Configuring HTTPD with Relayd as a Reverse Proxy

Here's a fairly simple pair of configurations to enable **httpd** to serve web pages with **relayd** acting as a reverse proxy.

The main purpose of **relayd** here is to control access to certain parts of a web site. In this example, both httpd and relayd are on the **same host**.

First of all, enable the daemons.

```
# rcctl enable httpd
# rcctl enable relayd
```

/etc/httpd.conf

```
# $OpenBSD: httpd.conf,v 1.22 2020/11/04 10:34:18 denis Exp $
server "chunkymonkey.tld" {
#       listen on 192.168.0.10 tls port 443


        listen on 127.0.0.1 port 81


#       tls {
#               certificate "/etc/letsencrypt/live/chunkymonkey.tld/fullchain.pem"
#               key "/etc/letsencrypt/live/chunkymonkey.tld/privkey.pem"
#       }
#       location "*.php" {
#       fastcgi socket "/run/php-fpm.sock"
#       }
#       location "/*.php[/?]*" {
#        fastcgi socket "/run/php-fpm.sock"
#       }


        root "htdocs/chunkymonkey"
        directory index index.html


        errdocs "htdocs/errorpages"


        log access "access.log"
        log error "error.log"
        log style forwarded
}


# Include additional MIME types
types {
        include "/usr/share/misc/mime.types"
}
```

You can see above that the **TLS** sections of **`/etc/httpd.conf`** are commented out. This is because **relayd** will handle all encrypted connections to the web site, passing requests to **httpd**.

In these configurations, external clients connect to port 443 (TLS/SSL) and then **relayd** forwards the connection to **httpd** listening on port 81.

/etc/relayd.conf

```
# $OpenBSD: relayd.conf,v 1.5 2018/05/06 20:56:55 benno Exp $
#
# Macros
#
ext_addr="192.168.0.10"
webhost1="127.0.0.1"


log state changes
log connection


table <webhosts> { $webhost1 }
table <fallback> { 127.0.0.1 }


http protocol https {
        match request header set "X-Forwarded-For" value "$REMOTE_ADDR"
        match request header set "X-Forwarded-Port" value "$REMOTE_PORT"
        tcp { sack, backlog 128 }
        tls keypair "chunkymonkey.tld"
        #block request url "chunkymonkey.tld/testblock/"
        #pass request from 192.168.0.50 url "chunkymonkey.tld/testblock/"
}


relay wwwtls {
        # Run as an SSL/TLS accelerator
        listen on $ext_addr port 443 tls
        protocol https
        # Forward to hosts in the webhosts table using a src/dst hash
        forward to <webhosts> port 81 mode loadbalance \
                check http "/" code 200
}
```

In the commented access control section of the **relayd** configuration above, requests made to **https://chunkymonkey.tld/testblock** would be blocked, except if the requesting IP address is **`192.168.0.50`**. There are no error pages shown when this happens and the web browser involved will normally return an error stating something along the lines of the connection being dropped.

If you're using **Let's Encrypt** to generate TLS certificates then I've found that **relayd** only seems to like **RSA**.

```
# certbot --key-type rsa -d chunkymonkey.tld -d something.chunkymonkey.tld certonly
```

Once the certificates have been installed, creating symlinks in **`/etc/ssl`** and **`/etc/ssl/private`** seems to do the trick for **relayd**.

In **`/etc/ssl`**:

```
# ln -s /etc/letsencrypt/live/chunkymonkey.tld/fullchain.pem chunkymonkey.tld.crt
```

In **`/etc/ssl/private`**:

```
# ln -s /etc/letsencrypt/live/chunkymonkey.tld/privkey.pem chunkymonkey.tld.key
```

These certificate locations are referenced by:

```
tls keypair "chunkymonkey.tld"
```

... in **`/etc/relayd.conf`**.

You can always check your configuration files before restarting the servers using the following commands.

```
# httpd -n
# relayd -n
```

Thanks for reading.

Updated: 2023-09-07
Chosen OS: OpenBSD 7.3