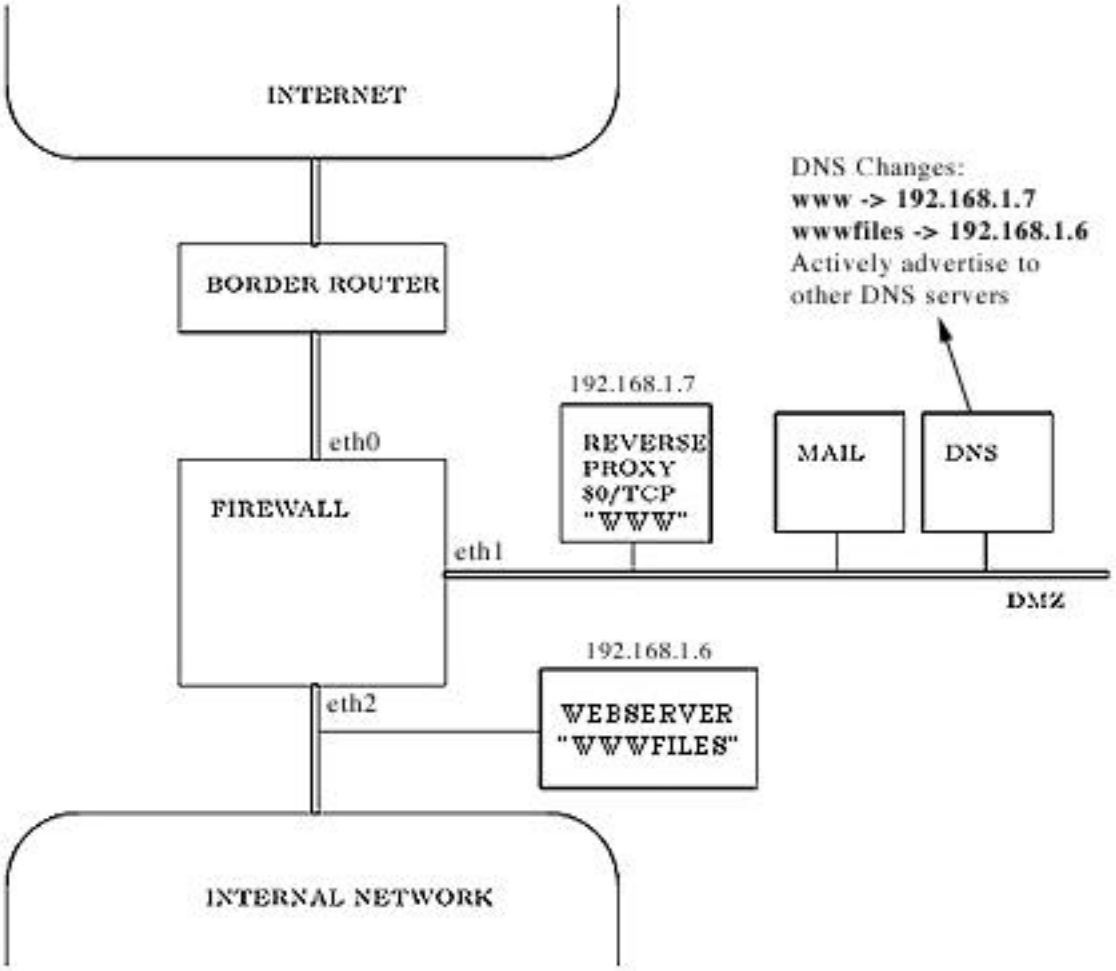

Fig. 7: Revised Network Layout

Firewall Rules AFTER Reverse Proxy:

SOURCE	DESTINATION	SERVICE	ACTION
Internet	Reverse Proxy	HTTP	Accept
Reverse Proxy	WWW	HTTP	Accept
Internet	Mail	SMTP/IMAP/POP	Accept
Internet	DNS	DNS	Accept
Internet	Local Net	ANY	Deny
DMZ	Local Net	ANY	Deny
Local Net	Internet	ANY	Accept
Local Net	DMZ	ANY	Accept
ANY	ANY	ANY	Deny

Fig. 8: Revised Firewall Rulebase

Requests from the outside world for "www.domain.com" are directed to the proxy. Only the proxy can connect to the webserver ("wwwfiles").

END