PSAwise2324Team10Aufgabe04

Setting Up a Web Server

VM5 is our webserver.

We chose to use Nginx as a reverse proxy and for serving static websites, in combination with Apache for dynamic content.

To test the websites from the terminal, use `curl -L --noproxy "*" <url>`. The L flag makes curl follow any http redirects automatically while the --no-proxy flag makes it ignore the TUM proxy when sending requests.

DNS

Give VM5 a second IP address in the netplan configuration, then add records for the second IP address and the CNAME www.psa-team10.cit.tum.de l to the zone file of the primary zone:

```
vmpsateam10-05  IN      A       192.168.10.5
vm5-other       IN      A       192.168.10.201 //second IP address

www             IN      CNAME   vmpsateam10-05 // www.psa-team10.cit.tum.de
```

("www" is effectively "www.psa-team10.cit.tum.de" because the domain name is appended to every name that doesn't end in a period.)

Nginx

Install, enable and start Nginx. Nginx will listen on the ports 80 and 443. However, only https traffic will actually be accepted. http traffic will be redirected to the https version of the address. Use `sudo systemctl restart nginx` whenever you want to apply configuration changes.

We set up the following domains:

- vmpsateam10-05.psa-team10.cit.tum.de
- www.psa-team10.cit.tum.de (CNAME)
- vm5-other.psa-team10.cit.tum.de (second IP address)

Open /wetc/nginx/sites-available/default. To redirect all http requests to https, add an entry like this for every URL:

```
# redirect http requests to https
server {
    listen 80; # listen for http

    server_name www.psa-team10.cit.tum.de;

    return 301 https://$server_name$request_uri; # "moved permanently" status
code, which prompts a redirect
}
```

Next, you have to add the configurations for https. Add an entry like this for every URL.

```
server {
        listen  443; #listen for https

        server_name www.psa-team10.cit.tum.de;

        root /var/www/www_html; # html of the website

        index index.html index.htm index.nginx-debian.html;

        location / {
                # First attempt to serve request as file, then
                # as directory, then fall back to displaying a 404.
                try_files $uri $uri/ =404;
        }
}
```

Since vm5-other.psa-team10.cit.tum.de uses a different IP address, the configuration requires an alteration. Specify the IP address in the listen command, like so, in every server block related to this URL:

```
listen 192.168.10.201:443;
```

SSL

To allow HTTPS traffic, we need certificates.

First of all, install `openssl` using apt. This is the tool we will be using for the process.
Navigate to the /etc/ssl directory.
We need our own CA (certificate authority) in order to be able to add it to the trusted certificate list of a machine later, so create a key and certificate for the CA first:

```
#CA key
sudo openssl genrsa -out /etc/ssl/private/CA.key 4096

# CA cert
sudo openssl req -x509 -new -nodes -key /etc/ssl/private/CA.key -sha256 -days
1024 -out /usr/local/share/ca-certificates/CA.crt -subj
"/C=DE/ST=Bavaria/L=Munich/O=TEAM10/OU=CA/CN=vmpsateam10-05.psa-
team10.cit.tum.de"
```

Then, create individual private keys for each domain our webserver uses:

```
sudo openssl genrsa -out /etc/ssl/private/vmpsa.key 2048

sudo openssl genrsa -out /etc/ssl/private/www.key 2048

sudo openssl genrsa -out /etc/ssl/private/otherIP.key 2048
```

Next step is to create certificate signing requests using these keys. You can also provide some general information about the certificate. Most of these fields are voluntary, but make sure you specify the CN (common name). It needs to be the actual resolvable FQDN of the domain.

```
sudo openssl req -new -sha256 -key /etc/ssl/private/vmpsa.key -subj
"/C=DE/ST=Bavaria/L=Munich/O=TUM/OU=PSA/CN=vmpsateam10-05.psa-
team10.cit.tum.de" -out vmpsa.csr

sudo openssl req -new -sha256 -key /etc/ssl/private/www.key -subj
"/C=DE/ST=Bavaria/L=Munich/O=TUM/OU=PSA/CN=www.psa-team10.cit.tum.de" -out
www.csr

sudo openssl req -new -sha256 -key /etc/ssl/private/otherIP.key -subj
"/C=DE/ST=Bavaria/L=Munich/O=TUM/OU=PSA/CN=vm5-other.psa-team10.cit.tum.de" -
out otherIP.csr
```

Then, you can move onto signing the certificate signing requests using the CA key:

```
sudo openssl x509 -req -in vmpsa.csr -CA /usr/local/share/CA.crt -CAkey
/etc/ssl/private/CA.key -CAcreateserial -out /etc/ssl/certs/vmpsa.crt -days
500 -sha256

sudo openssl x509 -req -in www.csr -CA /usr/local/share/CA.crt -CAkey
/etc/ssl/private/CA.key -CAcreateserial -out /etc/ssl/certs/www.crt -days 500
-sha256

sudo openssl x509 -req -in otherIP.csr -CA /usr/local/share/CA.crt -CAkey
/etc/ssl/private/CA.key -CAcreateserial -out /etc/ssl/certs/otherIP.crt -days
500 -sha256
```

Finally, the files have to be specified in the Nginx configuration, which handles all HTTPS requests. Add these lines to each port 443 server block in /etc/nginx/sites-available/default:

```
SSLCertificateFile /etc/ssl/certs/<domain>.crt
SSLCertificateKeyFile /etc/ssl/private/<domain>.key
```

and change each `listen 443;` to `listen 443 ssl;`
At this point, the self-signed certificates are established but will not be trusted by the machine. To add them to the machine's trusted list, you need to run `sudo update-ca-certificates`, which will read the CA certificate from /usr/local/share/ca-certificates. This needs to be done for any machine that should trust this webserver. We've only done it on the webserver itself (VM5), to demonstrate, as we don't plan to be accessing the webserver from any of our other VMs.
Once everything is configured, run `sudo systemctl restart nginx` for Nginx to start using the new certificates.

Apache

The user homepages are hosted on the Apache server.

Install apache2 with `sudo apt install apache2`.
Before you start it, you need to configure what ports Apache will listen to, in order to avoid conflicts with the Nginx server. Since Nginx occupies port 80 and port 443, we gave Apache port 8080. Open /etc/apache/ports.conf and comment out all the current settings. Add the line `Listen 127.0.0.1:8080`

Now, enable and start Apache:

```
sudo systemctl enable apache2
sudo systemctl start apache2
```

To pass requests to the Apache server through Nginx, add the following location block to the configuration of vmpsateam10-05.psa-team10.cit.tum.de in /etc/nginx/sites-available/default:

```
        # proxy to apache server port
        location ~ ^/~[a-z0-9]+/ { # only pass requests that match the
homepage pattern
                allow all;

                proxy_pass http://127.0.0.1:8080; # pass to this port on
localhost

                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header X-Forwarded-Proto $scheme;

        }
```

Next, add these lines to the Apache configuration in /etc/apache2/sites-available/000-default.conf:

```
# inserts <username> at *
UserDir /home/*/.html-data
UserDir /home/*/.cgi-bin

<Directory "/home/*/.html-data">
        Options +SymlinksIfOwnerMatch
        # Allow accesses by everyone
        Require all granted
</Directory>


<Directory "/home/*/.cgi-bin">
        Options +ExecCGI +SymlinksIfOwnerMatch
        SetHandler cgi-script
        Require all granted
</Directory>
```

With this, you specify user directories. You might also need to run `sudo a2enmod userdir` for this to work. Next, you need to add rules for rewriting the URL to get the actual file path to the user homepage files. Add this line `RewriteEngine on` to 000-default conf. Then, create a VirtualHost block with these contents:

```
<VirtualHost 127.0.0.1:8080>

        ServerName vmpsateam10-05.psa-team10.cit.tum.de

        # if request ends in cgi-bin
        RewriteCond %{REQUEST_URI} ^/~[a-zA-Z0-9]+/cgi-bin/
        # translate to filepath .../username/.cgi-bin
        RewriteRule "^/~([a-zA-Z0-9]+)/cgi-bin/(.+)$"  "/~$1/.cgi-bin/$2"
[S=1,PT]
        # if not, same thing but .html-data (html-data is the default, does
not have to be specified in
url)
        RewriteRule "^/~([a-zA-Z0-9]+)/(.*)"   "/~$1/.html-data/$2"

        # record warnings and anything more serious
        LogLevel warn
</VirtualHost>
```

In order to run dynamic contents as the user who owns the script, you also need to add these lines to the VirtualHost block:

```
        # Suexec
        UserDir disabled root
        UserDir /home
```

and this line outside of it:

```
Suexec On
```

All that's left is to generate the contents of the homepages. For this, we used the following bash script:

```bash
#!/bin/bash

filepath="/home/patricia/create_user_homepages"
template="$filepath/template"



users=("schoe" "steph" "pahll" "frisc" "zette" "klaku" "dietr" "tongu" "haugs"
"yesse" "stoec" "wittm" "grote" "deike" "meuse" "schub" "wothg" "songl"
"horva" "kastl")

for user in "${users[@]}"; do

    # directories need to be accessible by the webserver
    sudo chmod +rx "/home/$user"
    sudo mkdir "/home/$user/.html-data"
    sudo mkdir "/home/$user/.cgi-bin"
    sudo chmod +rx "/home/$user/.html-data"
    sudo chmod +rx "/home/$user/.cgi-bin"

    # create html for user homepage
    # sed subtitutes USERNAME in the html file with actual name
    user_html="/home/$user/.html-data/index.html"
    sudo sed "s/USERNAME/$user/g" "$template" | sudo tee "$user_html" >
/dev/null

    # create dynamic content
    user_script="/home/$user/.cgi-bin/print_time.py"
    sudo cp "$filepath/print_time.py" "$user_script"

    # permissions. Scripts will be executed by the owner set here!
    sudo chown $user:$user "/home/$user/.cgi-bin/"
    sudo chown $user:$user "/home/$user/.html-data/"
    sudo chown $user:$user $user_html
    sudo chown $user:$user $user_script
    sudo chmod 755 "$user_script"

done
```

"template" and "print_time.py" are a basic HTML file and python script we created for each user. The HTML template is personalized to include the user's name. Restart Apache, and the user homepages should be reachable through URLS like https://vmpsateam10-05.psa-team10.cit.tum.de/~username or https://vmpsateam10-05.psa-team10.cit.tum.de/~username/cgi-bin/print_time.py.

Logfiles

Nginx

Open /etc/nginx&nginx.conf and add the following to the http block, then restart :

```
# specify custom format
log_format access '$remote_user [$time_local] "$request" '
              '$status $body_bytes_sent "$http_referer" '
              '"$http_user_agent"';

# specify where to use it
access_log /var/log/nginx/access.log access;

error_log warn;
```

Apache

Open /etc/apache/apache2.conf and add the following lines, then restart:

```
# create custom log format for access logs, which doesn't contain any IP
addresses
LogFormat "%l %u %t \"%r\" %>s %b" access

# tell apache to use this format for access logs
CustomLog "/var/log/apache2/access.log" access
```

```
# set error log format which includes client IP address
ErrorLogFormat "[%t] [%l] [pid %P] %F: %E: [client %a] %M"
```

Log Rotation

We used logrotate for logrotation. There are already pre-existing logrotate config files for apache2 and nginx, located in /etc/logrotate.d. As a first step, navigate to this directory.

Now, open apache2 with a texteditor and change the contents to the following:

```
/var/log/apache2/access.log {
    daily
    missingok
    rotate 5
    maxage 5
    compress
    delaycompress
    notifempty
    create 640 root adm
    sharedscripts
    prerotate
        if [ -d /etc/logrotate.d/httpd-prerotate ]; then
            run-parts /etc/logrotate.d/httpd-prerotate
        fi
    endscript
    postrotate
        if pgrep -f ^/usr/sbin/apache2 > /dev/null; then
            invoke-rc.d apache2 reload 2>&1 | logger -t apache2.logrotate
        fi
    endscript
}

/var/log/apache2/error.log {
    daily
    missingok
    rotate 3
    maxage 1
    compress
    delaycompress
    notifempty
    create 640 root adm
    sharedscripts
    prerotate
        if [ -d /etc/logrotate.d/httpd-prerotate ]; then
            run-parts /etc/logrotate.d/httpd-prerotate
        fi
    endscript
    postrotate
        if pgrep -f ^/usr/sbin/apache2 > /dev/null; then
            invoke-rc.d apache2 reload 2>&1 | logger -t apache2.logrotate
        fi
    endscript
}

/var/log/apache2/other_vhosts_access.log {
    daily
    missingok
    rotate 5
    maxage 5
    compress
    delaycompress
    notifempty
    create 640 root adm
    sharedscripts
    prerotate
        if [ -d /etc/logrotate.d/httpd-prerotate ]; then
            run-parts /etc/logrotate.d/httpd-prerotate
        fi
    endscript
    postrotate
        if pgrep -f ^/usr/sbin/apache2 > /dev/null; then
            invoke-rc.d apache2 reload 2>&1 | logger -t apache2.logrotate
        fi
    endscript
}



/var/log/apache2/suexec.log {
    daily
    missingok
    rotate 3
    maxage 1
    compress
    delaycompress
    notifempty
    create 640 root adm
    sharedscripts
    prerotate
        if [ -d /etc/logrotate.d/httpd-prerotate ]; then
            run-parts /etc/logrotate.d/httpd-prerotate
        fi
    endscript
    postrotate
        if pgrep -f ^/usr/sbin/apache2 > /dev/null; then
            invoke-rc.d apache2 reload 2>&1 | logger -t apache2.logrotate
        fi
    endscript
}
```

These configuration blocks are closely based on what was originally in this file. We only made these few changes:

- made separate block for each type of log file (by changing the name)
- decreased rotate; rotate indicates how many times a log file can be rotated before being removed
- added maxage; this indicates how many days a log file is kept before it is deleted; in our case 5 days for access logs and 1 day for error logs

Next, do the same thing for nginx. Open /etc/logrotate.d/nginx and give it these contents:

```
/var/log/nginx/access.log {
        daily
        missingok
        rotate 5
        maxage 5
        compress
        delaycompress
        notifempty
        create 0640 www-data adm
        sharedscripts
        prerotate
                if [ -d /etc/logrotate.d/httpd-prerotate ]; then \
                        run-parts /etc/logrotate.d/httpd-prerotate; \
                fi \
        endscript
        postrotate
                invoke-rc.d nginx rotate >/dev/null 2>&1
        endscript
}


/var/log/nginx/error.log {
        daily
        missingok
        rotate 3
        maxage 1
        compress
        delaycompress
        notifempty
        create 0640 www-data adm
        sharedscripts
        prerotate
                if [ -d /etc/logrotate.d/httpd-prerotate ]; then \
                        run-parts /etc/logrotate.d/httpd-prerotate; \
                fi \
        endscript
        postrotate
                invoke-rc.d nginx rotate >/dev/null 2>&1
        endscript
}
```

Once that is done, run `sudo logrotate -vf <config file>` once for the apache and nginx config file, to see if it's working. After that, it should run automatically every day.

Testing

```bash
#!/bin/bash

failed_tests=0

fail() {
    ((failed_tests++))
    echo "FAIL $@"
}

ok() {
    echo "OK $@"
}

request() {
    response=$(curl -s -o /dev/null -w "%{http_code}" --noproxy "*" -L -k
"$1")
    if [ "$response" -eq "$2" ]; then
        ok "$1"
    else
        fail "$1 | status: $response"
    fi
}

https_urls=("https://vmpsateam10-05.psa-team10.cit.tum.de" "https://www.psa-
team10.cit.tum.de/"
"https://vm5-other.psa-team10.cit.tum.de/" "https://vmpsateam10-05.psa-
team10.cit.tum.de/~kastl/"
"https://vmpsateam10-05.psa-team10.cit.tum.de/~kastl/cgi-bin/print_time.py")
http_urls=("http://vmpsateam10-05.psa-team10.cit.tum.de/" "http://www.psa-
team10.cit.tum.de/"
 "http://vm5-other.psa-team10.cit.tum.de/")


for url in "${https_urls[@]}"; do
    request $url "200"
done


for url in "${http_urls[@]}"; do
    request $url "200"
done


echo "Failed tests: $failed_tests"
```