ComS 252 Homework 10: Web server

Group assignment (with 5% penalty per group member)

Due November 9, 2021

1 Objectives

For this assignment, you will configure network settings and build a web server, in Linux. For more information, consult

- Chapter 17 of the textbook.
- Chapter 26 of the textbook.

2 Download

Download the virtual machine `Hw10.ova`. Accounts are `root` and `user`, with passwords `rootpw` and `userpw`, as usual. You should not need to install any packages for this assignment.

Currently, the virtual machine has one virtual network adapter enabled, with type "NAT". This is used to connect to the Internet, and is required for submitting your work. Do not change this. You will add and configure a second network adapter for a virtual network between your host machine (i.e., whatever machine you are using to run VirtualBox) and the virtual machine. This way, you can test the web server on the VM by running a browser on your *host* machine[1].

3 Configuring the VM

3.1 Build a virtual network

In the VirtualBox menu, select "File" and then "Host Network Manager". Add a "host–only network" here (or use one that already exists), and make a note of its name. This is a virtual private network that VMs and your host machine can connect to. Next, you need to configure the network. Make a note of the IPv4 address and network mask. The IPv4 address will be the address of your *host* machine on this network; this will be referred to later as `www.xx.yy.h`. Also, make sure the "DHCP server" is *disabled* for this network. You will set up the VM to use a static IP address on this network.

3.2 Connect the VM to the network

Go to the settings for the VM for this assignment, and under "Network", enable "Network Adapter 2". The network type should be "Host–only adapter", and be sure that the network name matches the one you just set up. Make a note of the MAC addresses for both network adapters.

4 Network configuration in Linux

The following will require you to use a text editor to edit the network interface configuration files in directory `/etc/sysconfig/network-scripts`.

[1]How cool is this?

4.1 Network adapter 1

The first network adapter should already be configured, on interface `enp0s3`. However, it was configured for a machine with only one network interface. Edit the configuration file for interface `enp0s3`, and specify the MAC address for the adapter (this is necessary for machines with more than one network interface), by adding a line of the form

```
HWADDR=xx:xx:xx:xx:xx:xx
```

where "`xx:xx:xx:xx:xx:xx`" is replaced by the MAC address for Network Adapter 1.

4.2 Network adapter 2

The second network adapter is partially onfigured on interface `enp0s8`. Edit the configuration file for this interface, by adding or updating the following entries.

```
ONBOOT=yes
IPADDR=www.xx.yy.z
NETMASK=nnn.nnn.nnn.nnn
HWADDR=yy:yy:yy:yy:yy:yy
```

Note that `www.xx.yy.h` is the *host machine* IP address on the host–only network, and you must use a different IP address `www.xx.yy.z`, but one on the same subnet, for the VM. Replace "`yy:yy:yy:yy:yy:yy`" with the MAC address for Network Adapter 2.

4.3 Testing

Reboot the virtual machine. Run `ifconfig -a` to check that all network interfaces are up. In particular, check for `inet` addresses for interfaces `enp0s3` and `enp0s8`. Interface `enp0s8` should have the static IP address `www.xx.yy.z` that you specified in the configuration file.

Verify that `enp0s3` is still working correctly. Since `enp0s3` is the interface that connects the VM to the Internet, you can do this with `ping google.com`. If packets can reach `google.com`, then `enp0s3` is configured correctly. You need this to work correctly for submitting your work.

On your host machine, start a shell[2] and try both

- `ping www.xxx.yy.h`

  (the host sending packets to itself) and

- `ping www.xxx.yy.z`

  (the host sending packets to the VM on network adapter 2).

Again, note that IP addresses `www.xxx.yy.h` and `www.xxx.yy.z` are *different*, but on the same subnet. If packets can get through, then you are ready to configure the web server.

5 Basic Web server

First, you will configure a basic web server on the VM, and see how to access it from a browser on the host machine.

1. Start the `httpd` service, and set `httpd` to start at boot time using `systemctl`.

2. You can test the server locally (on the VM) with `curl` or `lynx` and using the URL `http://localhost`. This will fail with a message like "unable to connect to remote host" if the server is not working; otherwise, you should see a test page that says that the server is working but has not been configured yet. (In `lynx`, the test page appears a few seconds after a "403 Forbidden" alert, so be patient!)

[2]This works in Windows, Linux, and Mac.

3. Configure the firewall so that HTTP packets are accepted (in the default "zone"). This change should be permanent (i.e., it should persist when the VM is rebooted). You may disable the firewall if necessary for testing, but in the end you should have the firewall running and allowing HTTP packets through.

4. You should now be able to open a browser on the host machine, and using URL `http://www.xx.yy.z` (use the IP address of network adapter 2 on the VM), you should obtain the same test page that you saw on the virtual machine.

5. Access logs are kept under `/var/log/httpd` on the virtual machine; these can be *extremely* helpful for server debugging.

6. You should **NOT** need to update the `httpd` configuration file(s) for this assignment.

From this point on, all testing of the server may be done with a browser in the host machine, instead of in the virtual machine.

6 Static content

The Apache configuration page will indicate where static content should be placed; usually this is `/var/www/html`. Create an example HTML file here, with name `index.html`. This should be visible at URL `http://www.xx.yy.z/`. Some example HTML:

```
<html>
<title>Foomatic Industries, Inc.</title>
<body>
<h1>Foomatic Industries</h1>
<p>
Foomatic Industries is a wholly-owned subsidary of Foocorp,
the world's leader in the production of foo.
</p> </body> </html>
```

7 Dynamic content

Web pages can be dynamically created by the server. One way to do this is to write a bash[3] script (or other executable) whose output becomes the web page sent by the server. These are called "CGI scripts". For this assignment, your CGI scripts should always output one of the two following partial headers, followed by a blank line:

- `Content-type:  text/plain`

  which indicates that the rest of the script output should be interpreted as plain text.

- `Content-type:  text/html`

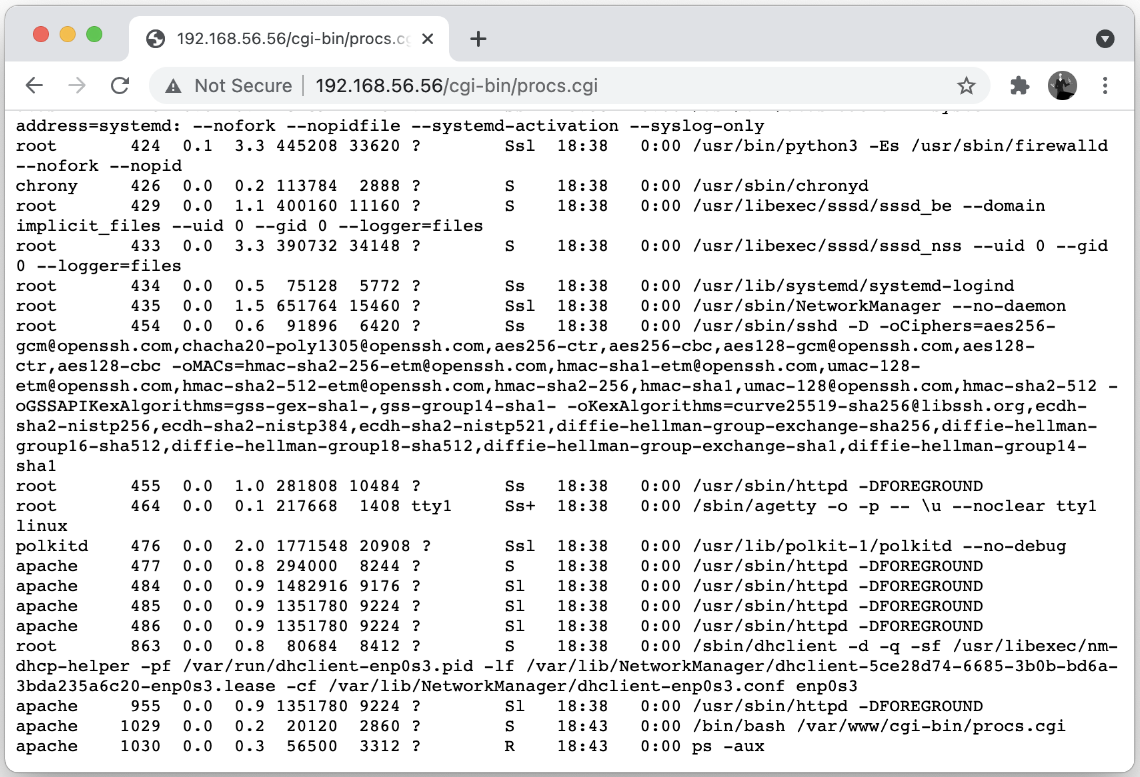
  which indicates that the rest of the script output should be interpreted as HTML.

Your CGI scripts should be placed in directory `/var/www/cgi-bin`, and should always be bash scripts.

1. To begin, create a simple "hello world" script that simply generates the text "Hello, world". You should test both a plain text version of the script:

   ```
   #! /bin/bash
   #
   echo "Content-type: text/plain"
   echo
   echo "Hello, world!"
   ```

[3]Bash is probably not the best choice, but we cover it in class. Perl, Python, or PHP is often a better choice.


192.168.56.56/cgi-bin/procs.cg
Not Secure
192.168.56.56/cgi-bin/procs.cgi

```
address=systemd: --nofork --nopidfile --systemd-activation --syslog-only
root        424  0.1  3.3 445208 33620 ?        Ssl  18:38   0:00 /usr/bin/python3 -Es /usr/sbin/firewalld
--nofork --nopid
chrony      426  0.0  0.2 113784  2888 ?        S    18:38   0:00 /usr/sbin/chronyd
root        429  0.0  1.1 400160 11160 ?        S    18:38   0:00 /usr/libexec/sssd/sssd_be --domain
implicit_files --uid 0 --gid 0 --logger=files
root        433  0.0  3.3 390732 34148 ?        S    18:38   0:00 /usr/libexec/sssd/sssd_nss --uid 0 --gid
0 --logger=files
root        434  0.0  0.5  75128  5772 ?        Ss   18:38   0:00 /usr/lib/systemd/systemd-logind
root        435  0.0  1.5 651764 15460 ?        Ssl  18:38   0:00 /usr/sbin/NetworkManager --no-daemon
root        454  0.0  0.6  91896  6420 ?        Ss   18:38   0:00 /usr/sbin/sshd -D -oCiphers=aes256-
gcm@openssh.com,chacha20-poly1305@openssh.com,aes256-ctr,aes256-cbc,aes128-gcm@openssh.com,aes128-
ctr,aes128-cbc -oMACs=hmac-sha2-256-etm@openssh.com,hmac-sha1-etm@openssh.com,umac-128-
etm@openssh.com,hmac-sha2-512-etm@openssh.com,hmac-sha2-256,hmac-sha1,umac-128@openssh.com,hmac-sha2-512 -
oGSSAPIKexAlgorithms=gss-gex-sha1-,gss-group14-sha1- -oKexAlgorithms=curve25519-sha256@libssh.org,ecdh-
sha2-nistp256,ecdh-sha2-nistp384,ecdh-sha2-nistp521,diffie-hellman-group-exchange-sha256,diffie-hellman-
group16-sha512,diffie-hellman-group18-sha512,diffie-hellman-group-exchange-sha1,diffie-hellman-group14-
sha1
root        455  0.0  1.0 281808 10484 ?        Ss   18:38   0:00 /usr/sbin/httpd -DFOREGROUND
root        464  0.0  0.1 217668  1408 tty1     Ss+  18:38   0:00 /sbin/agetty -o -p -- \u --noclear tty1
linux
polkitd     476  0.0  2.0 1771548 20908 ?       Ssl  18:38   0:00 /usr/lib/polkit-1/polkitd --no-debug
apache      477  0.0  0.8 294000  8244 ?        S    18:38   0:00 /usr/sbin/httpd -DFOREGROUND
apache      484  0.0  0.9 1482916 9176 ?        Sl   18:38   0:00 /usr/sbin/httpd -DFOREGROUND
apache      485  0.0  0.9 1351780 9224 ?        Sl   18:38   0:00 /usr/sbin/httpd -DFOREGROUND
apache      486  0.0  0.9 1351780 9224 ?        Sl   18:38   0:00 /usr/sbin/httpd -DFOREGROUND
root        863  0.0  0.8  80684  8412 ?        S    18:38   0:00 /sbin/dhclient -d -q -sf /usr/libexec/nm-
dhcp-helper -pf /var/run/dhclient-enp0s3.pid -lf /var/lib/NetworkManager/dhclient-5ce28d74-6685-3b0b-bd6a-
3bda235a6c20-enp0s3.lease -cf /var/lib/NetworkManager/dhclient-enp0s3.conf enp0s3
apache      955  0.0  0.9 1351780 9224 ?        Sl   18:38   0:00 /usr/sbin/httpd -DFOREGROUND
apache     1029  0.0  0.2  20120  2860 ?        S    18:43   0:00 /bin/bash /var/www/cgi-bin/procs.cgi
apache     1030  0.0  0.3  56500  3312 ?        R    18:43   0:00 ps -aux
```

Figure 1: Example of `procs.cgi` in a browser

and an HTML version of the script:

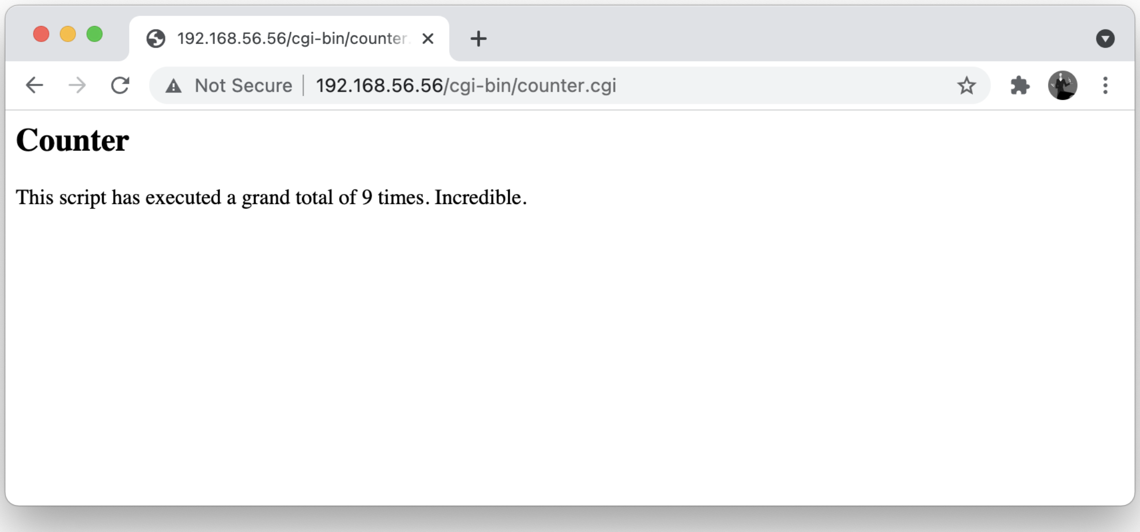
```
#! /bin/bash
#
echo "Content-type: text/html"
echo
echo "<h2>Hello, world!</h2>"
```

Save the file as `hello.cgi` and turn on execute permission. You should then be able to view it at URL `http://www.xx.yy.z/cgi-bin/hello.cgi`, which will execute the script on the server, with the script output displayed in the browser. Of course, you can also execute the script in a shell on the server, if you need to debug its output.

2. Create a CGI script named `time.cgi` that prints the current date and time, as HTML. Verify that the time changes in a browser when the page is reloaded.

3. Create a CGI script named `procs.cgi` that displays all running processes (use `ps -aux`). This script should produce plain text. Check the script in a browser. An example is shown in Figure 1; note that refreshing the page will cause the bottom two processes, which are running because of the CGI script, to change.

4. Create a CGI script named `env.cgi` that displays the environment variables (using `env`) passed to the script. This script should produce plain text.


192.168.56.56/cgi-bin/counter
Not Secure
192.168.56.56/cgi-bin/counter.cgi

Counter

This script has executed a grand total of 9 times. Incredible.

Figure 2: Example of `counter.cgi` in a browser

5. Create a CGI script named `counter.cgi` that produces an HTML page that keeps track of the total number of times the script has been executed. You will need to use text file to save the value of the counter. For full credit, the counter's text file should *not* be world-writable. For this assignment, do not worry about locking the counter's text file to prevent simultaneous changes. The actual count is unimportant, as long as the count increments each time the script runs in a browser on the host machine. Figure 2 shows an example; refreshing the page causes the counter to increment.

8 Submitting your work

Login as `root`, and run `Turnin yourISUusername` to automatically submit your work. If you worked in a group, run `Turnin` once with the usernames of everyone in your group. Check the `man` page for `Turnin` for more information.